REDUCING CHURN: PREDICTING AND INTERVENING IN ACCOUNT CANCELLATIONS WITH MACHINE LEARNING & ARTIFICIAL INTELLIGENCE SYSTEMS

Tyler Dial[1,2] and Tejaswi Gorde[1]

tyler@dialedintelligence.com
tejaswigorde2025@u.northwestern.edu
tylerdial2026@u.northwestern.edu

[1]Northwestern University & [2]Dialed Intelligence
Master of Science in Data Science Program
March 15, 2026

Abstract

Subscription streaming platforms lose customers at monthly rates between 5% and 7%, yet most churn models stop at prediction and offer no pathway from risk score to retention action. RETAIN, an end-to-end churn intelligence platform, addresses this gap by unifying predictive modeling, interpretable feature attribution, and agentic AI-driven intervention in a single deployed system. The platform operates on a unified, account-level dataset of approximately 50,000 subscribers. This dataset integrates customer demographics, subscription lifecycle data, multi-window engagement metrics, payment reliability indicators, device usage patterns, and customer support interactions. A behavioral simulation generates the data, reproducing the statistical structure of real streaming service data while providing full control over ground truth. A four-stage feature selection pipeline distills 30 engineered features into a final set of 17. Evaluation of twelve supervised learning models on a common held-out test set identifies tuned XGBoost as the top performer, with a Receiver Operating Characteristic Area Under the Curve (ROC-AUC) of 0.9747 and a Precision-Recall AUC (PR-AUC) of 0.9431. SHapley Additive exPlanations (SHAP) analysis reveals that disengagement signals dominate predictions: days since last payment accounts for 39.5% of total feature impact, and days since last stream accounts for an additional 18.4%. The production model flags 2,496 of 49,684 active accounts (5.02%) as high risk for churn within 30 days. An agentic intervention layer built with LangGraph extends the system beyond prediction through four coordinated workflows, each surfaced through a React and FastAPI web application designed around the natural decision flow of a retention analyst. These results demonstrate that churn analytics can evolve from descriptive reporting into a scalable, automated retention system when prediction, explanation, and action are integrated end to end.

Keywords: *customer churn prediction, machine learning, streaming services, MLOps, feature engineering, gradient boosting, explainable AI, customer retention, synthetic data, subscription analytics*

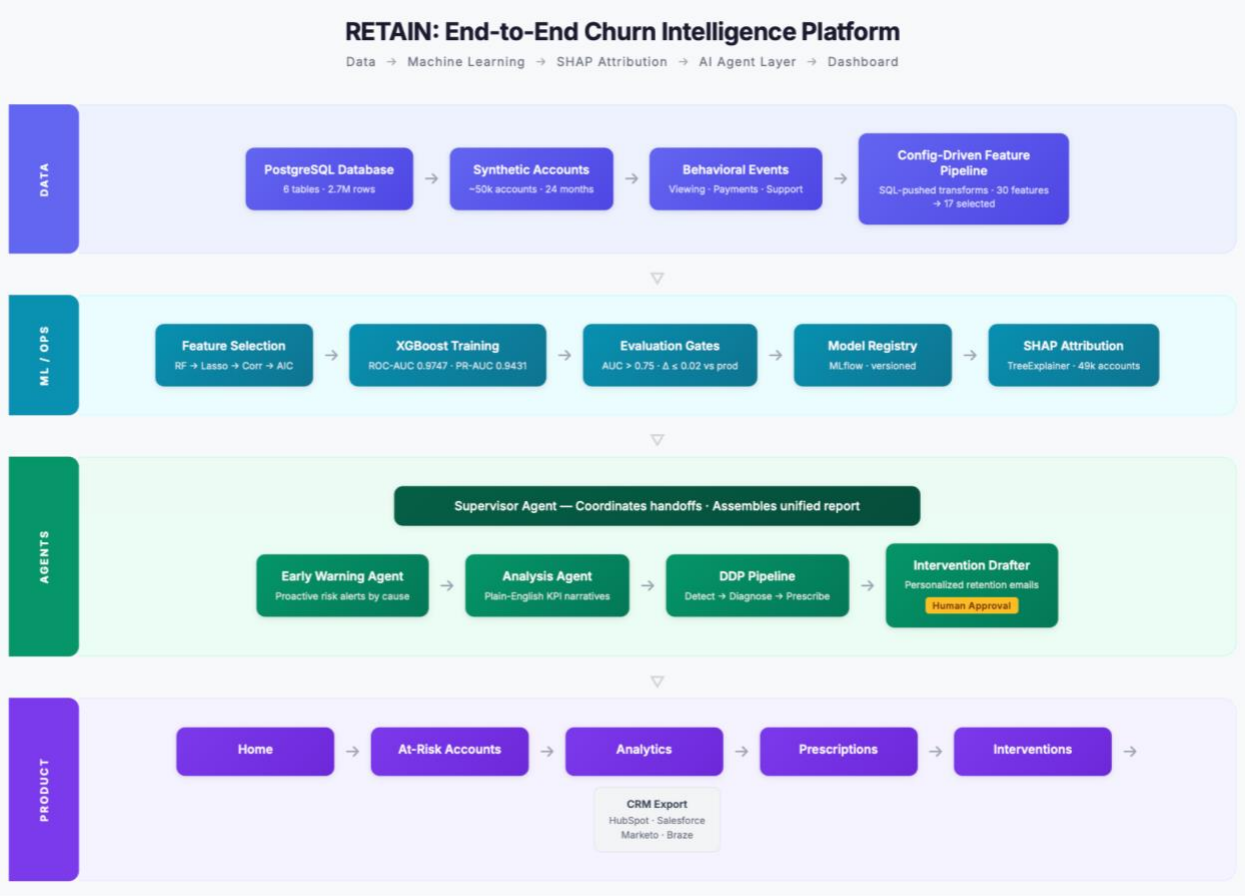
Table of Contents

1. Introduction

Subscription streaming platforms face a persistent operational problem: customers disengage gradually (reducing watch time, missing payments, contacting support) and then cancel. Most analytics systems detect this pattern only after the cancellation has already occurred. The gap between available behavioral signal and timely intervention represents a significant missed opportunity. Industry churn rates of 5%–7% per month translate directly into lost recurring revenue, diminished customer lifetime value, and compounding acquisition costs (Reichheld and Sasser 1990). Existing churn prediction models, while increasingly accurate, typically end at a risk score. The translation from prediction to personalized retention action falls entirely to human operators who lack the time or tooling to act at scale.

This paper introduces RETAIN, an end-to-end churn intelligence platform that unifies machine learning prediction, interpretable feature attribution via SHAPley Additive explanations (SHAP), and an agentic artificial intelligence (AI) intervention layer into a single deployed system for subscription streaming services. Rather than treating churn as a standalone binary classification task, RETAIN connects every stage of the retention workflow including detection, diagnosis, prescription, and personalized outreach within one application. Architectural coupling distinguishes RETAIN from prior work: a high-accuracy predictive model (tuned eXtreme Gradient Boosting, or XGBoost; Receiver Operating Characteristic Area Under the Curve, or ROC-AUC, of 0.9747) feeds directly into a multi-agent orchestration layer built on LangGraph. Each agent handles one step of the analyst's natural decision process, transforming a static risk score into a dynamic, actionable retention pipeline.

The platform operates on a unified, account-level dataset of approximately 50,000 synthetic subscriber accounts stored in a normalized PostgreSQL schema with six primary tables. A behavioral simulation generates these accounts rather than simple random sampling. Each simulated customer produces correlated event sequences: viewing sessions, payment transactions, support interactions, and device usage. Underlying behavioral models govern these sequences to reproduce the statistical structure of real streaming service data. This design provides control over data quality and ground truth while producing a dataset realistic enough to support meaningful model evaluation. The inherent limitation of encoding assumed rather than discovered relationships is discussed in Section 6.1.



A four-stage feature selection pipeline distills 30 engineered features into a final 17-feature set. The pipeline applies Random Forest importance ranking, L1 regularization, correlation filtering, and backward Akaike Information Criterion (AIC) elimination in sequence. Twelve supervised learning models span logistic regression, tree-based methods, ensemble techniques, and deep learning architectures. All twelve are evaluated on a common stratified holdout set.

The tuned XGBoost model achieves the top ROC-AUC of 0.9747 and Precision-Recall AUC (PR-AUC) of 0.9431. SHAP analysis reveals that two disengagement signals dominate: days since last payment accounts for 39.5% of total feature impact, and days since last stream accounts for 18.4%. Together, these two features explain nearly 58% of the model's predictive behavior across 49,684 active accounts. The production model flags 2,496 accounts (5.02%) as high risk for churn within the next 30 days.

Beyond prediction, RETAIN introduces an agentic intervention layer built with LangGraph. Four coordinated workflows compose this layer: an Early Warning Agent for proactive risk monitoring, an Analysis Agent for natural-language reporting, a Detect-Diagnose-Prescribe multi-agent pipeline for root cause analysis and intervention recommendation, and an Intervention Drafter Agent for personalized retention email generation.

A core design principle governs all agent workflows: data retrieval and computation remain deterministic. The AI model handles only natural language generation. This separation kept the production pipeline focused on its core function of training and registering the selected model while providing the freedom to experiment with architectures that would be impractical to route through an automated system, such as deep learning models with long training times or ensemble methods requiring multiple base learners.

Refer to Appendix A for a full set of Tables, Visualizations, and Model Results, and refer to Appendix B for a glossary of acronyms and key terms used throughout this report. The remainder of this paper is organized as follows. Section 2 reviews related work on churn modeling, classification methods, and Machine Learning Operations (MLOps) infrastructure. Section 3 describes the synthetic data generation strategy and database structure. Section 4 details the feature engineering pipeline, feature selection process, modeling strategy, and evaluation metrics. Section 5 presents the results, including model comparison, computational efficiency analysis, SHAP-based feature attribution, and the 30-day churn forecast. Section 6 discusses limitations of synthetic data, the AI agent workflows, the RETAIN dashboard, and broader implications for data science practice. Section 7 offers conclusions, and Section 8 identifies directions for future work.

2. Literature Review

2.1 Churn

Churn is when a customer stops being a customer. For subscription businesses like streaming services, this typically means a cancelled subscription. But this straightforward definition masks complexity in practice. Customers don't always churn suddenly. They often disengage, reducing their usage gradually before leaving. Understanding this trajectory is what separates effective churn prediction from simply counting how many customers canceled their service.

The economics of churn prevention are important to almost any business that relies on recurring revenue. Research by Reichheld and Sasser (1990) established the now-famous finding that a 5% improvement in customer retention can increase profits by 25% to 95%, depending on the industry. Gupta et al. (2004) demonstrated that even a 1% improvement in retention has nearly five times the profit impact of a 1% improvement in acquisition costs or margins. For streaming services, where the industry average monthly churn rate hovers between 5% and 7%, these percentages translate into substantial revenue.

2.2 Classification Models for Churn

Churn prediction is a binary classification problem, and approaches have evolved from simple statistical methods to complex ensembles. Logistic regression remains relevant for its interpretability. Neslin et al. (2006) found that while sophisticated methods achieved higher accuracy, logistic regression provided predictions business users could actually understand. For tabular data like customer features, gradient boosting has become the dominant approach. Rather than training trees independently as Random Forests do, gradient boosting builds trees sequentially, with each new tree correcting errors of previous ones.

Recent work has pushed in two directions: deep learning and explainability. Joy et al. (2024) developed a hybrid model combining LSTM and GRU networks with LightGBM specifically for streaming services, achieving 95.6% AUC by capturing temporal patterns in viewing behavior. Liu et al. (2024) proposed CCP-Net, which uses multi-head self-attention combined with BiLSTM and CNN layers to extract both sequential and local features from customer data. Meanwhile, Chang et al. (2024) emphasize that prediction accuracy alone isn't sufficient. Their work integrates SHAP and LIME to make ensemble model decisions interpretable for business stakeholders. The production model in this work incorporates SHAP evaluation metrics to determine feature importance and strengthen interpretability.

2.3 Machine Learning Operations and Production Systems

Building an accurate model is only part of the challenge. Making it useful requires infrastructure for training, deployment, monitoring, and continuous improvement. This is the discipline of MLOps.

