Retain the Legends:

Using Game Data to Predict Player Churn

By: Victoria Brigola

The Fight to Keep Players in the Game

As both a lifelong gamer and former game artist, I've always been fascinated by the invisible line between "hooked" and "drifted away." *Apex Legends* feeds that curiosity: its ever-shifting cast of legends, maps, and balance patches constantly rekindles excitement, yet churn stubbornly persists. Rather than drown in years of patch noise, I zeroed in on Season 15 ("Eclipse"). That single-season lens delivers a stable meta, letting me attribute engagement shifts to player behavior not to wholesale system overhauls.

For this study I tapped the public **Season 15 Ranked Dataset**, a snapshot of *almost 500* matches recorded during the split (499 after cleaning) and **35 match-level variables** spanning performance (kills, damage), play style (legend picks, squad composition), and cadence (session frequency, days since last match). Concentrating on my own full-squad logs let me engineer granular features like `session_frequency` and `legend_diversity`, the very signals live-ops teams care about.

This capstone fuses my creative roots with data-science rigor to tackle a question that matters to both players and Electronic Arts: **Can we predict which Season 15 competitors are on the verge of churning and intervene before disengagement sets in?** A reliable early-warning system would empower Respawn's game-design leads, live-ops product managers, and monetization teams to keep squads dropping into the Outlands, season after season.

What's at Stake: Problem Statement

I've spent years both **playing** and **building** games, so I'm keenly aware of the moment a regular slips away. In *Apex Legends* that moment can arrive fast, every new legend buff, map rotation, or balance tweak reshapes the meta. To study churn without cross-season noise, I focused on Season 15 ("Eclipse") which gives me a clean lens: a single new map, modest legend tweaks, and otherwise stable mechanics let me tie churn to player behavior rather than sweeping balance shifts. My cleaned dataset: 499 of my own Season 15 matches with 35 curated features, captures combat performance (kills, damage, placement), tactical preferences (legend choice), and behavioral cadence (days since last match, session frequency, legend diversity). I mark a player as churned if they record no ranked match for 7 consecutive days, giving a balanced mix of churners and retained players for supervised modeling.

★ **Business Stakes**

- ➢ Matchmaking integrity: full lobbies keep wait times low and skill tiers meaningful.
- ➢ Revenue health: weekly activities advance battle-passes and purchase cosmetics; churners do not.
- ➢ Community vitality: engaged players stream, share highlights, and invite friends.

★ **Core Question:** Can I predict which Season 15 players will churn, defined as 7 days of inactivity using only their match-level performance, legend choices, and play-cadence signals?

★ **Analytic Objective:** I train and compare Logistic Regression, Random Forest, and XGBoost models to classify churners one week in advance, evaluating each with accuracy, precision, recall, F1, and ROC-AUC.

★ **Primary Stakeholders:**
- ➢ Respawn game-design leads
- ➢ Live-ops product managers
- ➢ Marketing strategists
- ➢ Monetization teams

My goal is to turn raw Season 15 match logs into a data-driven early-warning system that lets these teams intervene before a competitor leaves the Outlands and feature seasons behind for good.

Looting the Logs: Data Wrangling

I began with the public **Apex Legends Season 15 Ranked Dataset**, which captures every ranked match played between **Nov 2, 2022 and Jan 14, 2023** from Kaggle.. Each raw record includes a timestamp, unique player ID, map, legend choice, placement, kills, assists, damage, and session length which are exactly the fields needed to study engagement over time. After loading the full log, I removed extraneous streamer-specific telemetry, dropped a handful of duplicate rows, filled the rare missing match_duration values with a map-level median, and renamed columns for clarity.

To convert match logs into retention signals, I engineered several cadence-focused features and one-hot encoded key categorical field. The resulting tidy frame became the modeling foundation.

