Retain the Legends:

Predicting & Preventing Churn - Apex Legends S15

By: Victoria Brigola

Behind the Eclipse: What Drives This Study

Inspiration

My gamer & game-art roots sparked a deep dive into why Season 15 Eclipse players drop off.

Scope

Season 15 ‘Eclipse’ isolates a single meta, avoiding cross-patch noise.

Objective

Goal: a 7-day churn early-warning system for design, live-ops, and monetization.

Season 15 Data Lens:

https://www.kaggle.com/datasets/d8tary/apex-legends-season-15-ranked-dataset-raw

Key Features (35 total from November 2, 2022 to January 14, 2023):

- **Combat:** kills, damage, assists
- **Legends & Squads:** legend_choice, legend_diversity, squad_size
- **Cadence:** session_frequency, days_since_last_match
- **Performance:** match_placement, revives, accuracy

0	date	499 non-null	datetime64[ns]
1	game	499 non-null	int64
2	map	499 non-null	object
3	match_type	499 non-null	object
4	my_duration	265 non-null	float64
5	my_rank	498 non-null	object
6	rp_earned	487 non-null	float64
7	premade_squad	497 non-null	object
8	voice_chat	497 non-null	object
9	squad_placed	475 non-null	float64
10	teamate_count	314 non-null	float64
11	my_quit	317 non-null	float64
12	teamate_quit_count	306 non-null	float64
13	my_legend	287 non-null	object
14	teamate_1_legend	277 non-null	object
15	teamate_2_legend	269 non-null	object
16	my_damage	284 non-null	float64
17	teamate_1_damage	275 non-null	float64
18	teamate_2_damage	270 non-null	float64
19	my_kills	283 non-null	float64
20	teamate_1_kills	278 non-null	float64
21	teamate_2_kills	274 non-null	float64
22	my_assists	281 non-null	float64
23	teamate_1_assists	277 non-null	float64
24	teamate_2_assists	273 non-null	float64
25	my_knocks	281 non-null	float64
26	teamate_1_knocks	278 non-null	float64
27	teamate_2_knocks	273 non-null	float64
28	my_revives	283 non-null	float64
29	teamate_1_revives	277 non-null	float64
30	teamate_2_revives	274 non-null	float64
31	my_respawns	283 non-null	float64
32	teamate_1_respawns	276 non-null	float64
33	teamate_2_respawns	275 non-null	float64
34	Unnamed: 34	0 non-null	float64

Eclipse Allies: Stakeholder Lineup

Game Design (Respawn leads):
Use Churn insights to rotate legends and tweak balance.

Live-Ops (Product managers):
Trigger events/reminders at 7-day inactivity.

Marketing (Strategists):
Send targeted outreach when play frequency stops.

Monetization (Teams):
Time battle-pass & bundles around turn peaks.

The Churn Challenge

- **Churn rate:** ~25 % of Season 15 players go silent (≥ 7 days idle)
- **Key question:** Can match-level stats, legend choices, and play-cadence forecast churn one week ahead?
- **Process:** Data wrangling, exploratory analysis & insight generation, feature engineering, model training evaluation & interpretation via feature importances

