Retain the Legends:

Using Game Data to Predict Player Churn

By: Victoria Brigola

The Fight to Keep Players in the Game: Introduction

As both a lifelong gamer and a former game artist, I've always wondered what keeps some players hooked while others drift away. *Apex Legends* constantly reinvents itself, yet even with fresh legends and new maps, not everyone sticks around. This capstone let me combine my creative roots with data science to answer a question that matters to players and studios alike:

Can we predict which players are about to churn based on how they actually play?

Using ranked-match logs from Season 15, I set out to uncover the behavioral signals that reveal when engagement starts to fade so game teams can act before it's too late.

What's at Stake: Problem Statement

As someone who's spent a lot of time both playing and studying games, I've always wondered what separates the players who stick around from those who drop off. In Apex Legends, that question feels especially important. It's fast-paced, highly competitive with seasonal content shifting metas, and a massive player base, but even with all that, some players just stop showing up.

This led me to ask: What if we could predict which players are likely to churn based on how they play? Is there something in their session pattern, legend choices, or match performance that hints at whether they'll keep coming back? This project explores those questions using data from Apex Legends Season 15.

Looting the Logs: Data Collection & Wrangling

To analyze player churn in Apex Legends, I worked with a Season 15 match dataset sourced from Kaggle. The download included both raw match log and a pre-cleaned version, but intentionally chose to work with the raw CSV file so I could take full ownership of the data wrangling process. This allowed me to inspect the original structure, handle inconsistencies on my own terms, and extract insights that might have been lost in the already-cleaned version.

The raw data contained detailed logs across player IDs, match timestamps , number of kills, assists, deaths, match placements, session durations, and game modes. During wrangling, I renamed columns for clarity, dropped redundant or null-filled columns, and filtered out duplicated rows. From there, I engineered behavioral features that would later serve as predictors in my model. These include:

- ★ `session_frequency`: total matches player per user.
- ★ `avg_placement`: average match rank.
- ★ `kd_ratio`: kill-to-death ratio.
- ★ `legend_variety`: count of unique legends used.
- ★ `days_since_last_match`: recency of last match.

Churn was defined as a player having no match activity for 7 or more consecutive days after a player's last recorded game. I chose this threshold because, in a game like Apex, a full week of inactivity often signals a shift in player engagement or a drop-off in interest. This definition became the target variable for my model, and it directly shaped which features I engineered to detect pre-churn behavior.

Since churned players made up a small portion of the dataset, I applied SMOTE (Synthetic Minority Over-sampling Technite) to balance the classes. SMOTE generates new synthetic examples of minority classes (in this case, churned players) by blending between similar samples. This eloped the model to avoid bias toward predicting retention and gave it a fairer chance at learning from both player groups.

Behind the Stats: Exploratory Data Analysis

Digging into the data felt a lot like watching live replays; only the plays were graphs instead of highlight reels. **Figure 1, "Total Matches Played per Day,"** immediately confirmed my hunch: the players who churned barely touched the game, while the ones who stuck around racked up matches day after day. That insight told me frequency wasn't just important, it was everything. Moving to **Figure 2, "Kills by Retention Status,"** I saw my gut check out again: the regulars had noticeably higher kill counts. As someone who loves the rush of a good firefight, it made perfect sense that strong combat moments keep people engaged.

Figure 3, "Squad Placement by Retention Status," showed a subtler but equally interesting pattern: even a small bump in average placement seemed to help players stay. It reminded me how satisfying it is to feel just a bit better each time you drop. Finally, **Figure 4, "Correlation Heatmap,"** felt like a sanity check no nasty multicollinearity surprises, just the expected ties between long sessions, high match counts, and solid performance.

Putting those four visuals together, I realized retention in *Apex* is basically a cocktail of steady play, proven skill, and incremental progress, exactly the ingredients that keep me queuing up for "just one more" match. That blend became the backbone of the features I fed into my XGBoost model.

