Will my team win this season?



Somony Pou

13 OCT 2026

Given only the first 10 players I drafted, what is the chance my team finishes #1 in total points this fantasy season(end of week 22).



My league: Yahoo league 73783

10 teams, 10 starting slots, bench and injured reserve doesn't count. Each week my team faces one other team, and whoever scores the most fantasy points wins that week

Scoring(translates from the real games):
points 1
rebounds 1.2
assist 1.5
steals 3
block 3
turnovers -1,

Only games played count. Teams play different # of games and different teams each week.

Why did I chose this project?

The outcome of who wins is uncertain. Only 1 team will win this season, so the **chance of winning** is more valuable for an answer rather than who will win.

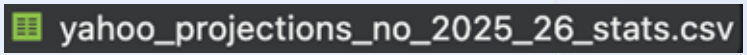
Data



4 seasons worth of player stats



10 players from team and 9 opponents' starters.



Yahoo projections for the 6 starters with no 2025-2026 stats(new drafted players this season).

Simulated league data

✓ **Iris**
species known

✓ **Product data**
career known

✗ **My fantasy league**
no past results

Real stats
4 seasons of player stats

→

Last season's stats
points per game and games played

→

Simulate drafts
12,000 drafted teams of 10 players

→

Check the answer
did the team really finish #1 the next season?

Each team becomes one row: 2 features + the answer

Talent	Availability	Finished #1?
363.1	73.6	1
387.1	68.0	0
331.6	72.9	0
388.5 (mine)	**72.1**	**?**
384.6 (kenny's)	**41.0**	**?**

Rows 1-3: simulated training teams. Yellow: my team and Kenny's, with the answer to predict.

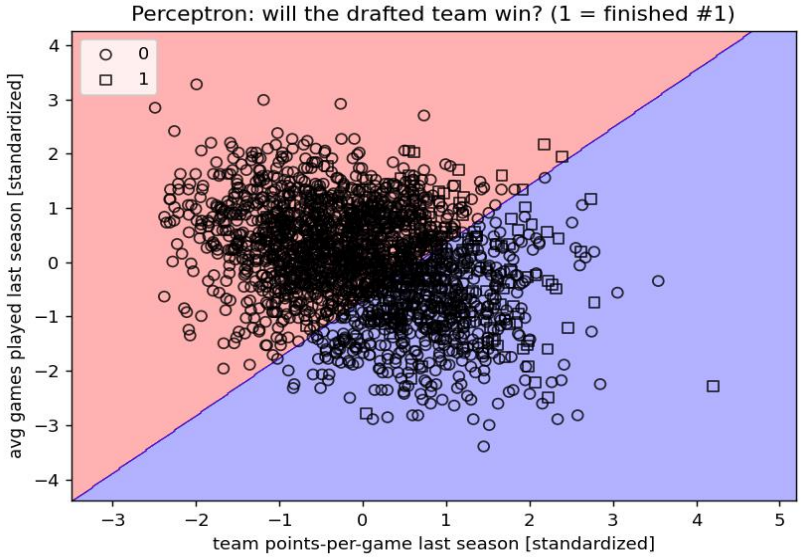
Talent: sum of the 10 players' points per game last season

Availability: average games they played. It separates teams with similar talent: mine 72, Kenny's 41

Label = 1: finished first, 10% of rows (1,200 of 12,000)

Simulated, not real past leagues. My real team: Yahoo roster with 2025-26 stats; Yahoo projections for 6 players with none

Perceptron (fantasy_prc_skl.py)



Blue = predicted WIN, red = predicted lose. Squares are teams that really finished #1.

```
Class labels: [0 1]
Labels counts in y: [10800  1200]
Labels counts in y_train: [7560  840]
Labels counts in y_test: [3240  360]
Misclassified samples: 1133
Accuracy: 0.685
```

