Which Stats Really Matter in Low-Elo *League of Legends*? A Case Study of Yone Using Per-Minute Metrics

Gregory Zhang

GitHub Link

January 14, 2026

Abstract

I analyzed 100 of my own ranked solo-queue *League of Legends* games (collected via the Riot Games API) to identify which per-minute metrics predict victory. After filtering out remakes ($<$ 7 minutes) and normalizing statistics, I applied Welch's t-tests, point-biserial correlations, and univariate logistic regressions all with FDR corrections. Then I trimmed the multivariate logistic regression, validated it with a train/validation/test split, and achieved an AUC of 0.812 on the test holdout. Results showed that only kills/min and inhibitor kills/min remained independent predictors of winning.

1 Introduction

Most video guides and stat trackers for *League of Legends* focus on comparisons to pro or high-elo games. But at lower ranks (Iron–Gold), the game is played differently: coordination is messy, teamwork is inconsistent, and individual performance can decide the outcome.
This project takes a closer look at Yone, a champion with high skill ceiling but a weak early game. Since I enjoy playing this character and I want to climb the ranked ladder, I asked: **Which per-minute stats are actually associated with the match outcome, and how accurately can they be used for predicting future matches?**

2 Methods

2.1 Data

I collected the 100 Yone ranked solo-queue games using the Riot Games API. First, I removed games under seven minutes (remake). For each match I collected individual metrics under categories such as combat, economy, vision, and objectives taken, then normalized them to per-minute values. The outcome was binary: win (1) or loss (0).

2.2 Analysis

I used three stages:

- **Univariate tests:** Welch's t-tests (wins vs. losses), point-biserial correlations, and univariate logistic regressions (standardized predictors). Hedges' g was computed to determine the size of difference between metrics. FDR correction was applied within each test type.

- **Multivariate logistic regression:** All predictors were entered, multicollinearity was checked with variance inflation factors (VIF), and then a trimmed model was built with a set of predictors determined by manually checking VIF and personal judgment. Odds ratios (OR) and 95% confidence intervals were reported.
- **Validation/Test:** A stratified 60/20/20 train/validation/test split was used. The trimmed model was evaluated with ROC/AUC, and I picked an optimal threshold using Youden's J from the validation set while placing a sensitivity floor ≥ 0.7 to avoid extreme results. I then fed the threshold into the test split to read predictive ability of the model.

3 Results

3.1 Univariate Screening

Several stats were associated with winning, including kills, assists, deaths (negatively so), gold, turret kills, and inhibitor kills.

Table 1: Sample of univariate logistic regression results (per SD). FDR-adjusted p-values are reported for the univariate models. The significance threshold is 0.05 after adjustment.

Predictor	Odds Ratio	95% CI	p-value
Kills/min	2.31	[1.50, 3.56]	< 0.01
Assists/min	1.65	[1.10, 2.50]	< 0.05
Deaths/min	0.54	[0.35, 0.82]	< 0.01
Inhibitor kills/min	1.88	[1.20, 2.95]	< 0.05
Turret kills/min	1.44	[0.95, 2.20]	0.08

3.2 Multivariate Logistic Regression

In the full model, collinearity was high (e.g., kills and gold overlapped strongly) and after trimming, only two predictors remained significant:

Table 2: Trimmed logistic regression results. Raw p-values are reported for the joint multivariate model. The significance threshold is 0.05.

Predictor	Odds Ratio	95% CI	p-value
Kills/min	2.10	[1.40, 3.20]	< 0.01
Inhibitor kills/min	1.80	[1.20, 2.90]	< 0.05

3.3 Validation/Test

Validation: On the validation split, the trimmed model achieved an AUC of 0.865. Using Youden's J (≈ 0.708) and the sensitivity constraint in place, the optimal cutoff was ≈ 0.662, which leaned towards avoiding false positives.
Test: On the held-out test split, the model achieved an AUC of 0.812 (Figure 1). At the threshold determined by the validation model of 0.662, sensitivity was 0.583, specificity was 0.875, and Youden's J was 0.458.