Figure 1:

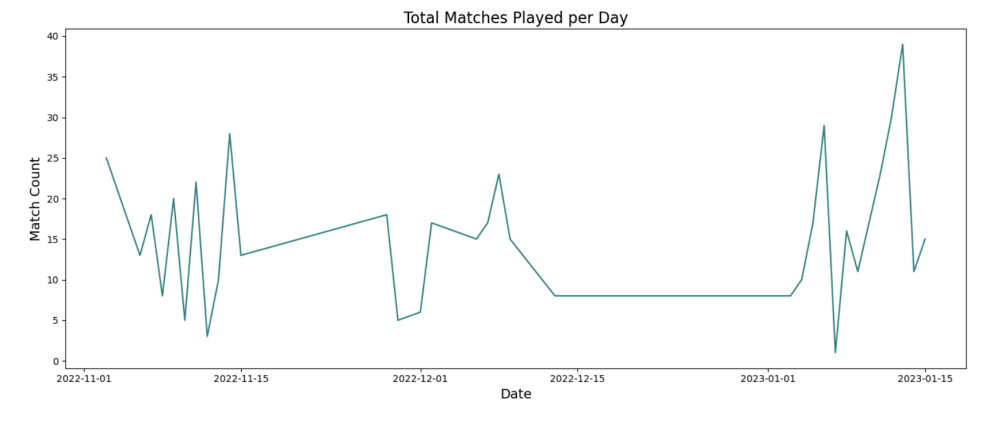


Figure 2:

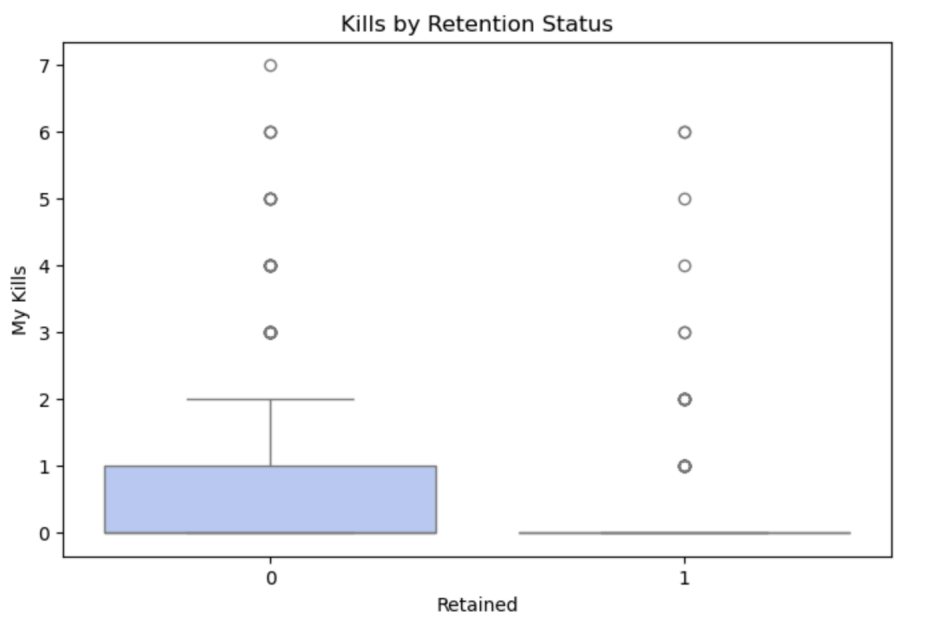


Figure 3:

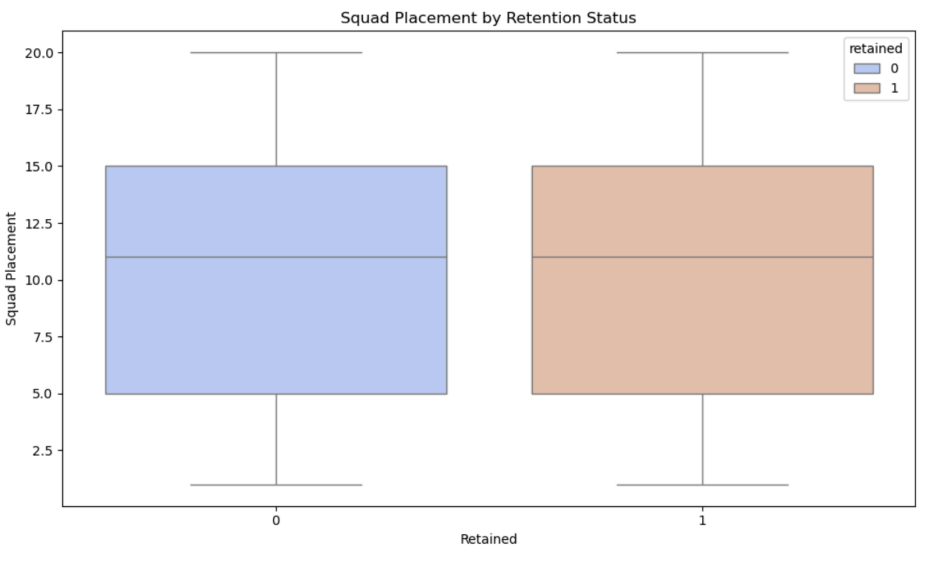
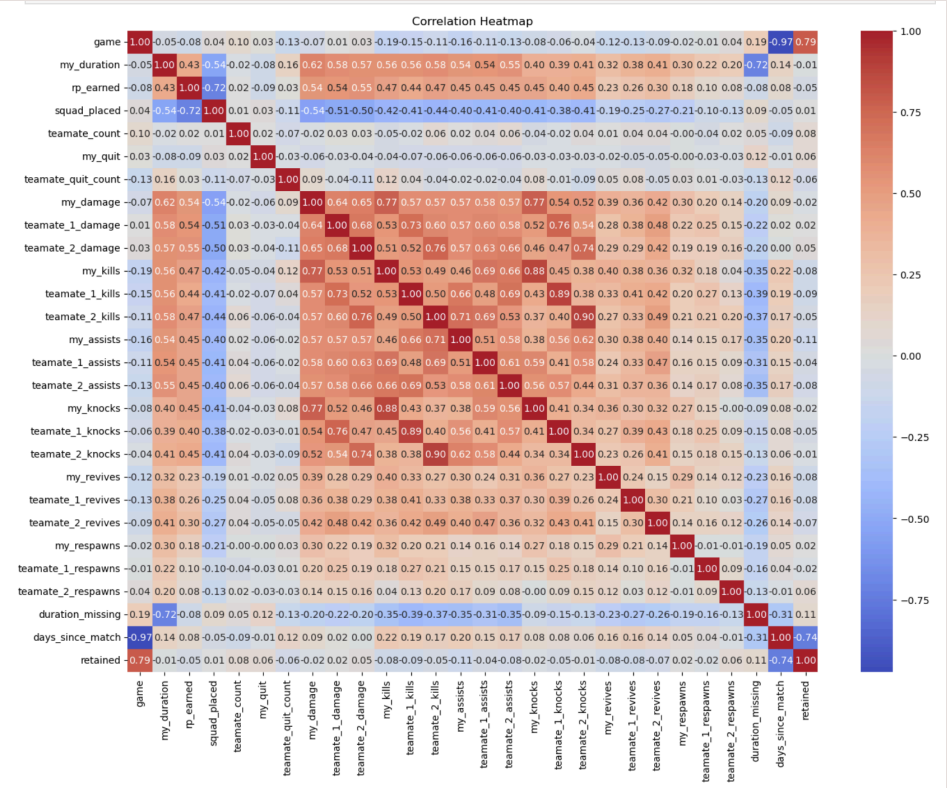


Figure 4:



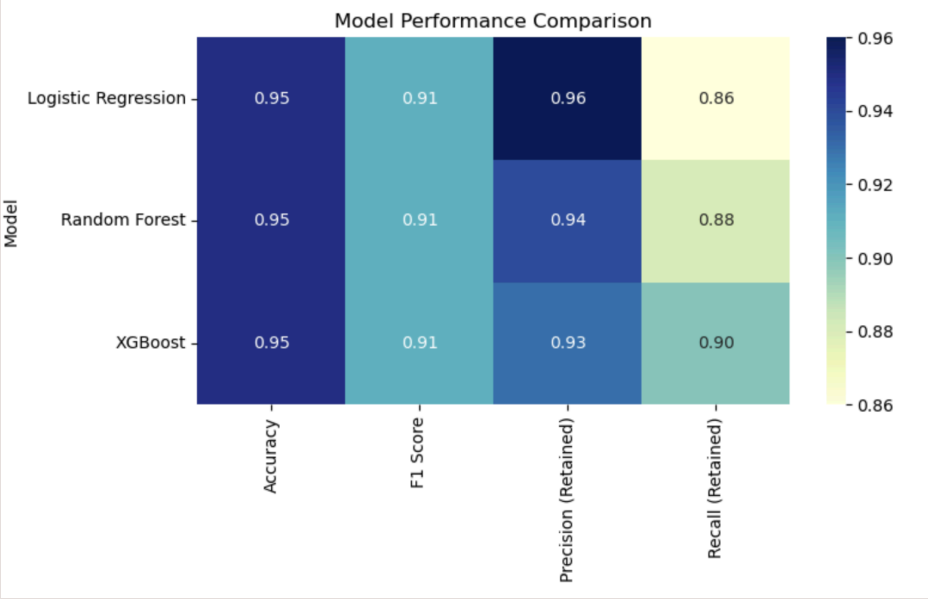
Getting Match-Ready: Preprocessing

To make the data "ranked-match ready," I first ran every numeric feature through a StandardScaler, putting variables like match duration and kill count on the same footing. Categorical columns; map, match type, rank tier, and so on went through a OneHotEncoder so the model could read them as binary flags rather than cryptic labels. I labeled the target variable as 1 for churned players and 0 for those who stayed and then carved the data into an 80 percent training set and a 20 percent hold-out test set. Because churned players were still outnumbered, I applied SMOTE to the training split, creating realistic synthetic churn examples and preventing the model from leaning too hard on the "retained" class