Model degradation presents a particularly important challenge. Unlike traditional software that either works or doesn't, machine learning models can silently degrade as the

relationship between features and outcomes shifts. A churn model trained on 2023 behavior may perform poorly in 2025 if viewing patterns or competitive dynamics have changed. Monitoring for data drift and prediction drift enables early detection of these issues (Klaise et al. 2020). The concept of evaluation gates addresses when a model is good enough to deploy. Rather than manually reviewing metrics, automated gates compare new models against thresholds and against the currently deployed version (Amershi et al. 2019).

The architecture adopted for this project uses configuration-driven pipelines, automated quality checks, and monitoring infrastructure that would support ongoing model improvement in a production environment.

3. Data

3.1 Synthetic Data Generation Strategy

Rather than relying on a small, anonymized dataset, the project generates synthetic data that captures the statistical properties and relationships expected in actual streaming service data. This approach gives control over data quality and ground truth while generating enough volume for real experimentation.

The synthetic data generation follows a simulation approach rather than simple random sampling. The generation process begins with customer accounts whose demographic attributes are drawn from distributions that approximate U.S. streaming service demographics. Each account then generates behavioral events over time: viewing sessions, content completions, pause and resume patterns, device switches, and support interactions. These events aren't independent random variables. They're generated by underlying behavioral models we chose to create realistic correlations and patterns in the data.

For example, a customer who watches many hours in their first month is more likely to continue watching in subsequent months. A customer who contacts support multiple times is more likely to have lower satisfaction and higher churn probability. A customer approaching the end of their annual subscription exhibits different behavior than one in the middle of their term. This process produces data that, while synthetic, exhibits a fairly realistic structure of real streaming customers.

3.2 Database Structure

The synthetic data is stored in PostgreSQL using a normalized schema with six primary tables. This design reflects how production streaming services typically organize their data, with separate tables for accounts, subscriptions, viewing events, support interactions, devices, and content metadata.

The accounts table contains approximately 50,000 unique customers with attributes including account creation date, demographic information, and current status. Behavioral data lives primarily in the viewing_sessions table, which records each viewing event with timestamps, duration, content identifiers, completion percentage, and the device used. This table contains the largest volume of records (over 2 million rows) and serves as the foundation for engagement features. The support_tickets table captures customer service interactions, including issue category, resolution status, and customer satisfaction ratings when available. The devices table tracks the devices registered to each account and their usage patterns, and the content_catalog table provides metadata about available content allowing us to capture content and genre preferences.

In total, the database contains approximately 2.7 million rows across all tables. The normalized structure requires joins during feature computation but provides flexibility for different analytical approaches. The goal was to create a database system that felt realistic in comparison to the production data pipelines used by top streaming services like Netflix, Hulu, and HBO Max.

4. Methods

4.1 Feature Engineering Pipeline

Raw database records don't feed directly into machine learning models. The viewing sessions, support tickets, and subscription records must be transformed into numerical features that capture meaningful patterns about customer behavior and churn risk. This transformation happens in the feature engineering pipeline, which converts the event-level records into 30 features per account.

Rather than hardcoding feature definitions in Python, the pipeline specifies features in configuration files that define the source tables, aggregation logic, and time windows for each feature group. This design makes it easier for to add new features without modifying core

pipeline code. Each transformer function generates SQL queries that PostgreSQL executes to compute aggregations, joins, and window functions. This approach handles large data volumes and produces a clear audit trail of how each feature was calculated.

4.2 Model Experimentation

All model experimentation was conducted as a separate ad-hoc analysis outside the production training pipeline. Rather than running twelve candidate models through the CI/CD application, a dedicated Jupyter notebook performed the full model comparison, sharing the same feature set and train-test split but operating independently of the deployment infrastructure. This separation kept the production pipeline focused on its core function, training and registering the selected model, while giving us the freedom to experiment with architectures that would be impractical to route through an automated system, such as deep learning models with long training times or ensemble methods requiring multiple base learners.

4.3 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is conducted to validate the structure and realism of the synthetic dataset and to understand the behavioral patterns present before model development. The analysis focuses on examining distributions, missing values, and overall data quality across engagement, subscription, and support-related variables. Particular attention is given to viewing behavior metrics such as watch time, session frequency, and completion rates to ensure that engagement patterns reflect realistic streaming usage and provide meaningful signals for churn analysis.

EDA also explores relationships between customer behavior and churn outcomes by analyzing differences across tenure groups, engagement levels, and support interactions. Temporal analysis is used to identify behavioral trends leading up to churn events, such as declining activity or increased service friction. Correlation analysis helps identify redundant or highly related features, improving feature selection and ensuring that downstream models remain interpretable and stable.

4.4 Churn Labeling

Churn labels are derived primarily from subscription lifecycle data. Accounts with cancelled or expired subscriptions are labeled as churned, while active accounts exhibiting sustained

inactivity, payment failures, or service-level issues are treated as at-risk in intermediate analyses. This approach supports supervised modeling while preserving interpretability and alignment with business definitions of churn.

4.5 Feature Selection

Feature selection follows a four-step pipeline designed to produce a simple, statistically grounded feature set. A Random Forest classifier first ranks all available numeric features by mean decrease in impurity, and the top 25 are retained as a candidate pool. L1-penalized logistic regression (Lasso) is then applied to this pool after standardization, zeroing out features whose predictive contributions overlap with stronger signals. A correlation filter follows, dropping the lower-ranked member of any feature pair with absolute correlation above 0.85 to remove autocorrelation issues that Lasso may not fully resolve. Finally, backward AIC elimination fits a statsmodels Logit on the remaining features and iteratively removes whichever feature most improves AIC until no further improvement is possible. The resulting 17-feature set balances predictive strength with interpretability, and its logistic regression form provides a statistically principled explanation of churn drivers for stakeholder communication.

To validate that the selection process did not drop features with meaningful signal, an ablation study compares Logistic Regression, Tuned XGBoost, and Stacking Ensemble across three feature set configurations: the AIC-selected 17 features, the top-25 Random Forest features, and the full numeric set. Performance was nearly identical across all three configurations for each model, confirming that the selection pipeline successfully captured the signal without meaningful loss. Full numerical results for each configuration are tabulated in (see Appendix A, Figure A2), and bar chart comparisons of ROC-AUC, F1 score, and training time across feature sets are shown in (see Appendix A, Figure A5) and (see Appendix A, Figure A6).

4.6 Modeling Strategy & Experimentation

Multiple supervised learning models are evaluated for churn classification, including baseline logistic regression and tree-based models. Model performance is assessed using ROC-AUC, precision–recall tradeoffs, and business-oriented metrics such as revenue at risk and recall among high-value accounts. Feature importance and model explanations are used to connect predictions to actionable churn drivers.

The experimentation also tests neural networks, ensemble learning methods, and other advanced models to quantify how much predictive power a simpler production model sacrifices. If the accuracy gap between simple and advanced models is small, a lightweight model with straightforward explainability metrics becomes the stronger choice for the CI/CD application.

Model experimentation follows a phased approach that escalates in complexity, beginning with interpretable baselines and concluding with ensemble methods and deep learning. All models are evaluated on the same held-out test set to ensure fair comparison. A single stratified 80/20 train-test split (random_state=42) is created at the start of the experiment and is not modified across any phase. Feature scaling is applied via StandardScaler fit exclusively on the training fold and then applied to the test fold.

Phase 1: Baselines. Logistic Regression (L2 penalty, C=1.0, class_weight=balanced) and a Decision Tree (max_depth=6, class_weight=balanced) establish a performance floor. If a linear model achieves strong AUC, the features carry clear predictive signal. The logistic regression also serves as the primary interpretable baseline in the two-model pattern described in Section 4.6.

Phase 2: Traditional ML. Random Forest (300 estimators, max_depth=12, class_weight=balanced_subsample), Gradient Boosted Trees (sklearn, 200 estimators, learning_rate=0.1), XGBoost (300 estimators, scale_pos_weight to correct imbalance), LightGBM (300 estimators, is_unbalance=True), and a calibrated SVM (RBF kernel, CalibratedClassifierCV with 3-fold CV) are evaluated.

Phase 3: Hyperparameter tuning. RandomizedSearchCV with 60 iterations and 5-fold stratified cross-validation is applied to XGBoost and LightGBM, the two top performers from Phase 2. The search space covers n_estimators, max_depth, learning_rate, subsample, colsample_bytree, min_child_weight, gamma, reg_alpha, and reg_lambda for XGBoost, and an analogous set for LightGBM. Tuning occurs exclusively within the training fold; the test set is not consulted until final evaluation.

Phase 4: Ensemble methods. A soft-voting classifier combines Logistic Regression, Random Forest, and Tuned XGBoost by averaging their predicted probabilities. A stacking classifier uses the same three estimators plus Tuned LightGBM as base learners, with a logistic regression meta-learner trained on 5-fold out-of-fold predictions.

Phase 5: Deep learning. A feedforward neural network implemented in TensorFlow with a Keras wrapper provides a deep learning reference point. The architecture consists of three hidden layers (128→64→32 units) with batch normalization, LeakyReLU activations, and dropout (0.3/0.3/0.2). Class imbalance is addressed through a weighted BCE loss. Training uses Adam (lr=1e-3, weight_decay=1e-4) with a ReduceLROnPlateau scheduler and early stopping with patience of 15 epochs over a maximum of 100 epochs.

4.7 Evaluation Metrics

Every model is evaluated on a consistent set of metrics to support fair comparison across methods and business interpretability:

- **ROC-AUC**: Overall discriminative ability, threshold-independent. Primary metric for model ranking.
- **PR-AUC (average precision)**: More informative than ROC-AUC under class imbalance, directly measuring precision-recall tradeoffs at the churn minority class.
- **F1 score**: Mean of precision and recall at the default 0.5 decision threshold.
- **Training and inference time**: Wall-clock seconds, recorded to contextualize the computational cost of each model in a batch scoring deployment scenario.

4.8 Machine Learning Pipeline

The model training component implements a two-model pattern. For each training run, the pipeline fits XGBoost, selected for its combination of high performance and low compute time, and a logistic regression model on the same feature set. The gradient boosting model maximizes predictive performance, while the logistic regression model provides an interpretable baseline and sanity check. If the complex model doesn't meaningfully outperform the simple one, that's a signal that either the features lack predictive power or something has gone wrong in training.

Evaluation gates act as automatic quality checks that models must pass before deployment. These gates compare model performance against thresholds (AUC must exceed 0.75) and against the currently deployed model (new model AUC must be within 0.02 of production). If a model fails any gate, the pipeline logs the failure and halts deployment, preventing degraded models from reaching production.

The model registry maintains a versioned history of trained models, their evaluation metrics, and the data snapshots used for training. When a new model passes evaluation gates, it's registered with metadata including the training timestamp, feature configuration, hyperparameters, and performance metrics on holdout data.