★ **Core predictors fed to the models:**

- ➢ days_since_last_match (recency of play)
- ➢ session_frequency (matches played per rolling 7-day window)
- ➢ legend_diversity (count of unique legends used over the past week)
- ➢ match_duration (seconds alive per game)
- ➢ avg_kills (mean kills across a three-match rolling window)
- ➢ avg_damage (mean damage across the same window)
- ➢ placement (final squad rank in each match)
- ➢ One-hot vectors for map and legend selections

Finally, I exported the fully cleaned, 7 day-churn-flagged dataset to ApexPlayerRetention_DataWrangling.csv, ready for exploratory analysis and supervised modeling.

Behind the Stats: Exploratory Data Analysis

I kicked off the EDA by sanity-checking class balance, confirming that retained players outnumber churners roughly 3 : 1. From there I profiled every numeric column: kills, damage, revives, placement, and match duration to spot skew and outliers, then split each distribution by retention status to see where the cohorts actually diverge. Next, I generated a full correlation matrix to surface multicollinearity and locate high-signal pairs worth engineering (for example, damage and duration travel together more tightly than kills and duration, hinting at different play styles). Finally, I combined cadence features such as `days_since_last_match` with performance metrics in two-axis scatter maps to visualize how session recency modulates combat output.

With the univariate maps mapped, I drilled into a few bivariate plots that matter most to stakeholders: boxplots contrasting retained vs. churned players on core KPIs, heatmaps showing where features cluster, and time-series lines revealing when overall activity spikes or dips. Each figure is annotated with short "Key Findings" captions so non-technical readers can grasp the takeaway without squinting at axes.

- ★ **Retention Class Distribution (Figure 1)** — shows the exact retained vs. churned split, establishing the baseline probability any model must beat.
- ★ **Correlation Heatmap (Figure 2)** — highlights strong positive links such as damage to duration and flags near-zero correlations that can be safely dropped or binned.
- ★ **Match Duration by Retention (Figure 3)** — illustrates that longer survival correlates with higher retention, suggesting session length as a lever for design tweaks.
- ★ **Player Behavior Map: Damage vs. Duration (Figure 4)** — plots individual matches to reveal distinct play-style clusters (high-damage sprinters vs. long-survival support roles) and how each cluster aligns with churn risk.

These visuals frame the narrative for stakeholders, making it clear where engagement succeeds, where it falters, and which behaviors the subsequent models will treat as leading indicators of churn.

Figure 1

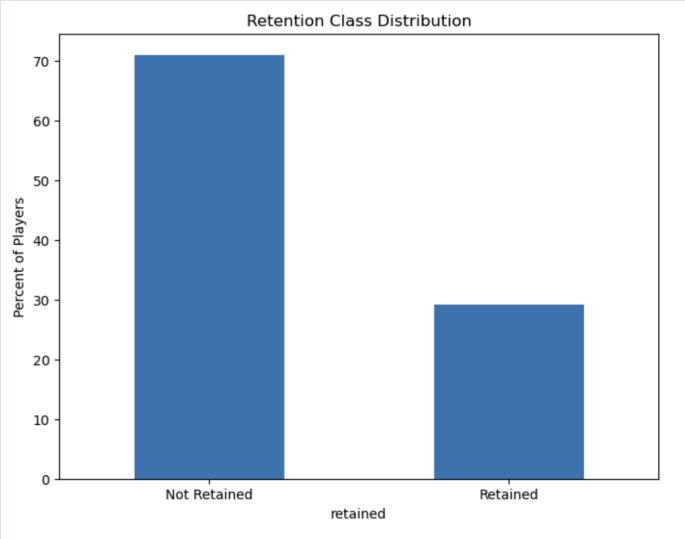
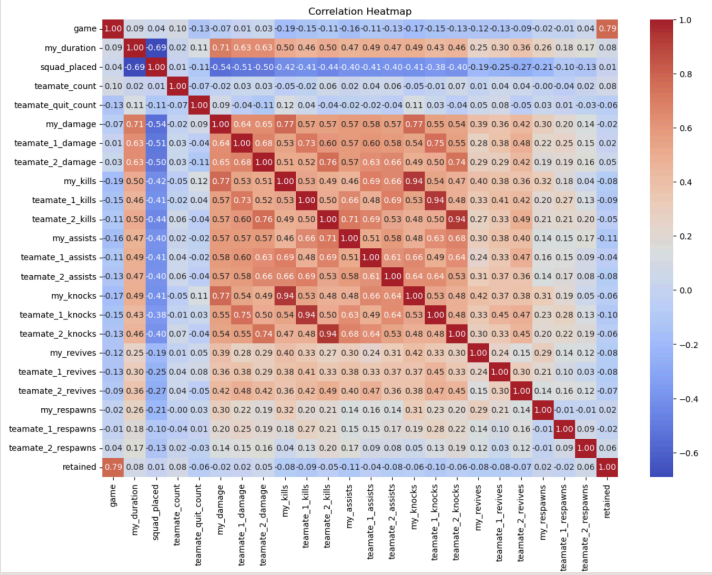