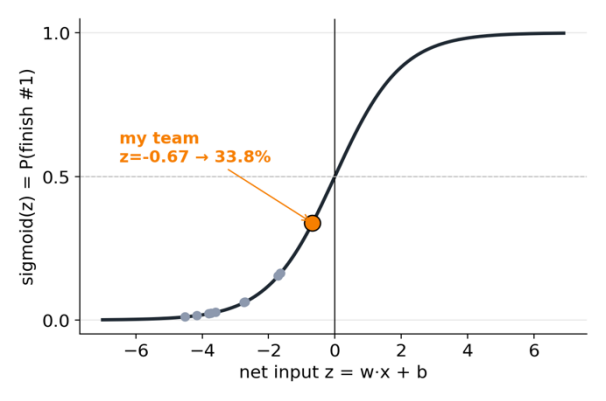
What it tells me

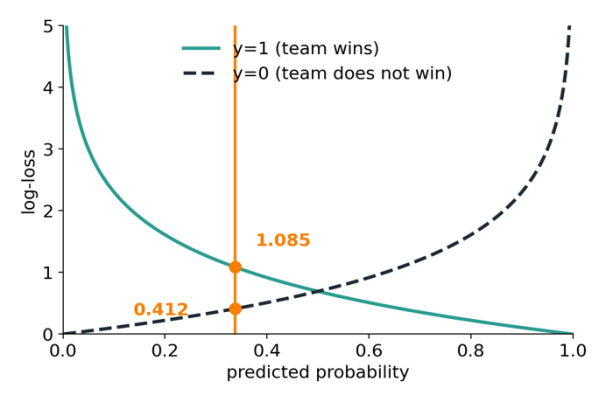
It labels 9 of my league's 10 teams WIN, including mine. A hard yes/no can't say how likely a win is, and the games-played weight is negative (−0.47), which makes no sense.

Sigmoid and log-loss

```
raw features:              team_fpg_sum=388.5  avg_games=72.1
standardized features:     [1.481 0.623]
net input  z = w.x + b = -0.673
sigmoid(z) = P(my team finishes #1) = 0.338
log-loss if my team DOES win  (y=1): -ln(p)   = 1.085
log-loss if my team does NOT win (y=0): -ln(1-p) = 0.412
```

Net input z	→	Sigmoid	→	Probability	→	Log-loss
−0.67		**σ(z)**		**33.8%**		**1.09 / 0.41**



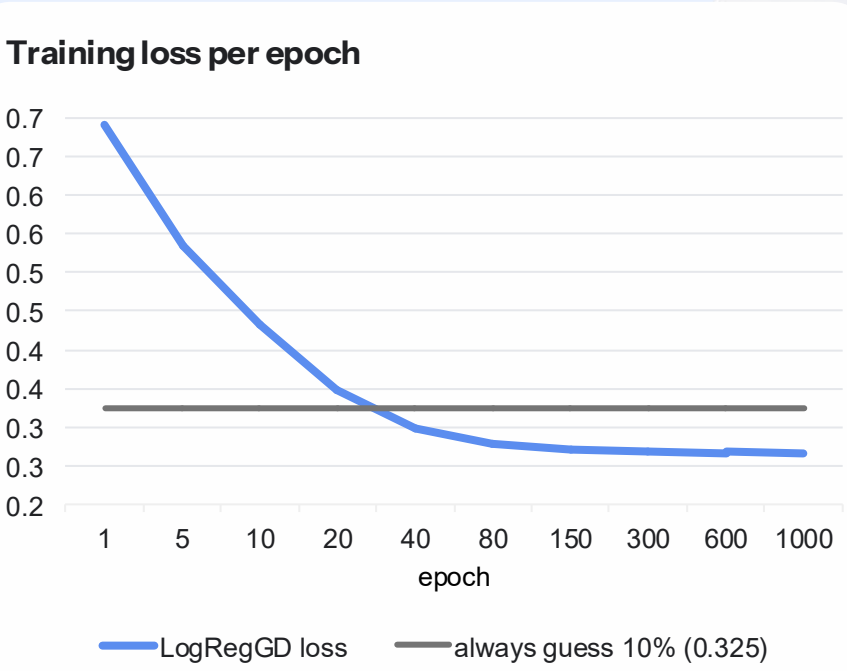


sigmoid.py: z = w·x + b is squeezed into 0-1. My team's z is −0.67, so 33.8%; the other 9 teams (gray dots) sit lower.
loss_function_sig.py: if my team wins, a 33.8% prediction costs −ln(0.338) = 1.09; if not, only 0.41. Confident wrong answers cost the most.