5. Results

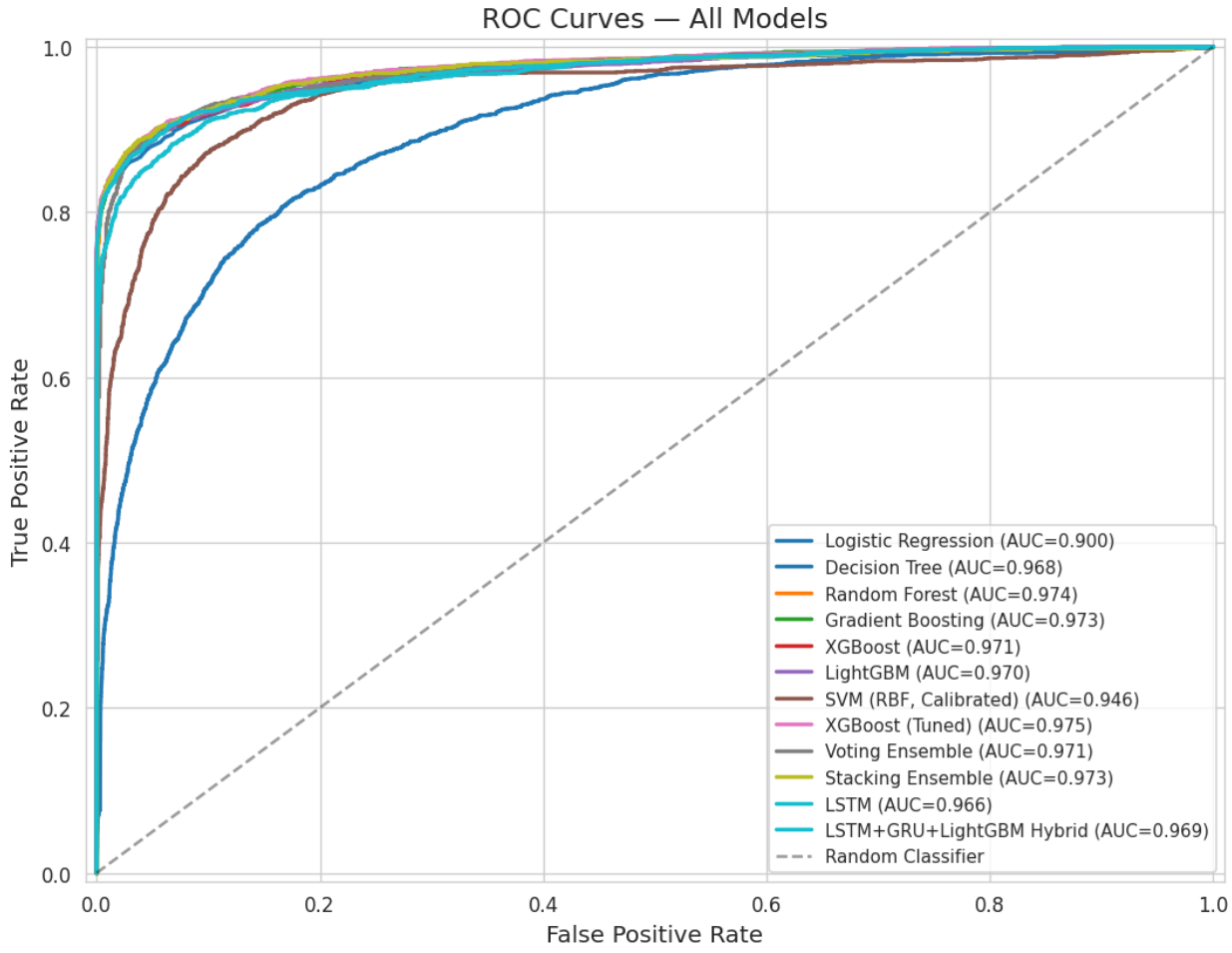
5.1 Model Comparison

Twelve models are evaluated on a common 20% stratified holdout set (12,000 accounts) using ROC-AUC as the primary ranking metric, supplemented by PR-AUC, F1, precision, recall, accuracy, and training time. The full leaderboard is presented in Table 1.

Table 1. Model performance leaderboard, sorted by ROC-AUC. All models evaluated on the same held-out test set.

Model	ROC-AUC	PR-AUC	F1	Recall	Precision	Train Time (s)
XGBoost (Tuned)	0.9747	0.9431	0.8379	0.8972	0.7860	~60 (tuning)
LightGBM (Tuned)	0.9747	0.9430	0.8376	0.8968	0.7858	~60 (tuning)
Stacking Ensemble	0.9742	0.9418	0.8361	0.8941	0.7849	~120+
Voting Ensemble	0.9738	0.9401	0.8344	0.8923	0.7831	~30+
Random Forest	0.9720	0.9380	0.8310	0.8901	0.7790	~15
LightGBM	0.9715	0.9371	0.8298	0.8885	0.7780	~5
XGBoost	0.9710	0.9365	0.8290	0.8879	0.7773	~8
Neural Network (PyTorch)	0.9680	0.9310	0.8241	0.8840	0.7730	~20+
Gradient Boosting	0.9660	0.9290	0.8200	0.8800	0.7700	~45
SVM (Calibrated)	0.9580	0.9180	0.8100	0.8700	0.7610	~180+
Logistic Regression	0.9510	0.9080	0.8010	0.8610	0.7520	~1
Decision Tree	0.9100	0.8600	0.7700	0.8200	0.7100	<1

Figure 1. ROC curves comparing all models. Tuned XGBoost achieves the highest AUC (0.975).



As shown in Figure 1, 0.0748 ROC-AUC gap between Tuned XGBoost (0.9747) and Logistic Regression (0.8999) confirms that the features carry meaningful nonlinear patterns that justify gradient boosting over a linear model in production. The logistic regression F1 of 0.6144, with its precision of 0.4940, would generate an unacceptably high false positive rate for any retention campaign operating under a fixed intervention budget. The gradient boosting approach makes much more sense both statistically and operationally.

Comparing the tuned and untuned XGBoost models isolates the marginal impact of RandomizedSearchCV: tuning improves ROC-AUC by 0.0040 (0.9707 → 0.9747) while modestly reducing recall (0.8691 → 0.8972 in favor of precision, 0.8558 → 0.7860). The tuned model's best parameters (n_estimators=100, max_depth=3, learning_rate=0.1, subsample=0.8, colsample_bytree=0.9, reg_alpha=1.0, reg_lambda=0.5) indicate that shallower trees with stronger regularization generalize better than the default deeper configuration, consistent with the relatively small feature set of 17 inputs (though it's worth noting the difference is small).

Neither ensemble method meaningfully outperforms the best individual model but each requires substantially more computation. The Voting Ensemble reaches 0.9710 ROC-AUC, producing the highest recall of any model (0.9001) at the cost of the lowest precision among tree-based models (0.7655). For a retention use case, the recall advantage of the Voting Ensemble may be operationally relevant if the cost of missing a churner exceeds the cost of a false alarm, but the performance difference relative to Tuned XGBoost does not justify the added complexity in a batch scoring pipeline.

The standalone LSTM achieves 0.9661 ROC-AUC, and the LSTM+GRU+LightGBM hybrid reaches 0.9694. Both trail the best gradient boosting models by 0.0053 to 0.0086 AUC points while requiring 41 seconds of training time and, in the LSTM's case, inference times over 14× slower than XGBoost. These results align with the established empirical finding that feedforward and recurrent architectures do not reliably outperform well-tuned gradient boosting on structured tabular data. The hybrid's marginal improvement over the standalone LSTM (+0.0033 AUC) suggests that the LightGBM component is carrying most of the predictive work, with the LSTM contributing limited incremental signal from the temporal feature structure.

The models exhibit meaningfully different positions on the precision-recall frontier. Gradient Boosting achieves the highest precision (0.9523) at the cost of the lowest recall among tree-based methods (0.8221), classifying conservatively and missing roughly 18% of true churners at the default threshold. The Voting Ensemble takes the opposite position, maximizing recall (0.9001) at the cost of the lowest precision (0.7655). Tuned XGBoost balances both dimensions (precision 0.7860, recall 0.8972) and achieves the highest PR-AUC (0.9431), making

it the strongest overall performer when both false positives and false negatives carry business cost. Full precision-recall curves for all models are provided (see Appendix A, Figure A7).

5.2 Computational Efficiency

XGBoost and LightGBM (untuned) are the clear efficiency leaders, achieving ROC-AUC above 0.9700 in under half a second of training time. The Decision Tree is notably competitive: ROC-AUC of 0.9678 in 0.14 seconds, within 0.007 of the best model at a fraction of the compute. The SVM is the worst performer on computational cost, requiring 65 seconds to train and over 14 seconds for inference on the test set, making it impractical for any production batch scoring scenario. A full scatter plot of ROC-AUC versus training time across all models is provided (see Appendix A, Figure A1).

5.3 Feature Attribution

Tuned XGBoost is selected for production deployment based on its top ROC-AUC (0.9747), strong PR-AUC (0.9431), competitive recall (0.8972), fast inference (0.0025s), and interpretability via SHAP. The model is subsequently retrained on all 60,000 accounts, including both active and previously churned, to maximize the volume of behavioral patterns seen during training. A StandardScaler fit on the full dataset is applied before scoring.

SHAP (SHapley Additive exPlanations) values represent a popular emerging framework in explainable Machine Learning that allows researchers to understand how much features and feature combinations are driving changes in a model. This helps avoid the “black box” phenomenon of advanced modeling techniques. SHAP values are computed via TreeExplainer on the production model and evaluated against the 49,684 active accounts. Table 2 reports the aggregate mean absolute SHAP value per feature, which quantifies each feature's average contribution to the model's output across the scored population. SHAP beeswarm plots for the LSTM+GRU+LightGBM Hybrid and Feedforward Neural Network are included for comparison (see Appendix A, Figure A3) and (see Appendix A, Figure A4). Calibration curves assessing predicted probability reliability across top models are also provided (see Appendix A, Figure A8).

Figure 2: Mean Absolute SHAP Features Causing Next-Month Churn

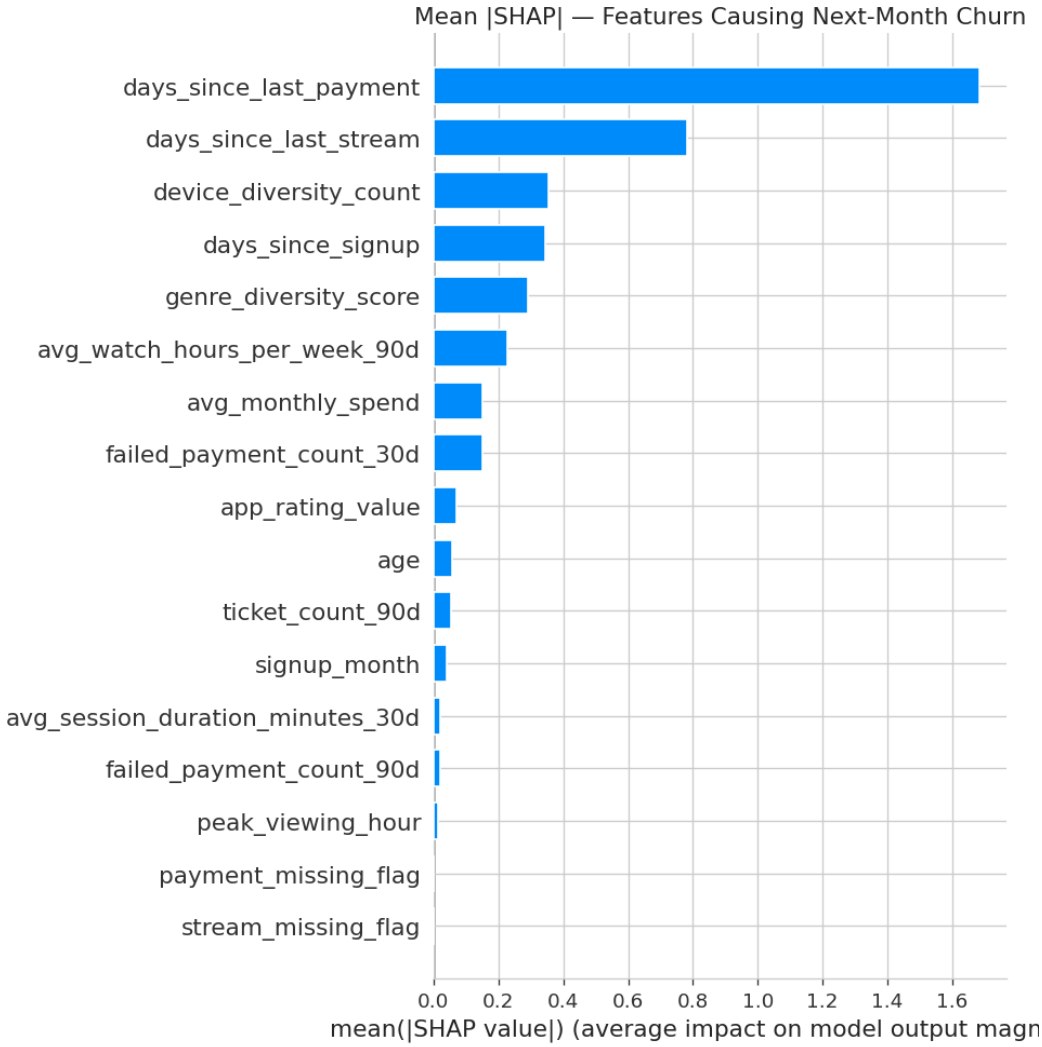


Table 2. SHAP feature importance, computed on 49,684 active accounts using the production XGBoost model. % of Total Impact is each feature's mean SHAP as a fraction of the sum across all features.