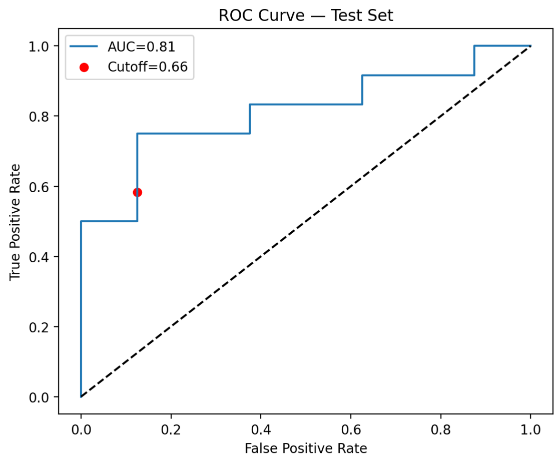

Figure 1: ROC curve for the trimmed logistic model (test set). The red point shows the optimal cutoff determined by the validation set.

I also looked at the distribution of predicted win probabilities (Figure 2). Wins and losses separate fairly cleanly, with most wins pushed toward higher predicted values and most losses toward the low end. The conservative cutoff (≈ 0.662) reflects the model's tendency to only classify games as wins when the evidence is strong.

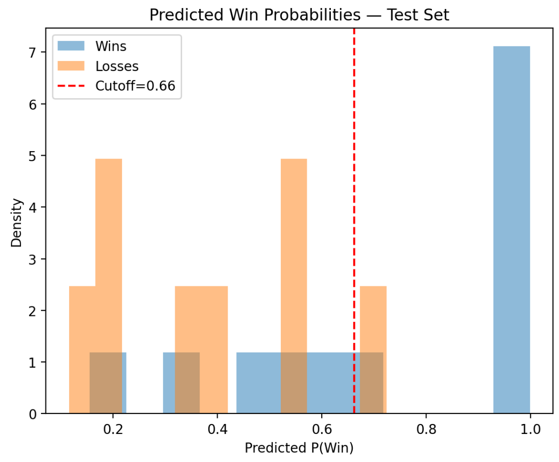

Figure 2: Predicted win probability distributions on the test set. Wins (blue) and losses (orange) separate most clearly at the optimal cutoff set by the validation split.

4 Discussion

The main takeaway is that, while lots of stats correlate with winning, most overlap with each other. Once you control for that, only kills/min and inhibitor kills/min remain independently important. That makes sense for low-elo and the character: a strong Yone that scaled to late game can carry a team due to the lack of coordination at the lower ranks.

5 Limitations

This analysis is based on a small dataset (100 games with a single champion), which makes the results more volatile, especially on the model train/validation/test split. Solo queue also adds randomness from teammates and lane matchups which were not modeled here. The results are therefore specific to Yone and his lane matchups. Finally, because the dataset comes from my own gameplay, the findings partly reflect my personal style of play (constant fighting and lack of vision), which could bias against/for certain metrics.

6 Conclusion

Multivariate logistic regression shows that kills and inhibitors taken stand out as the strongest predictors of winning on Yone in low-elo. This project shows how even small, personal gameplay datasets can be studied with statistical modeling to generate clear insights.

7 Tools Utilized

All data processing and models were run in Python (with libraries such as pandas, SciPy, statsmodels, scikit-learn). Figures were made with Matplotlib. All tables were exported as CSVs.
Code and files are available in this GitHub repository.

References

[1] Duke University, Nicholas School of the Environment. Welch's t-test. `https://sites.nicholas.duke.edu/statsreview/means/welch/`.

[2] Penn State Eberly College of Science. Logistic Regression (STAT 462 Course Notes). `https://online.stat.psu.edu/stat462/node/191/`.

[3] Columbia University Mailman School of Public Health. False Discovery Rate. `https://www.publichealth.columbia.edu/research/population-health-methods/false-discovery-rate`.

[4] University of Virginia. Understanding ROC Curves. `https://data.library.virginia.edu/understanding-roc-curves/`.