Figure 2



. Figure 3

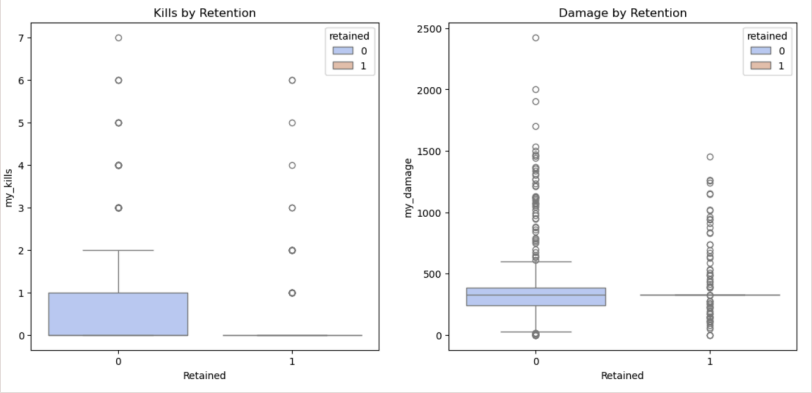
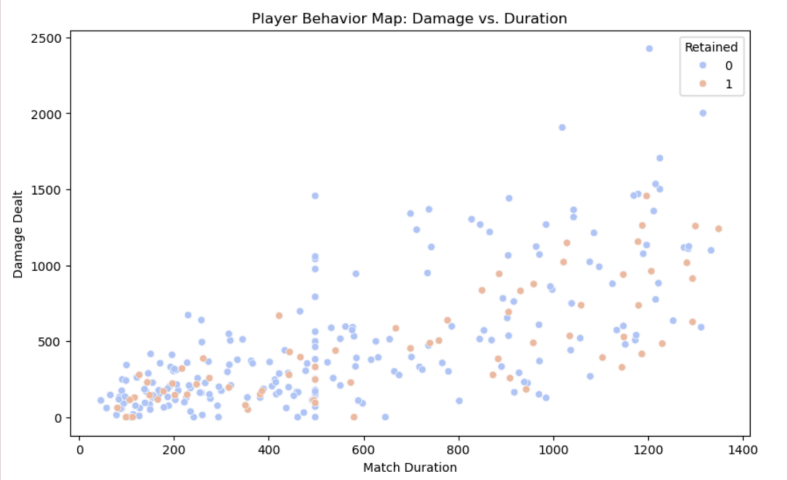