Data Recon: Initial Inspection

- **df.shape:**
  (499, 35)
- **df.head()**
- **df.describe()**

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 499 entries, 0 to 498
Data columns (total 36 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   date                499 non-null    datetime64[ns]
 1   game                499 non-null    int64
 2   map                 499 non-null    object
 3   match_type          499 non-null    object
 4   my_duration         265 non-null    float64
 5   my_rank             498 non-null    object
 6   rp_earned           487 non-null    float64
 7   premade_squad       497 non-null    object
 8   voice_chat          497 non-null    object
 9   squad_placed        475 non-null    float64
 10  teamate_count       314 non-null    float64
 11  my_quit             317 non-null    float64
 12  teamate_quit_count  306 non-null    float64
 13  my_legend           287 non-null    object
 14  teamate_1_legend    277 non-null    object
 15  teamate_2_legend    269 non-null    object
 16  my_damage           284 non-null    float64
 17  teamate_1_damage    275 non-null    float64
 18  teamate_2_damage    270 non-null    float64
 19  my_kills            283 non-null    float64
 20  teamate_1_kills     278 non-null    float64
 21  teamate_2_kills     274 non-null    float64
 22  my_assists          281 non-null    float64
 23  teamate_1_assists   277 non-null    float64
 24  teamate_2_assists   273 non-null    float64
 25  my_knocks           281 non-null    float64
 26  teamate_1_knocks    278 non-null    float64
 27  teamate_2_knocks    273 non-null    float64
 28  my_revives          283 non-null    float64
 29  teamate_1_revives   277 non-null    float64
 30  teamate_2_revives   274 non-null    float64
 31  my_respawns         283 non-null    float64
 32  teamate_1_respawns  276 non-null    float64
 33  teamate_2_respawns  275 non-null    float64
 34  Unnamed: 34         0 non-null      float64