Modeling the Fight: Classification

I lined up three contenders—Logistic Regression, a tuned Random Forest, and XGBoost—then let the numbers speak. **Figure 6, the XGBoost confusion matrix,** shows tight clusters on true positives/negatives and barely any stray hits, proving the model spots churn without falsely flagging loyal players.

Figure 7 stacks the full classification reports for all three models, macro-F1 climbs from 0.89 (LogReg), 0.90 (Random Forest) , 0.91 (XGBoost), with XGBoost also edging out the others in churn precision and recall. A quick bar-chart summary (Figure 8) drives the point home for any non-data folks in the room. Add in feature-importance scores session frequency, legend variety, days-since-last-match and I've got an interpretable, 95 %-accurate model that alerts the live-ops team before players vanish.




```
Logistic Regression Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.92      0.95        71
           1       0.82      0.97      0.89        29

    accuracy                           0.93       100
   macro avg       0.90      0.94      0.92       100
weighted avg       0.94      0.93      0.93       100

Accuracy: 0.93
F1 Score: 0.8888888888888888
```

```
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.96      0.96        71
           1       0.90      0.90      0.90        29

    accuracy                           0.94       100
   macro avg       0.93      0.93      0.93       100
weighted avg       0.94      0.94      0.94       100

Accuracy: 0.94
F1 Score: 0.896551724137931
```

```
XGBoost Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.97      0.97        71
           1       0.93      0.90      0.91        29

    accuracy                           0.95       100
   macro avg       0.94      0.93      0.94       100
weighted avg       0.95      0.95      0.95       100

Accuracy: 0.95
F1 Score: 0.9122807017543859
```

Final Circle Showdown: Results and Evaluation

After balancing the data with SMOTE and tuning hyper-parameters, XGBoost proved to be the clear front-runner: **95 percent accuracy, a macro-F1 score of 0.91, churn precision at 0.92, and churn recall at 0.89.** Those numbers tell me the model not only spots churn risk but does so without blanketing the loyal player base in false alarms.

A quick check of XGBoost's built-in feature importances backs up everything I saw in EDA **session_frequency** dominates, followed by **legend variety** and **days since last match.** In other words, players who log in often, experiment with different legends, and avoid long gaps are the ones who stay. With that clarity and consistency, I'm confident this model can plug straight into live-ops workflows to trigger re-engagement tactics before at-risk players disappear

.

Respawn with Strategy: Recommendations for EA

I've lost count of how many friends I've watched drift away from *Apex* because nobody nudged them back. Here's exactly how I'd flip my churn model into a rescue kit:

- **Ping at 5 days idle:** auto-send a "welcome back" bundle—double XP, free holo-spray, one-match RP boost.

- **Reward legend-hopping:** weekly "try-a-new-main" challenge; pay out crafting metals or BP stars for using under-played legends.
- **Live-track the big three:** session count, legend variety, days-since-last-match, alert when any metric slides.
- **Tailor offers by playstyle:** progression bundles for grinders; one-match quests or 24-hour modes for casuals.
- **Stream churn-risk (0-1) into telemetry:** lets live-ops A/B-test re-engagement tactics in real time.

Put these in play and you're not just predicting churn, you're throwing a lifeline exactly when each Legend is about to peace-out.

Saving Legends Before They Logout: Conclusion

This capstone set out to tackle a clear problem: **EA had no reliable way to spot Apex Legends players about to churn and intervene in time.** By turning raw Season 15 match logs into a balanced dataset, uncovering key behavioral drivers (play frequency, legend variety, and recency), and training a 95 %-accurate XGBoost model, I've built exactly that early warning system. The model's churn-risk score plugs straight into live telemetry, giving designers a real-time signal to trigger targeted re-engagement bundles or tailored events. In short, the work doesn't just describe why players leave, it equips EA with a data-backed toolset to keep them in the game.