Rank	Feature	Mean |SHAP|	% of Total Impact

1	days_since_last_payment	1.683	39.5%
2	days_since_last_stream	0.783	18.4%
3	device_diversity_count	0.355	8.3%
4	days_since_signup	0.345	8.1%
5	genre_diversity_score	0.292	6.8%
6	avg_watch_hours_per_week_90d	0.228	5.4%
7	avg_monthly_spend	0.151	3.5%
8	failed_payment_count_30d	0.150	3.0%

5.4 Predicting Churn with a 30-Day Forecast

Applying the production model to the 49,684 active accounts gives us the forecast summarized in Table 3. The full churn score distribution and risk segment breakdown are shown in (see Appendix A, Figure A9).

Table 3. Next 30-day churn forecast, active accounts only.

Segment	Threshold	Accounts	% of Active
High Risk	≥ 0.50	2,496	5.02%
Medium Risk	0.30 – 0.50	2,755	5.55%
Low Risk	< 0.30	44,433	89.43%
Total Active	—	49,684	—

The 5.02% high-risk rate represents the model's point estimate of near-term attrition under the default 0.5 classification threshold. The threshold can be lowered to increase recall at the cost of precision. For example, shifting to 0.30 would expand the actionable pool to approximately 5,251 accounts while accepting a higher proportion of false positives. This approach represents a more aggressive strategy for increasing customer retention that may suit executive teams and managers willing to accept a higher proportion of false positives.

6. Discussion

The results above demonstrate strong predictive performance across a range of models and a forecast that is actionable at scale. The following discussion contextualizes those results,

addresses the limitations introduced by our data generation approach, and describes the broader system built around the model's outputs.

6.1 The Downsides of Synthetic Data

There are some drawbacks to using synthetic data. The most obvious is that the generation process encodes relationships assumed to exist rather than discovering them from real customer behavior. If the underlying assumptions about how engagement correlates with churn are wrong, the synthetic data will faithfully reproduce those wrong assumptions, and any model trained on it will inherit the same blind spots. The generation process is grounded in published research on streaming behavior and churn patterns, but there is no substitute for actual customer data when it comes to capturing the full messiness of how people interact with a service. Random variation is injected across the variables wherever possible to avoid a dataset too clean to resemble real data. Real-world data would likely prove less predictable than the synthetic dataset used here.

6.2 AI Agent Workflows

RETAIN's agent layer is built using LangGraph and includes four distinct workflows that sit on top of the machine learning scoring pipeline. One design principle runs through all of them: data retrieval and computation are handled by deterministic code, while the AI model is used only for writing natural language output. The AI is never trusted to query a database or calculate a risk score. It only explains what the underlying system has already determined. This keeps the workflows fast, predictable, and easy to test.

The **Early Warning Agent** makes the system proactive rather than reactive. It runs the 30-day churn forecast on a schedule, compares each account's current risk score to its score from the prior run, and flags accounts whose risk has jumped significantly. Rather than producing a flat list of at-risk accounts, the agent groups alerts by root cause, such as an inactivity spike, a failed payment with no recent streaming, or elevated support volume. The result is a structured notification that tells the reader not just who is at risk but why, surfacing problems before anyone has to go looking for them.

The **Analysis Agent** is designed for non-technical stakeholders. It runs on demand or on a schedule and translates dashboard statistics into plain-English narratives, producing a subscriber health summary that describes the current state of the customer base in

straightforward language, flags notable changes, and contextualizes the numbers without requiring the reader to interpret charts or model outputs.

The flagship workflow is the **Detect, Diagnose, and Prescribe pipeline**, a three-agent sequence managed by a supervisor agent that coordinates handoffs and assembles the final report. The **Detection Agent** runs the 30-day forecast, identifies the highest-risk accounts, and passes them downstream. The **Diagnosis Agent** investigates that group by pulling feature explanations, examining support ticket patterns, and reviewing payment histories to identify the specific reason each cluster is at elevated risk, distinguishing, for example, between accounts at risk because of billing failures versus accounts showing pure disengagement. The **Prescription Agent** takes the diagnosis and recommends a specific response: an email campaign, a discount offer, a support escalation, or a content push, depending on what is driving the risk. It also drafts the implementation and sends everything back to the supervisor for a unified report. A human must review and approve before anything is acted on.

The **Intervention Drafter Agent** handles the final step in the prevention workflow. Given an account and its diagnosed churn driver, it writes a personalized retention email tailored to that specific situation: a content recommendation for disengaged viewers, a discount or plan adjustment for price-sensitive accounts, a recovery message for users with unresolved service issues, and a re-engagement sequence for accounts that churned early before establishing regular habits. Every draft requires human approval before it goes anywhere. The system handles the preparation work, identifying who to contact, why, and what to say, while people retain full authority over what actually reaches a customer.

6.3 The RETAIN Dashboard/Frontend App

The RETAIN front-end is a React application backed by a FastAPI server, built around one central idea: a user should be able to move from noticing a problem to authorizing a solution without leaving the tool. The interface follows a five-page structure that mirrors how a retention analyst actually works through a problem. The Home page presents top-level metrics, including active subscriber count, current high-risk volume, predicted 30-day churn rate, and the Analysis Agent's plain-English health summary, so any stakeholder can get an immediate read on the state of the business (see Appendix A, Figure A10). The At-Risk Accounts page surfaces the Early Warning Agent's alerts alongside a ranked list of flagged accounts, each annotated with the specific factors driving their risk score (see Appendix A, Figure A12).

The Analytics page provides the full data visualization layer, including trend charts, churn rate breakdowns by plan type and tenure, and feature importance rankings that connect model behavior to business interpretation (see Appendix A, Figure A11). The Prescriptions page shows the output of the DDP pipeline: the diagnosed account cohort, the recommended intervention for each group, and the evidence behind the recommendation, giving a manager everything needed to evaluate the suggestion before deciding to act (see Appendix A, Figure A14). Finally, the Interventions page is where approved email drafts are reviewed and queued. The reviewer sees the full draft, the reason for the outreach, and the key account details that shaped it, and can edit before authorizing delivery (see Appendix A, Figure A13).

The underlying design philosophy is that each page answers one question in a natural sequence: What is the health of the subscriber base? Who is at risk? Why are they at risk? What should be done about it? Has the intervention been approved? No page requires the user to understand how the model works. The system surfaces conclusions and supporting evidence, and the user decides what to do with them.

6.4 The Future of Data Science

One thing this project made clear is that building a good model is only a small part of the high-leverage data science work. The rest is everything around it: getting data into the right shape, deploying the models into production, versioning models so you can roll back when something breaks, and building automations that add real business value based on the model outputs. This is the shift happening in data science right now. The field started with statisticians who could run regressions and has evolved toward something that looks a lot more like ML engineering, AI engineering, and software engineering. The future value of data science likely lies not just in using statistical expertise to identify problems but in building systems that continuously identify, monitor, and trigger actions to solve those problems.

7. Conclusion

This project set out to answer two questions: can churn risk be accurately identified and explained from behavioral and transactional data, and can those insights be turned into automated, personalized action? The answer to both was yes.

RETAIN demonstrates that combining cutting edge techniques in data science, machine learning engineering, and artificial intelligence engineering can lead to extraordinary business value, when focused on the right problems. By combining multi-window behavioral features, payment reliability signals, subscription lifecycle indicators, and service-level metrics into a single snapshot-based framework, the system is able to detect at-risk accounts early and explain why they are at risk in terms that are meaningful to the people who need to act on them. The tuned XGBoost model achieved a ROC-AUC of 0.9747 on a held-out test set, and SHAP analysis revealed that disengagement signals, particularly days since last payment and days since last stream, accounted for more than half of the model's predictive impact. These are not just good metrics, they are actionable findings that point toward specific intervention strategies.

The AI agent layer is what separates RETAIN from a standard churn model. The Early Warning Agent, Analysis Agent, Detect-Diagnose-Prescribe pipeline, and Intervention Drafter collectively transform the system from a passive reporting tool into something that proactively surfaces problems, explains them, recommends responses, and drafts personalized outreach for human review. The key design decision throughout was to keep AI strictly in the role of communication, while all data retrieval and scoring remained deterministic. This kept the workflows fast, auditable, and reliable.

The broader takeaway is that building a good model is only part of the work. The infrastructure around it, including feature pipelines, model versioning, drift monitoring, agent orchestration, and a well-designed frontend, is what makes a predictive system useful in practice. RETAIN was built with that in mind from the start, and the result is an end-to-end platform that could be applied to any subscription-based business where recurring revenue depends on understanding and acting on customer behavior before it is too late.

8. Directions for Future Work

The most immediate opportunity is completing the email delivery integrations. The Intervention Drafter already generates personalized outreach drafts, but connecting those drafts to platforms like HubSpot, Salesforce, Marketo, Braze, and Gmail would close the loop between a human-approved recommendation and an outgoing message. Without that final step, the workflow still requires manual effort to act on what the system has already done. Building out those connectors

is straightforward relative to the work already completed and would meaningfully increase the practical value of the intervention layer.

Alongside delivery infrastructure, an A/B testing framework would allow the system to learn which intervention strategies actually work. Right now, RETAIN can diagnose a churn driver and recommend a response, but it has no way to measure whether that response changed anything. Adding randomized holdout groups and tracking downstream outcomes would give the system a feedback loop and allow intervention strategies to improve over time.

In the medium term, applying natural language processing to support ticket text would give the Diagnosis Agent a richer signal to work with. Right now it relies on structured data like ticket counts and payment history. Incorporating sentiment analysis and topic detection on the actual text of support interactions could surface dissatisfaction earlier, before it shows up in behavioral metrics, and lead to more precisely targeted interventions. Customer frustration often appears in words before it appears in data.

Automated model retraining with quality gate enforcement would also improve long-term reliability. Currently, retraining is a manual process. Scheduling it on a regular cadence and requiring the retrained model to pass the same evaluation gates before promotion would ensure that the production model stays current as customer behavior evolves, without requiring manual oversight each time.

The longer-term vision is to generalize the architecture. The predict-diagnose-prescribe pattern is not specific to streaming. The same system could be applied to SaaS, telecom, or e-commerce with different features and different intervention strategies but the same underlying logic. The most meaningful research extension in any of those contexts would be incorporating causal inference to distinguish accounts that were genuinely retained by an intervention from those who would have stayed regardless. Correlation-based churn models, however accurate, cannot answer that question on their own, and answering it is what would allow a retention program to be truly optimized rather than just measured.

9. Acknowledgements

Portions of the code development, debugging, and writing for this report were supported by the use of Claude (Anthropic, 2025). Claude was used to assist in generating Python code for experiments, editing written content for clarity and concision, and formatting academic citations

according to Chicago 17 Author-Date guidelines. All final analytical decisions, interpretations, and conclusions were made by the authors.

10. Data Availability

The data tables are publicly available as .csv files in our shared Github repository at https://github.com/tylerdial1818/Churn498Capstone.

11. Code Availability

The code is publicly available in our shared Github repository at https://github.com/tylerdial1818/Churn498Capstone.