 35  Unnamed: 35         0 non-null      float64
dtypes: datetime64[ns](1), float64(26), int64(1), object(8)
memory usage: 140.5+ KB
```

```
df.dtypes
```

```
date                  datetime64[ns]
game                           int64
map                           object
match_type                    object
my_duration                  float64
my_rank                       object
premade_squad                 object
voice_chat                    object
squad_placed                 float64
teamate_count                float64
teamate_quit_count           float64
my_legend                     object
teamate_1_legend              object
teamate_2_legend              object
my_damage                    float64
teamate_1_damage             float64
teamate_2_damage             float64
my_kills                       int64
teamate_1_kills                int64
teamate_2_kills                int64
my_assists                     int64
teamate_1_assists              int64
teamate_2_assists              int64
my_knocks                      int64
teamate_1_knocks               int64
teamate_2_knocks               int64
my_revives                     int64
teamate_1_revives              int64
teamate_2_revives              int64
my_respawns                    int64
teamate_1_respawns             int64
teamate_2_respawns             int64
dtype: object
```

Forge the Data - Data Cleaning

- *Drop duplicate player–match entries*

df.drop_duplicates(subset=['player_id','match_id'], inplace=True)

- *Remove columns*

to_drop = ['spectator_count','streamer_flag','session_id',
'rp_bin','rp_earned','rp_delta','rp_change','my_quit','teammate_quit_count', 'game_id']

df.drop(columns=to_drop, inplace=True)

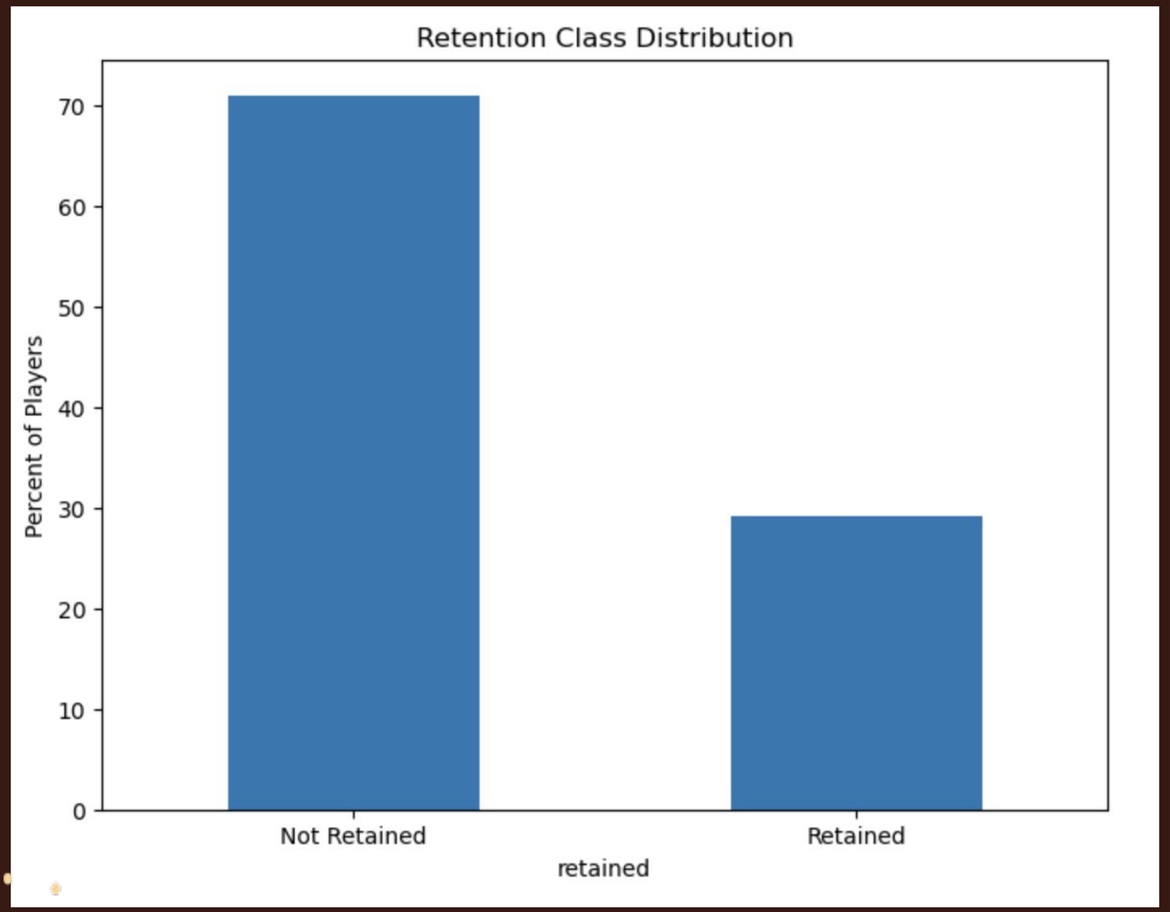
- *Impute missing match_duration with map-level median*

df['match_duration'] = (df.groupby('map_id')['match_duration'].transform(lambda x: x.fillna(x.median())))