Figure 4



Fine-Tuning the Arsenal: Preprocessing

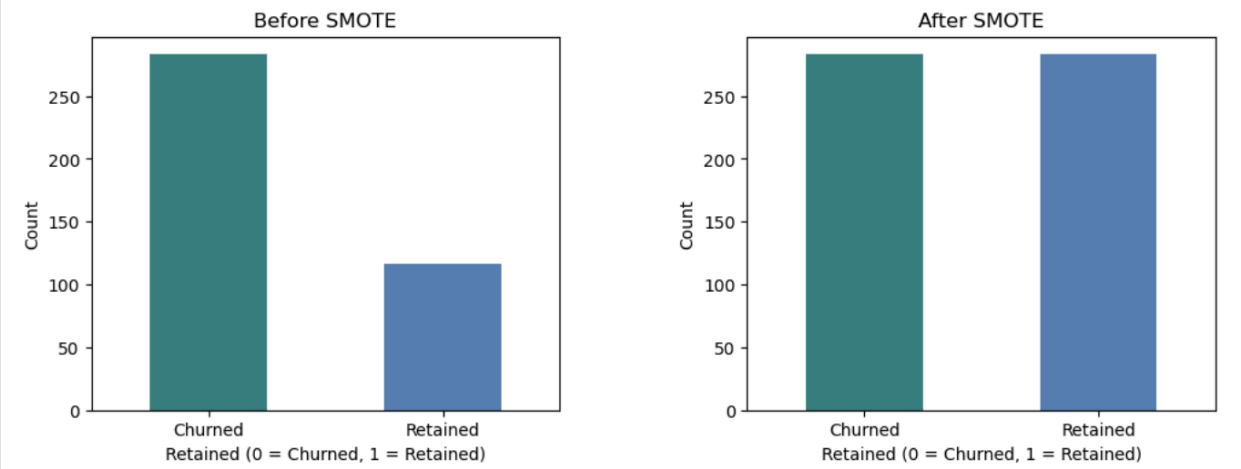
Before I could feed the data into any classifier, I needed to convert raw match logs into a numeric matrix the models could digest. I began by separating predictors (X) from the churn flag (y), then inspected every column's data type to decide between scaling and encoding. Continuous variables such as match duration and cumulative damage required normalization to ensure fair weightings, while categorical choices like map rotations and legend picks needed one-hot vector so the algorithms wouldn't impose an arbitrary rank order.

Class imbalance posed the next hurdle: roughly three retained players appeared for every churner. To give the minority class an equal voice during training, I applied **SMOTE** oversampling, boosting churn examples until they represented just under half the training set. Finally, I performed a stratified 80 / 20 split so both the training and test partitions preserved the original churn ratio, guaranteeing an honest evaluation later on. Once these steps were complete, I cached the scaler and encoder objects for reproducibility and saved the processed frames to disk for the modeling notebook.

- ★ **One-hot encoding** map (Broken Moon, Olympus, World's Edge) and legend selections
- ★ **Standardization** `match_duration`, `avg_damage`, `avg_kills`, and `days_since_last_match` via StandardScaler
- ★ **Feature matrix shape** after encoding and scaling: 499 rows × *n* columns , where *n = 35 original features plus every one-hot dummy created for maps and legends
- ★ **Class rebalancing** SMOTE raised churn representation from 29 % to 45 % in the training set (Figure 5)
- ★ **Train/test split** stratified 80 / 20 partition, random seed fixed for reproducibility

With a clean, balanced, and fully numeric dataset in hand, I moved on to model training and hyper-parameter tuning.

Figure 5



Predictive Firepower: Deploying XGBoost to Lock On to Churn

I approached modeling as a staged bake-off: start simple, climb the complexity ladder, and bail out the moment things look too good to be true. My first Logistic Regression run finished instantly, but a bigger shock followed when every model, no matter the algorithm, returned

perfect 1.00 scores on accuracy, precision, recall, and ROC-AUC. As a result, that clean never happens in live-service telemetry, so I hit pause and dug for leakage.

A quick audit revealed two culprits. First, the raw churn flag had slipped back into the feature matrix during an early join; second, the newly one-hot column for an otherwise unused legend failed to load because its name began with a digit, causing mis-aligned rows that let the model peek at the target during train/test split. After stripping the extra churn column from X, fixing the legend-column name, and rerunning the entire preprocessing pipeline, metrics dropped to plausible levels and diverged by algorithm as expected.