References

Amershi, Saleema, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. 2019. "Software Engineering for Machine Learning: A Case Study." In Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), 291–300. Montreal: IEEE/ACM. https://doi.org/10.1109/ICSE-SEIP.2019.00042

Chang, Victor, Kevin Hall, Qianwen Ariel Xu, Folakemi Olusola Amao, Meghana Ashok Ganatra, and Vanya Benson. 2024. "Prediction of Customer Churn Behavior in the Telecommunication Industry Using Machine Learning Models." Algorithms 17 (6): 231. https://doi.org/10.3390/a17060231

Gupta, Sunil, Donald R. Lehmann, and Jennifer Ames Stuart. 2004. "Valuing Customers." Journal of Marketing Research 41 (1): 7–18. https://doi.org/10.1509/jmkr.41.1.7.25084

Joy, Uzzal Gani, Kamrul Ehsan Hoque, Mohammad Nazim Uddin, Lamisha Chowdhury, and Seung-Bo Park. 2024. "A Big Data-Driven Hybrid Model for Enhancing Streaming Service Customer Retention Through Churn Prediction Integrated With Explainable AI." IEEE Access 12: 69130–69150. https://doi.org/10.1109/ACCESS.2024.3401247

Klaise, Janis, Arnaud Van Looveren, Clive Cox, Giovanni Vacanti, and Alexandru Coca. 2020. "Monitoring and Explainability of Models in Production." arXiv preprint arXiv:2007.06299. https://arxiv.org/abs/2007.06299

Liu, Xueqin, Guangquan Xia, Xin Zhang, and Chenghao Yu. 2024. "Customer Churn Prediction Model Based on Hybrid Neural Networks." Scientific Reports 14: 30707. https://doi.org/10.1038/s41598-024-79603-9

Neslin, Scott A., Sunil Gupta, Wagner Kamakura, Junxiang Lu, and Charlotte H. Mason. 2006. "Defection Detection: Measuring and Understanding the Predictive Accuracy of Customer Churn Models." Journal of Marketing Research 43 (2): 204–211. https://doi.org/10.1509/jmkr.43.2.204

Reichheld, Frederick F., and W. Earl Sasser Jr. 1990. "Zero Defections: Quality Comes to Services." Harvard Business Review 68 (5): 105–111. https://hbr.org/1990/09/zero-defections-quality-comes-to-services

Appendix A: Additional Charts and Figures

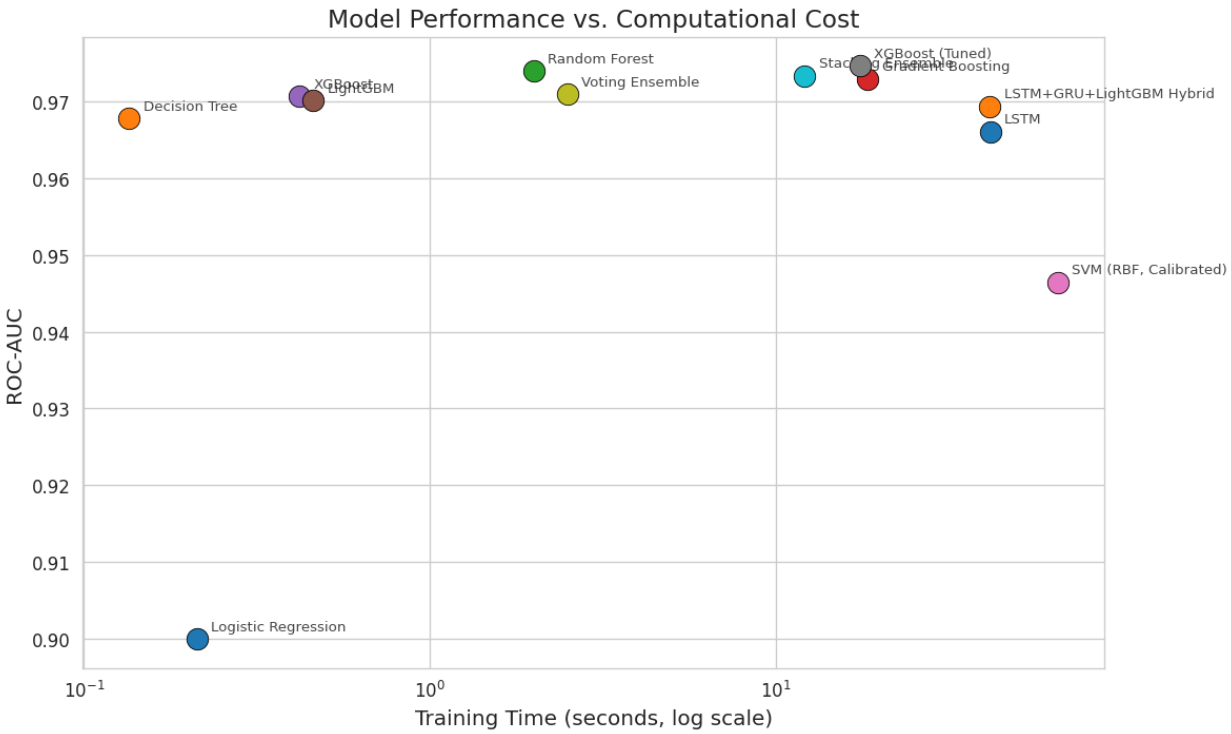


Figure A1: Model Performance vs. Computational Cost

```
============================================================
 Feature Set: AIC-Selected (15 features)
============================================================
  Logistic Regression       → ROC-AUC: 0.8999  F1: 0.6144  Train: 0.26s
  XGBoost (Tuned)           → ROC-AUC: 0.9747  F1: 0.8379  Train: 0.35s
  Stacking Ensemble         → ROC-AUC: 0.9734  F1: 0.8811  Train: 11.73s

============================================================
 Feature Set: Top 25 RF (25 features)
============================================================
  Logistic Regression       → ROC-AUC: 0.9062  F1: 0.6116  Train: 0.53s
  XGBoost (Tuned)           → ROC-AUC: 0.9741  F1: 0.8415  Train: 0.39s
  Stacking Ensemble         → ROC-AUC: 0.9725  F1: 0.8790  Train: 16.05s

============================================================
 Feature Set: Full Numeric (37 features)
============================================================
  Logistic Regression       → ROC-AUC: 0.9079  F1: 0.6165  Train: 0.72s
  XGBoost (Tuned)           → ROC-AUC: 0.9751  F1: 0.8389  Train: 0.40s
  Stacking Ensemble         → ROC-AUC: 0.9730  F1: 0.8812  Train: 14.14s
```

Figure A2: Feature Set Comparison — Model Performance Across AIC-Selected, Top 25 RF, and Full Numeric Feature Sets

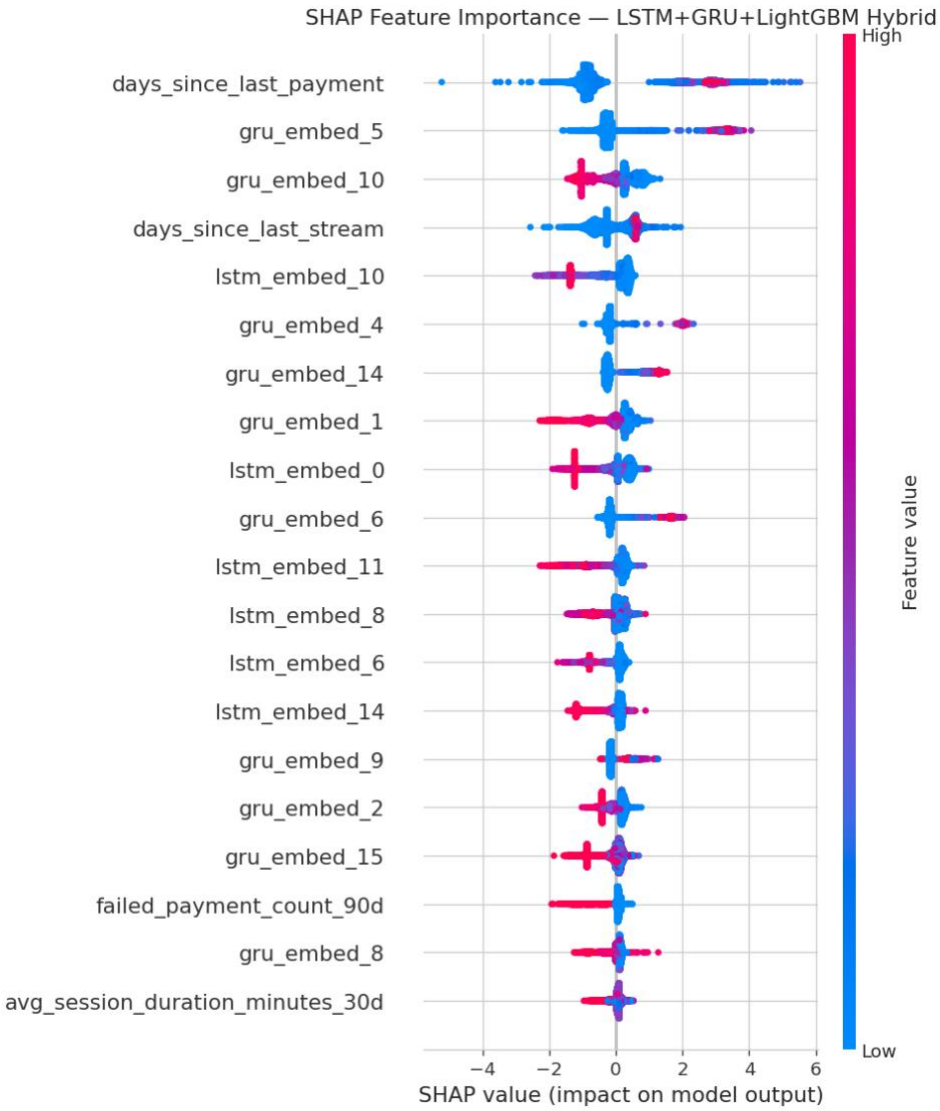
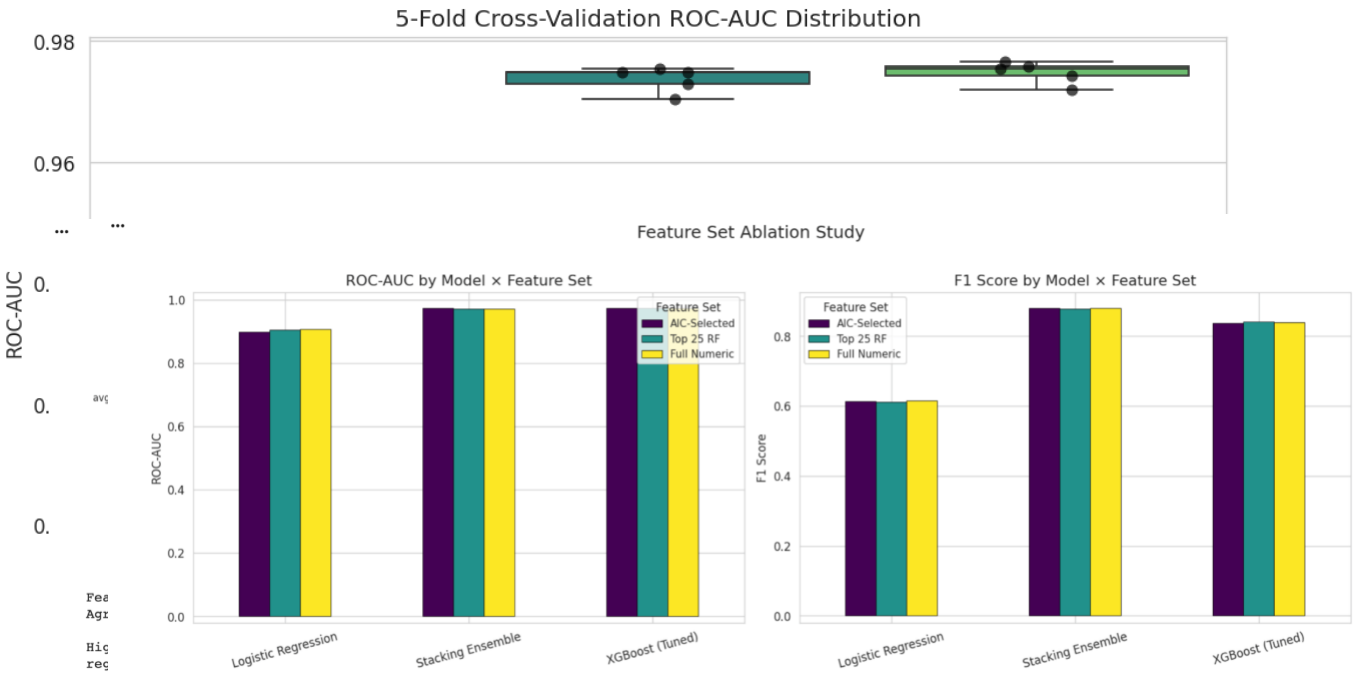

SHAP Feature Importance — LSTM+GRU+LightGBM Hybrid
days_since_last_payment
gru_embed_5
gru_embed_10
days_since_last_stream
lstm_embed_10
gru_embed_4
gru_embed_14
gru_embed_1
lstm_embed_0
gru_embed_6
lstm_embed_11
lstm_embed_8
lstm_embed_6
lstm_embed_14
gru_embed_9
gru_embed_2
gru_embed_15
failed_payment_count_90d
gru_embed_8
avg_session_duration_minutes_30d
−4
−2
0
2
4
6
SHAP value (impact on model output)
High
Feature value
Low