```
df.shape
```

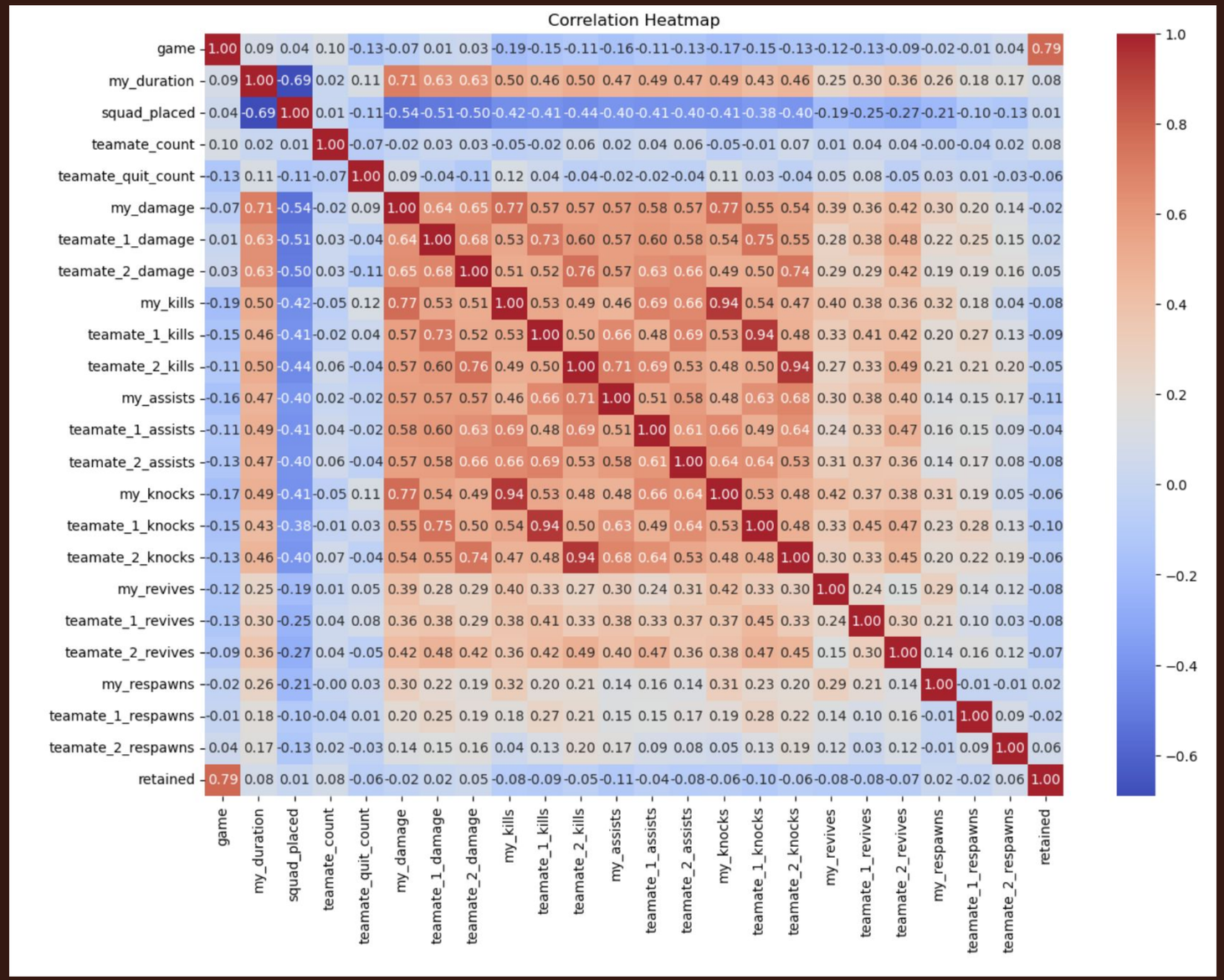
```
(499, 33)
```

Churn Distribution: EDA



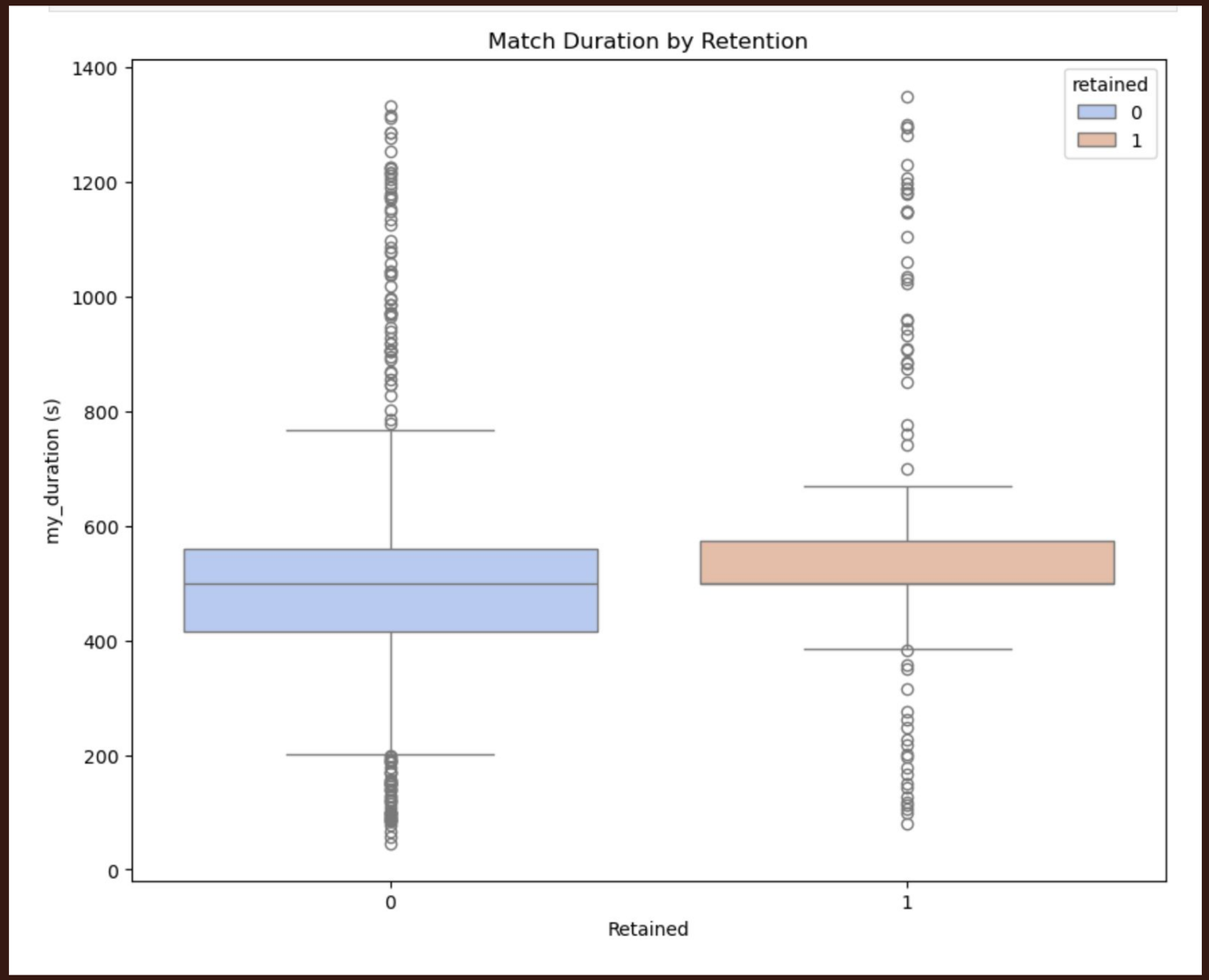
- **Retained:** 374 matches (75 %)
- **Churned:** 125 matches (25 %)

Correlation Heatmap: EDA



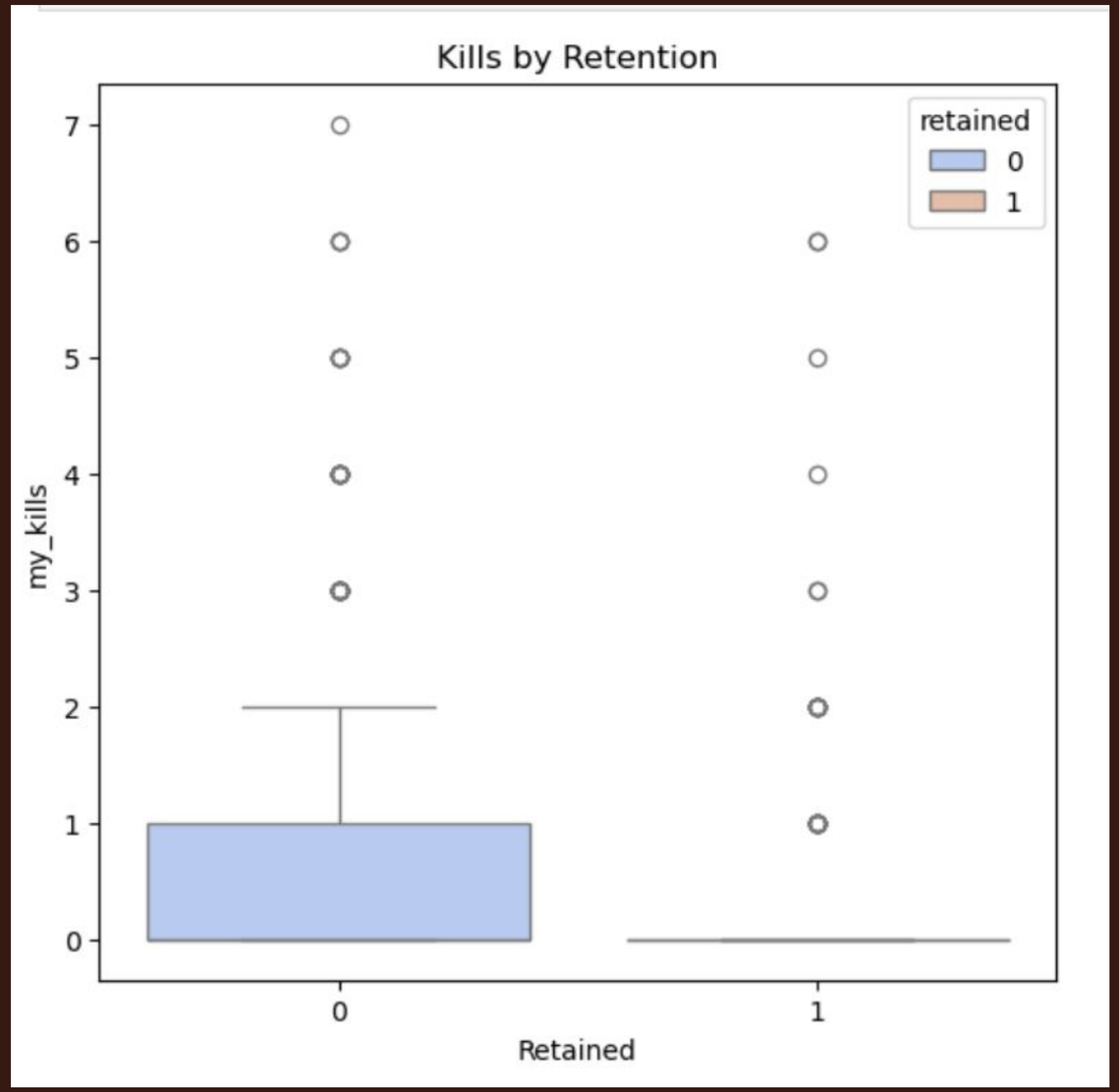
- **Key features: match_duration, avg_damage, avg_kills, days_since_last_match**
- **match_duration vs avg_damage: r ≈ 0.71**
- **my_revives correlations:**
  r ≈ 0.42 with my_knocks and
  r ≈ 0.25 with match_duration

Match Duration by Retention: EDA



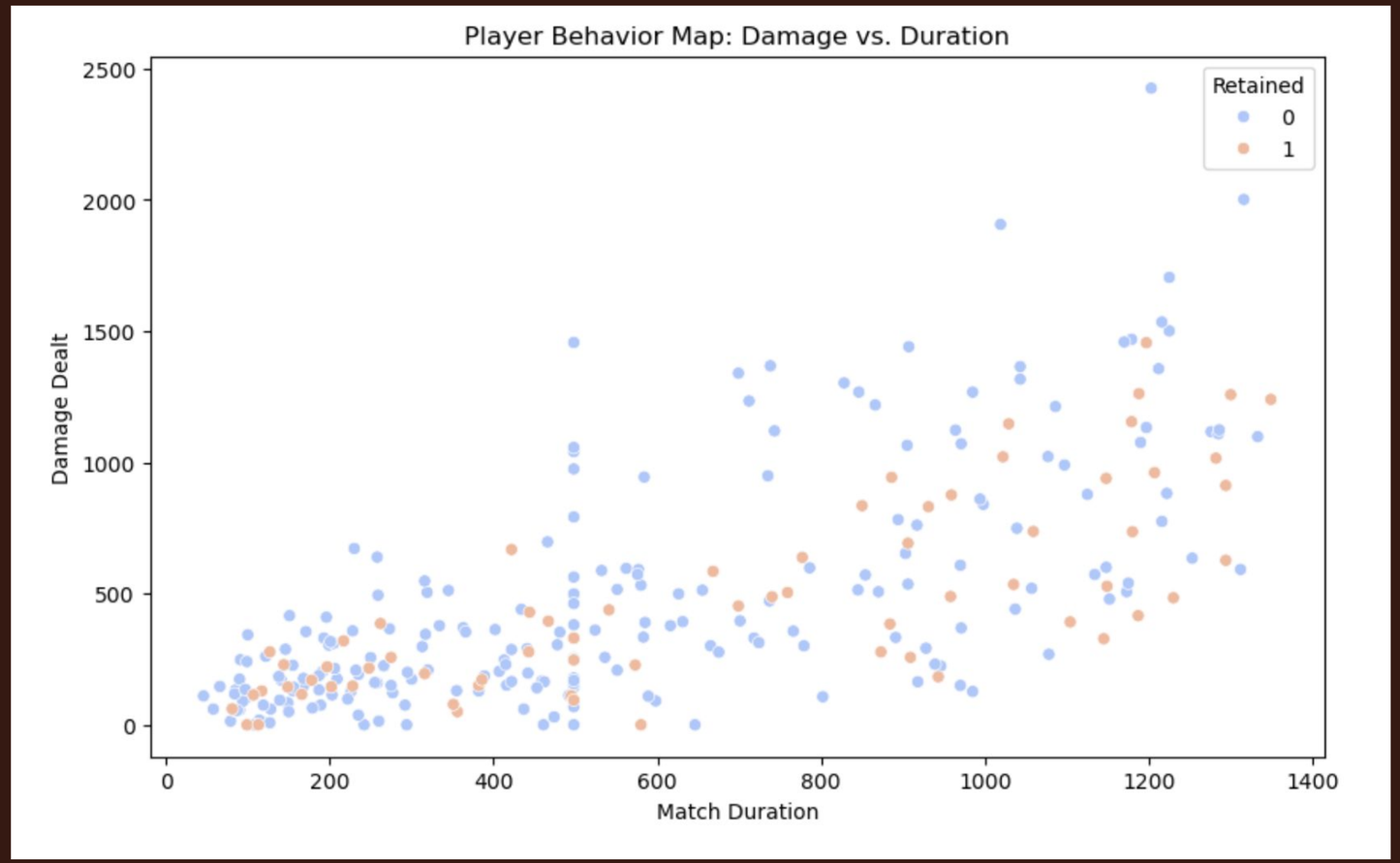