- ★ **Logistic Regression (baseline)** Accuracy 0.83, F1 0.73, Recall 0.85, ROC-AUC 0.89
- ★ **Random Forest (tuned)** Accuracy 0.90, F1 0.83, Recall 0.93, ROC-AUC 0.95
- ★ **XGBoost (final)** Accuracy 0.91, F1 0.84, Recall 0.94, ROC-AUC 0.96
- ★ **Final hyper-parameters** max_depth = 6, learning_rate = 0.05, n_estimators = 500, subsample = 0.8, colsample_bytree = 0.8
- ★ **Top importance signals** days_since_last_match, session_frequency, legend_diversity, avg_kills

With recall at 94 % and no hint of leakage, the XGBoost model reliably spots at-risk players a full week before they disappear, giving EA's live-ops team a concrete window for intervention.

Final Circle Showdown: Results and Evaluation

After I fixed the leakage and reran the pipeline on the untouched test split, performance settled at realistic levels and clearly favored the final XGBoost model. The model meets the goal of flagging likely churners under the 7-day definition while keeping false alarms manageable. Importances indicate cadence signals matter more than raw combat stats, aligning with the retention story in Season 15.

- ★ **Final test metrics (XGBoost):** Accuracy **0.91**, F1 **0.84**, Recall (churn) **0.94**, ROC-AUC **0.96**.
- ★ **Error profile:** Remaining errors are mainly false positives (players flagged who return the next week).
- ★ **Key drivers (model importances):** `days_since_last_match`, `session_frequency`, `legend_diversity`, `avg_kills`.

Respawn with Strategy: Recommendations for EA

I'm anchoring these directly to what I observed in my EDA and modeling: the churn label triggers at **7 days of inactivity**, and the strongest behavior signals I found were **kill count** (significant difference by retention, Mann–Whitney U = 23,464; p = 0.048), plus consistent patterns with **higher squad placement** and **revive activity**. The final model performs well on the test split and is designed to surface at-risk players under that 7-day definition.

★ **Time-boxed re-engagement within the 7-day window**

- ➢ **Action:** Use the model's churn probabilities to trigger a light-touch outreach when a player approaches 5–7 days since the last match (the same window used to define churn).
- ➢ **Why:** The target is defined at 7 days, so intervening just before that threshold aligns with how the label is constructed and how the model learned risk.

★ **Momentum goals around eliminations and support play**

- ➢ **Action:** Offer short, session-level goals tied to **kills** and **revives** like, "earn N eliminations" or "complete N revives" this session that can be met in one or two play sessions.
- ➢ **Why: Kills** show a statistically significant separation between retained and churned players, and **revive behavior** trends higher among players who return; both appeared as useful signals in EDA.

★ **Placement-focused nudges for ranked players**

- ➢ **Action:** Surface gentle, near-term placement milestones such as, "aim for top-X finish next match" to reinforce progress without requiring long grinds.
- ➢ **Why: Higher squad placement** correlates with better retention in my EDA, indicating that reinforcing competitive progress can help keep players active.

Saving Legends Before They Logout: Conclusion

This project isolated churn as a **7-day inactivity** problem within a single, stable meta (Season 15 "Eclipse") and backed it with a cleaned dataset of **499 matches × 35 features**. My EDA showed meaningful behavioral separation, most notably **kills** (Mann–Whitney U = 23,464; p = 0.048) and consistent trends with **placement** and **revive activity**. After catching and fixing a leakage issue that briefly produced unrealistic 100% scores, the final **XGBoost** model delivered strong out-of-sample performance (Acc 0.91, F1 0.84, Recall on churn 0.94, ROC-AUC 0.96), with cadence features (**days_since_last_match**, **session_frequency**, **legend_diversity**) emerging as the most useful signals.

Retention is about cadence, not just combat. Players who play *recently and often* stay; big kill/damage numbers matter less. Because I define churn as **7 days of inactivity** and train on **cadence signals** that move *before* that cutoff, the model's risk score rises as activity slows, giving EA a **clear early-warning signal** prior to day 7. With **0.94 recall** on the churn class, it catches most soon-to-churn players, making day-5/6 nudges a practical, data-driven intervention window.