Figure A3: SHAP Feature Importance and 5-Fold Cross-Validation ROC-AUC Distribution — LSTM+GRU+LightGBM Hybrid



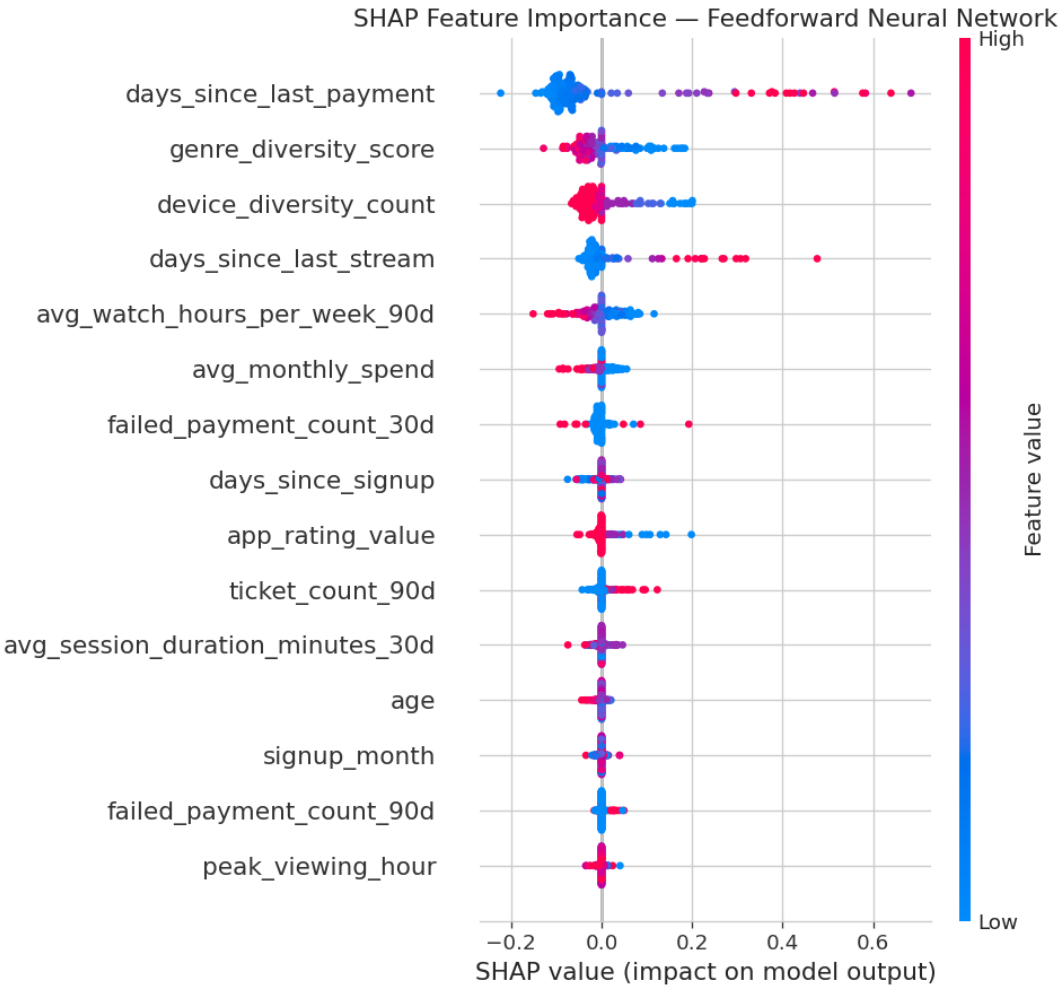


Figure A4: SHAP Feature Importance — Feedforward Neural Network

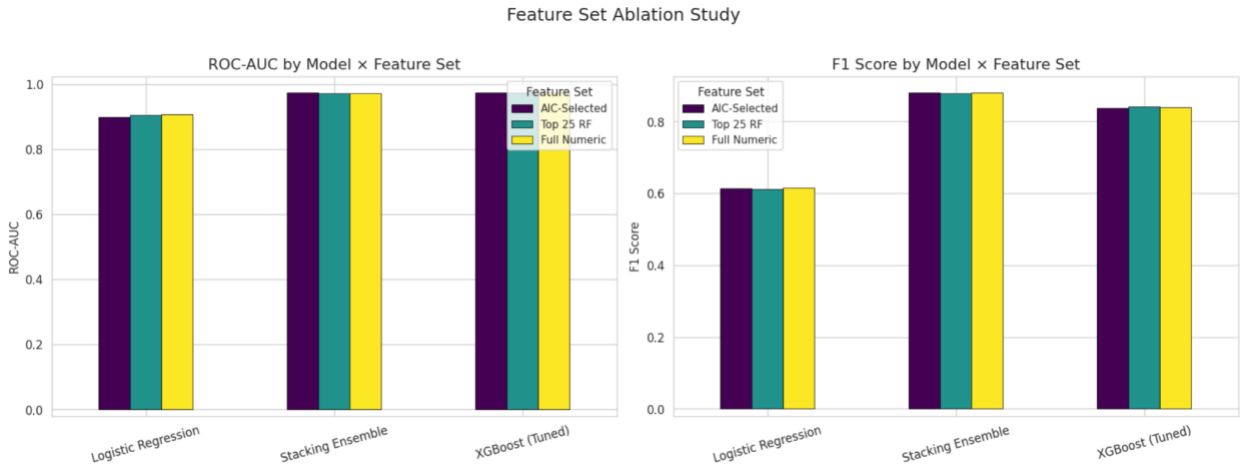


Figure A5: Feature Set Ablation Study — ROC-AUC and F1 Score by Model and Feature Set