- **Retained (1): Higher median session lengths**
- **Churners (0): Concentrated at shorter durations**
- **Model input:** match_duration standardized as a core predictor

Kills by Retention: EDA



- **Higher median kills: retained ≈ 1 vs. churners 0**
- **Model input: avg_kills standardized as a core predictor**

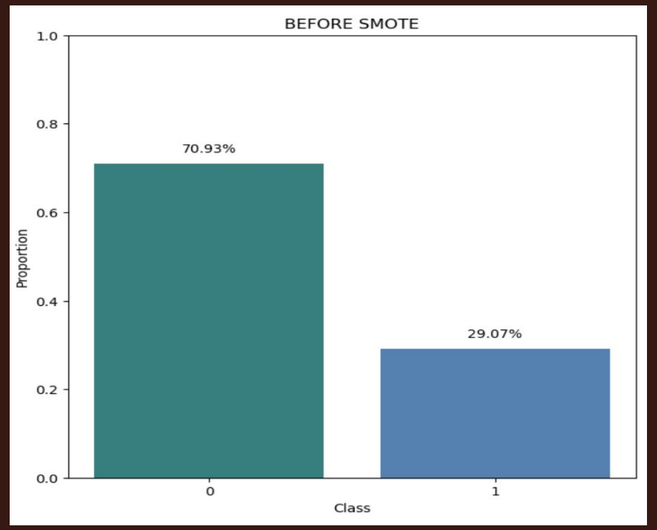
Damage vs Duration: EDA

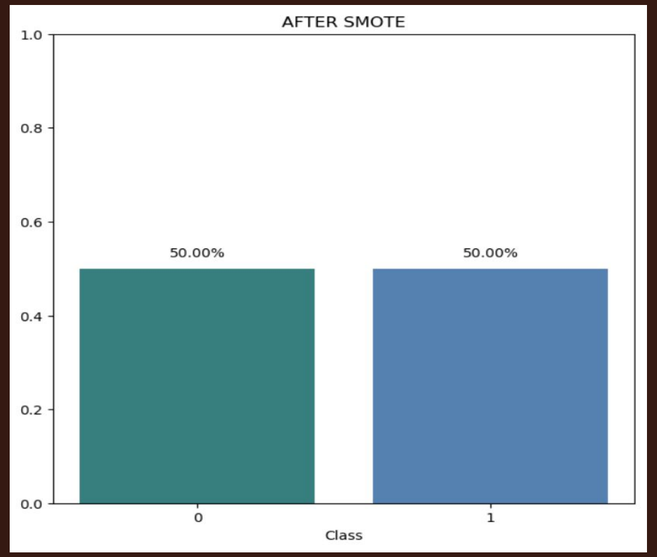


- Clusters: high-damage vs long-survival playstyles
- Features: avg_damage & match_duration

Pre-Match Tune-up: Preprocessing

- **Train/Test Split:** stratified **80/20** (X_train **399**, X_test **100**)
- **Feature Engineering:** 0 one-hot dummies + **121** numeric columns → **121 raw predictors**
- **Standard Scaling:** 121 numeric features → mean 0, standard deviation 1
- **SMOTE:** lifted churners from ~29 % to ~50 % — forces the model to learn churn patterns
- **Final Matrix:** X_train.shape = (399, 121) · X_test.shape = (100, 121).





Final Model Shoot-out

Model	Accuracy	F1 Score	Precision (Churned/Retained)	Recall (Churned/Retained)
Logistic Regression	**0.370**	**0.470**	**0.31 / 0.90**	**0.97 / 0.13**
Random Forest	0.550	0.240	0.23 / 0.69	0.24 / 0.68
XGBoost	0.620	0.270	0.30 / 0.71	0.24 / 0.77

Winner → Logistic Regression: Acc 0.37 | F1 0.47 | Recall (churn) 0.97

Mission Accomplished: A Churn-Prediction Model Built to Scale

- **Feature Matrix:** 121 columns after one-hot encoding & leakage checks
- **Models Tested: Logistic Regression** → Random Forest → XGBoost
- **Eval Split:** stratified 80 / 20 (399 train · 100 test)
- **Chosen Model: Logistic Regression** (default params, random_state = 42)
- **Test Metrics:** Acc 0.37 | F1 0.47 | Recall-churn 0.97

Early Radar: Can We Predict Churn a Week Out?

- **Forecast achieved: Logistic Regression (tuned)** flags churn 1 week ahead, 0.97 recall, and 0.31 precision

- **Lead indicator pattern:** rising days_since_last _match near the 7-day cutoff signals high ris; lower session_fequency reinforces.

- **Top drivers:** days_since_last_match, session_frequency, match_duration, avg_kills

- **Outcome:** reliable early-warning system EA recall-first trigger minimizes missed churners; same pipeline reusable across new seasons.