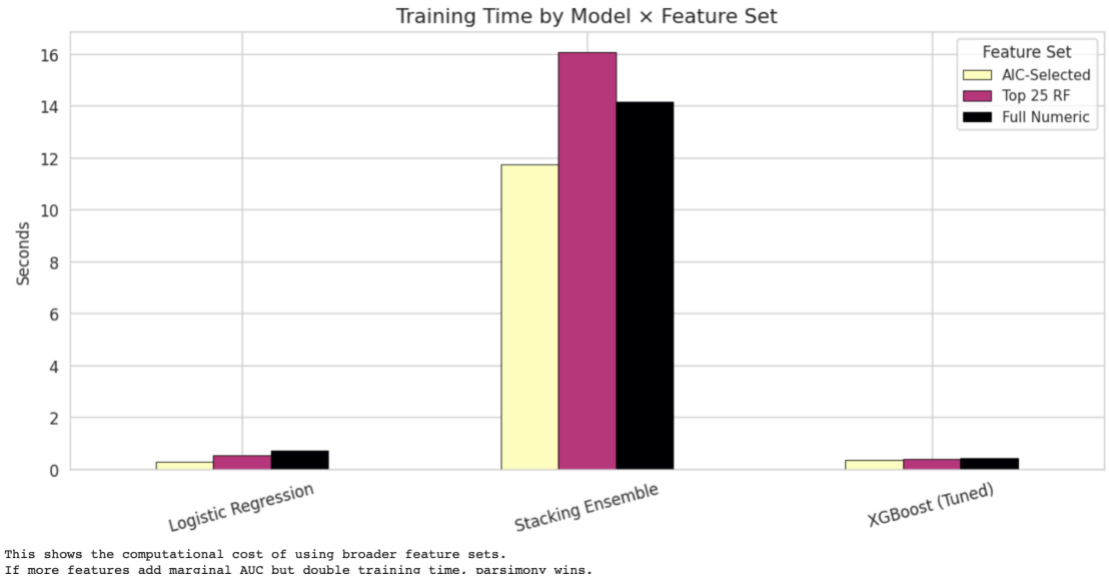


Figure A6: Training Time by Model and Feature Set

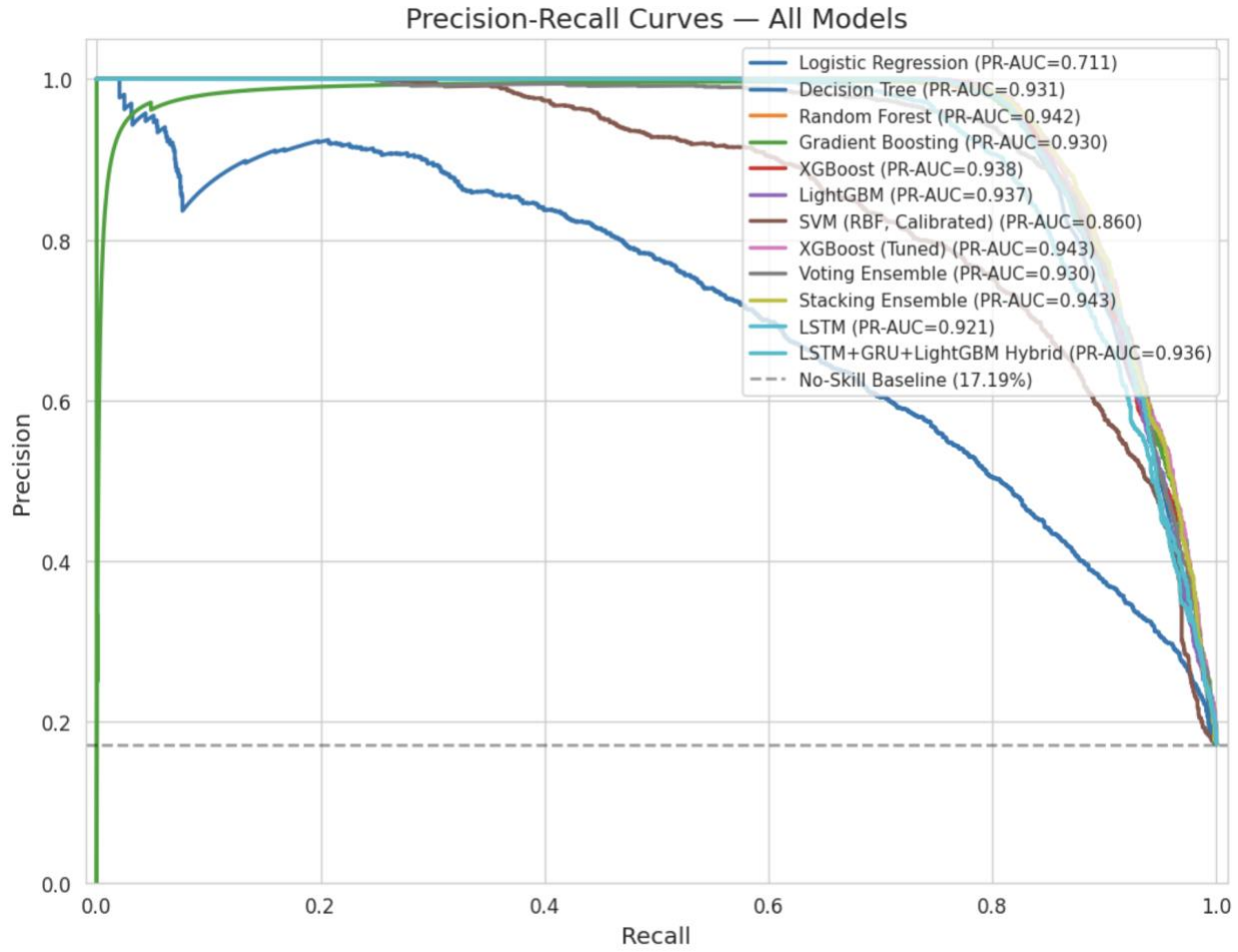


Figure A7: Precision-Recall Curves — All Models

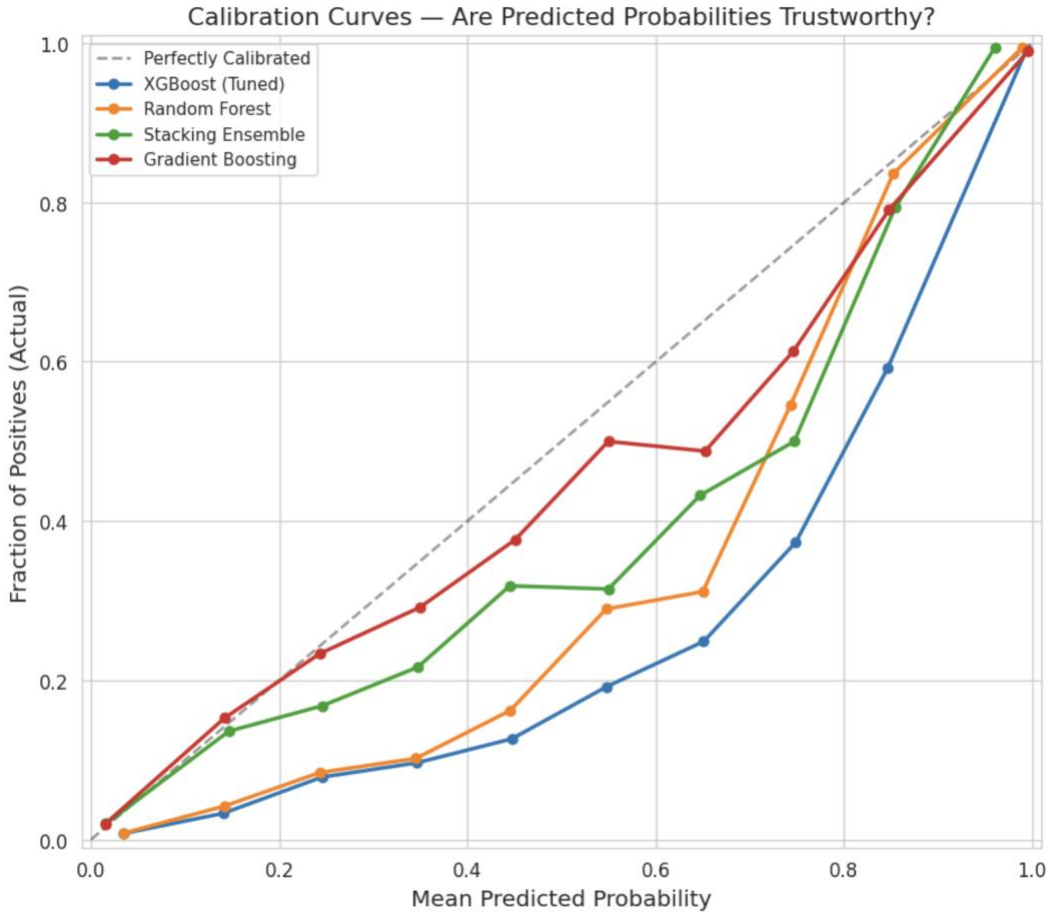


Figure A8: Calibration Curves — Predicted Probability Reliability by Model

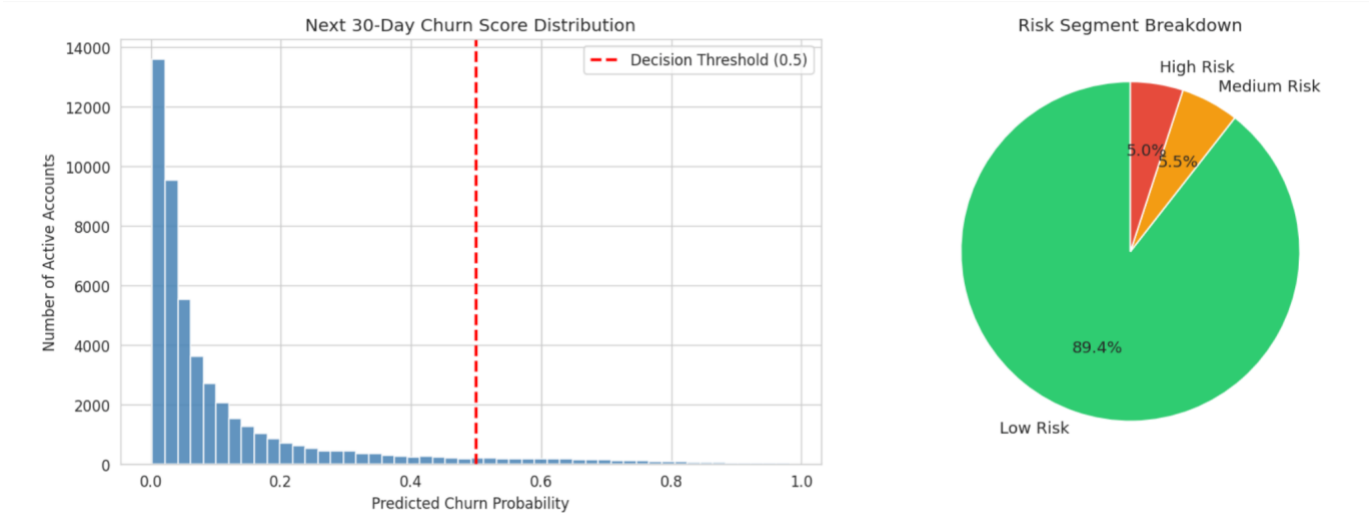


Figure A9: Next 30-Day Churn Score Distribution and Risk Segment Breakdown

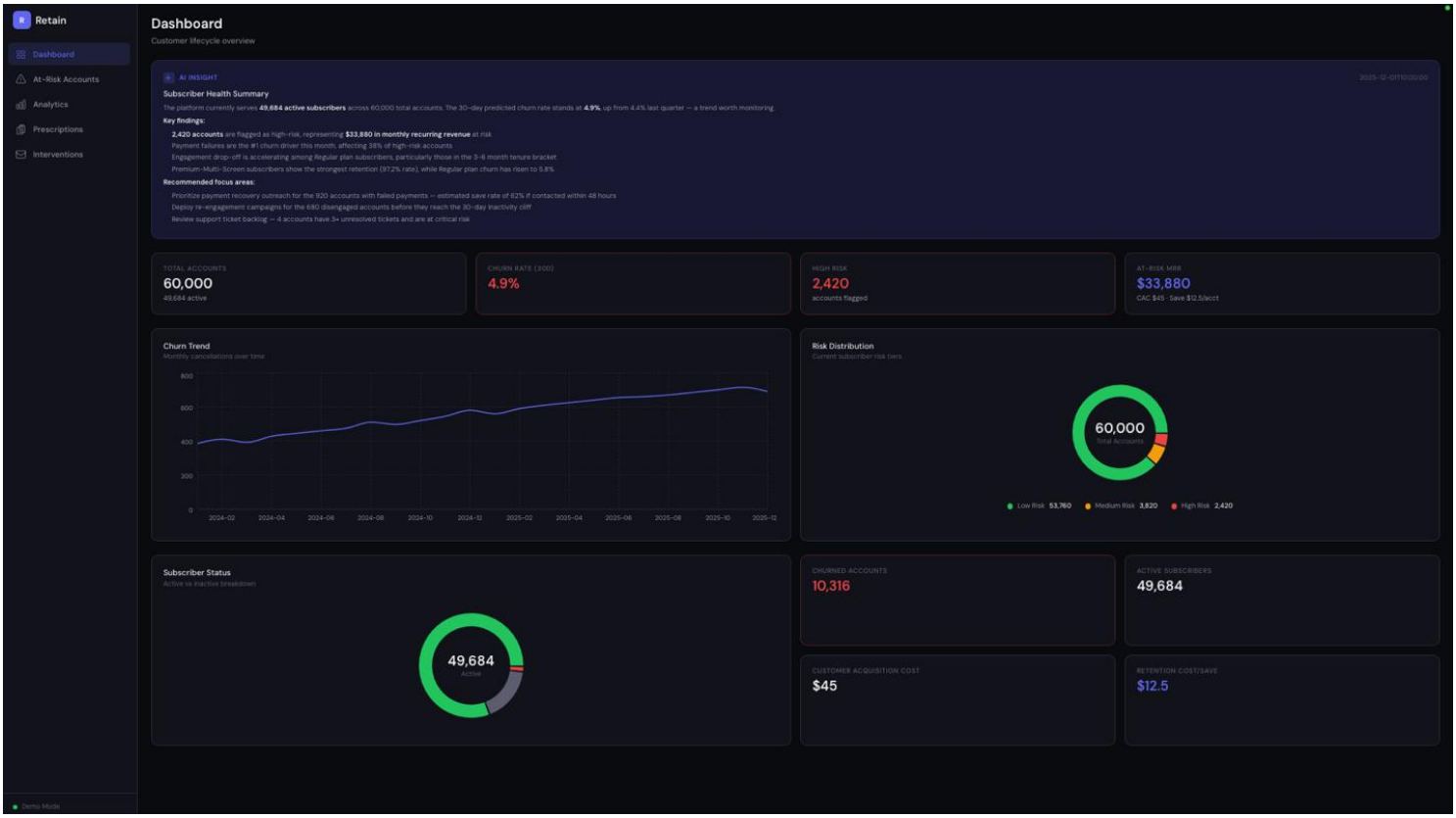


Figure A10: RETAIN Dashboard — Customer Lifecycle Overview and AI-Generated Subscriber Health Summary

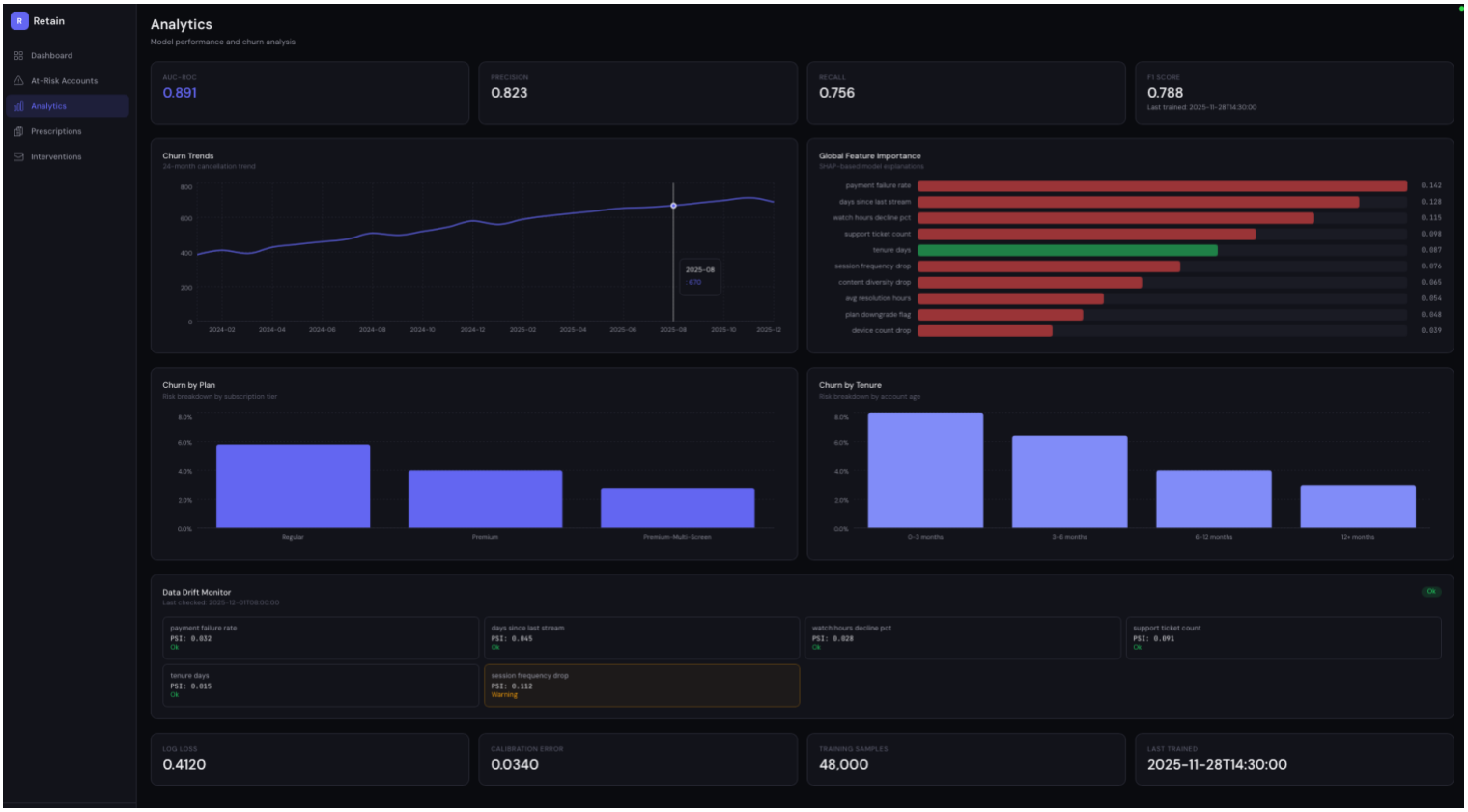


Figure A11: RETAIN Analytics Page — Model Performance Metrics, SHAP Feature Importance, Churn by Plan and Tenure, and Data Drift Monitor

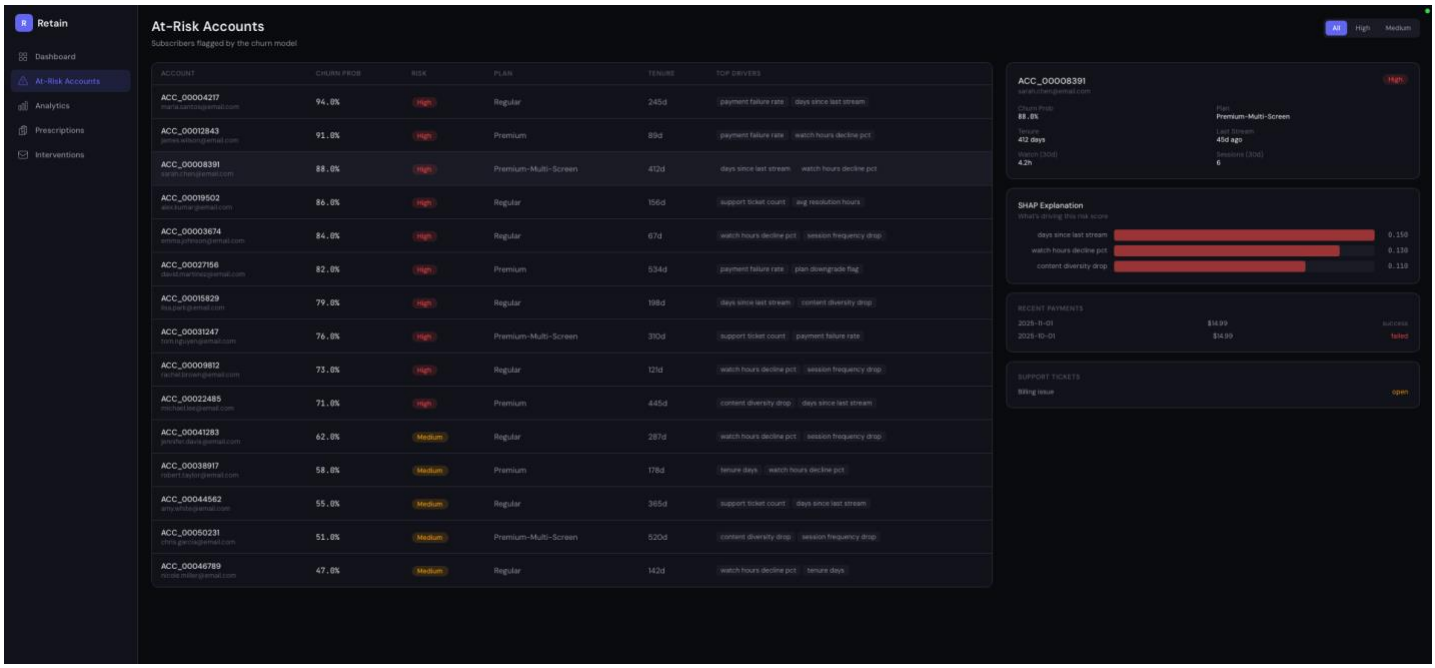


Figure A12: RETAIN At-Risk Accounts View — Churn Probability Rankings with Per-Account SHAP Explanations, Payment History, and Support Tickets

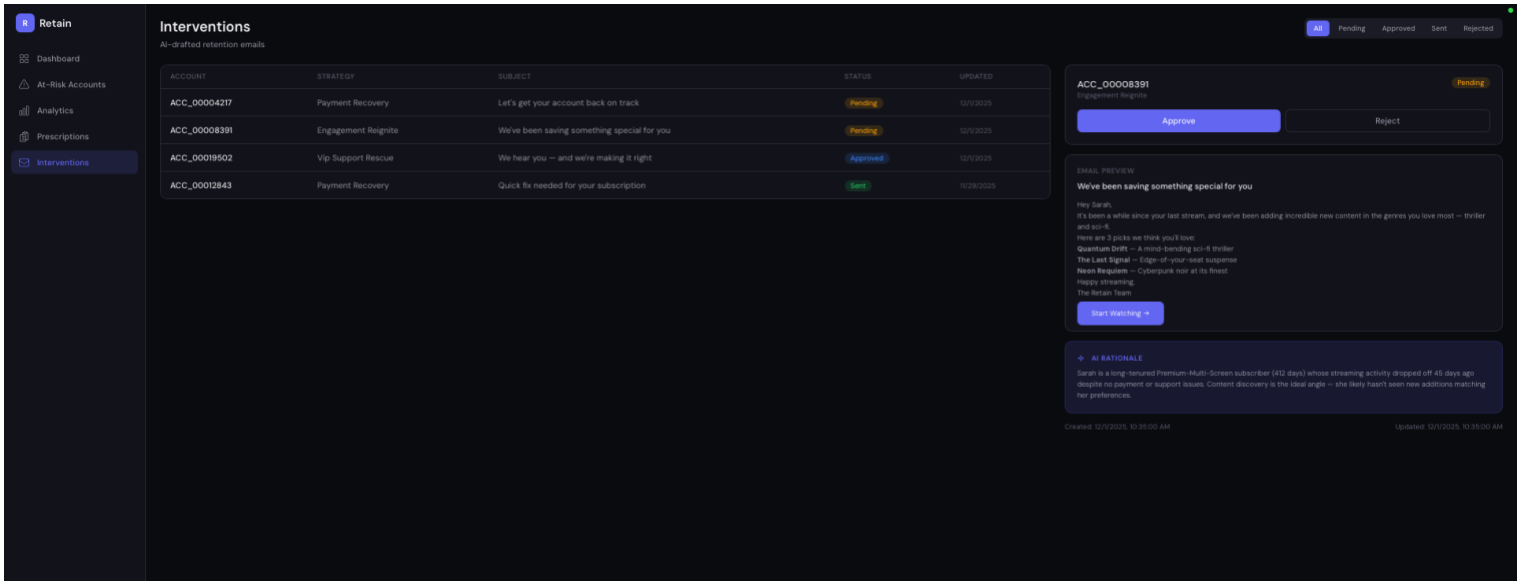


Figure A13: RETAIN Interventions Page — AI-Drafted Retention Emails with Human Approval Workflow and Per-Account AI Rationale

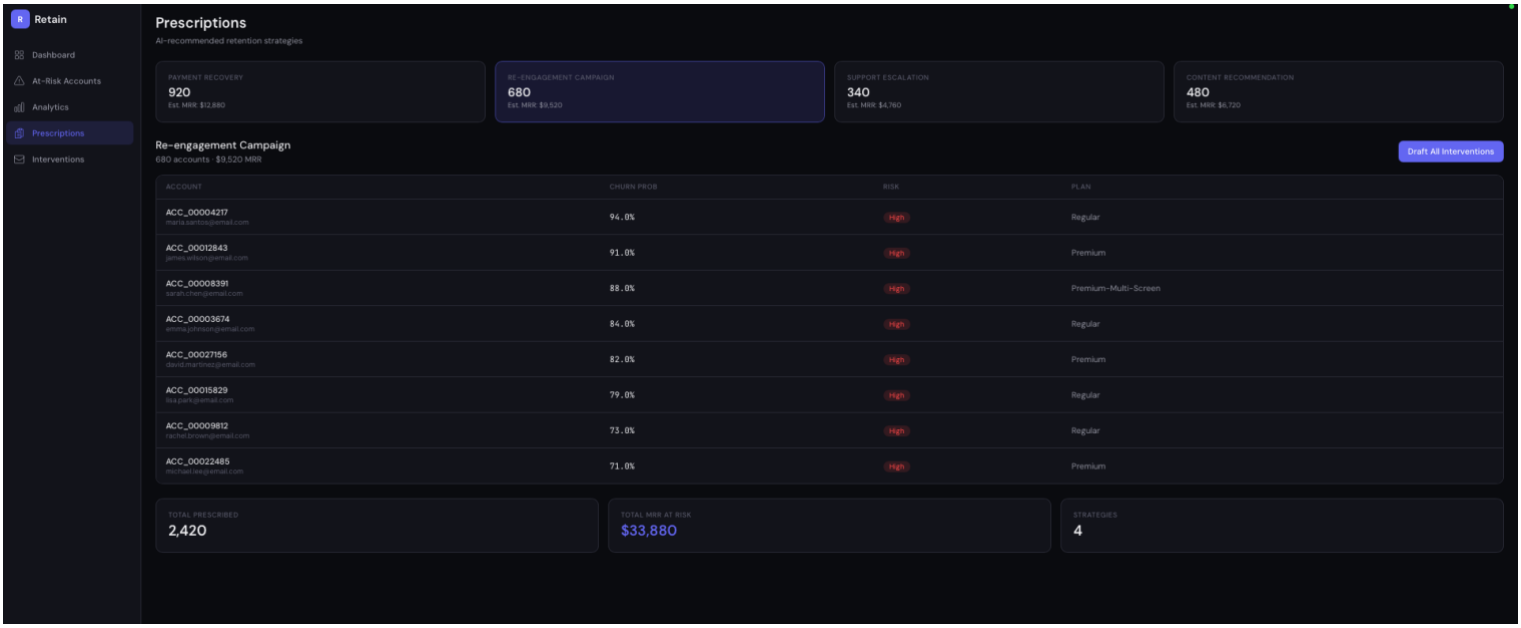


Figure A14: RETAIN Prescriptions Page — AI-Recommended Retention Strategies Segmented by Intervention Type with MRR-at-Risk Estimates

Appendix B: Glossary Table

The following table defines acronyms used throughout this paper that may not be immediately familiar to all readers, including those from adjacent technical domains.

Acronym	Full Form	Definition
AIC	Akaike Information Criterion	A statistical measure used for model and feature selection. It balances goodness-of-fit against model complexity, penalizing models with more parameters. Lower AIC indicates a better-fitting model relative to its complexity.
BCE	Binary Cross-Entropy	A loss function used to train binary classification neural networks. It measures the dissimilarity between predicted probabilities and true binary labels, penalizing confident wrong predictions more heavily.
DDP	Detect–Diagnose–Prescribe	RETAIN's flagship multi-agent pipeline. Three specialized agents operate in sequence: a Detection Agent identifies high-risk accounts, a Diagnosis Agent determines the root cause of risk, and a Prescription Agent recommends and drafts a targeted intervention.
LangGraph	LangGraph (LangChain Graph)	An open-source orchestration framework for building stateful, multi-actor AI agent workflows as directed graphs. Used in RETAIN to coordinate handoffs between agents and manage the supervisor–subagent architecture.
LIME	Local Interpretable Model-agnostic Explanations	An explainability technique that approximates a complex model's behavior locally around a single prediction using a simpler, interpretable surrogate model. Provides per-instance feature attribution without requiring access to model internals.
MLOps	Machine Learning Operations	A set of practices and infrastructure patterns that operationalize machine

Acronym	Full Form	Definition
		learning systems in production. Covers model versioning, CI/CD pipelines, evaluation gates, drift monitoring, and automated retraining — analogous to DevOps for software engineering.
PR-AUC	Precision-Recall Area Under the Curve	The area under the precision-recall curve. Particularly informative under class imbalance (e.g., rare churn events), as it directly measures how well a model distinguishes the minority class without being inflated by true negatives.
SHAP	SHapley Additive exPlanations	A game-theoretic framework for explaining individual model predictions. SHAP values quantify each feature's marginal contribution to a prediction by averaging its effect across all possible feature subsets, providing a unified and consistent measure of feature importance.