z = 1.277 × (talent, standardized) + 0.263 × (games, standardized) − 2.728

Manager	Talent	Avg games	Talent pts	Games pts	z	Chance of #1
Somony (me)	**388.5**	**72.1**	**+1.89**	**+0.16**	**-0.67**	**33.8%**
Kyle	386.7	62.8	+1.80	-0.71	-1.64	16.3%
Joaquin	378.1	66.5	+1.38	-0.36	-1.71	15.3%
Chris	356.0	67.4	+0.30	-0.28	-2.71	6.3%
George	365.4	62.2	+0.77	-0.77	-2.73	6.1%
Oliver	361.2	55.3	+0.56	-1.42	-3.59	2.7%
Brandon	335.7	67.1	-0.69	-0.31	-3.73	2.4%
Kenny	384.6	41.0	+1.70	-2.77	-3.79	2.2%
Sebastian	334.1	63.4	-0.77	-0.66	-4.15	1.5%
Jaelene	343.5	54.7	-0.31	-1.48	-4.51	1.1%

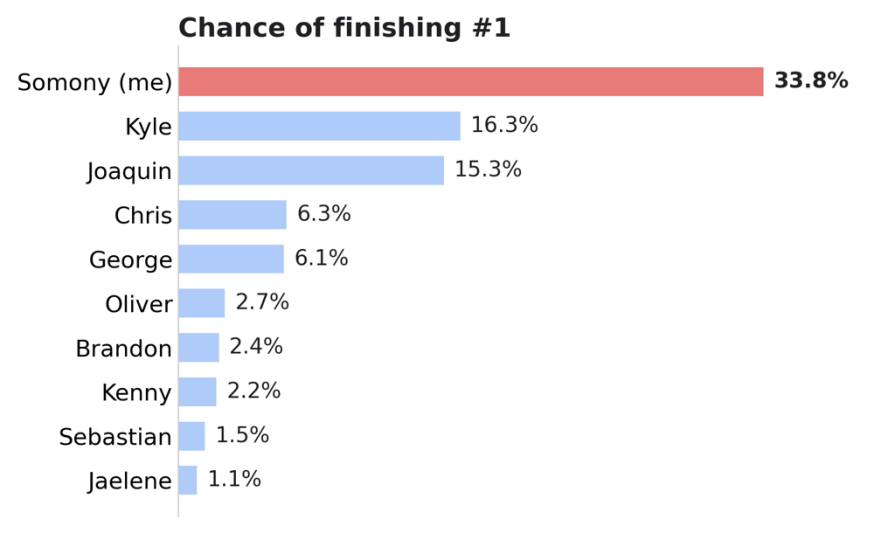
LogRegGD.py learns the weights




```
Class labels: [0 1]
train counts: [7560  840]  test counts: [3240  360]
LogRegGD  weights: [1.277 0.263] bias: -2.728
LogRegGD  test accuracy: 0.898   (always-"no" baseline 0.900)  teams predicted to win: 53
mean predicted P(win) in test: 0.099   actual win rate: 0.100
  predicted 0.00-0.05: 1568 teams, actual win rate 0.022
  predicted 0.05-0.10:  875 teams, actual win rate 0.090
  predicted 0.10-0.20:  693 teams, actual win rate 0.136
  predicted 0.20-0.40:  360 teams, actual win rate 0.303
  predicted 0.40-1.00:  104 teams, actual win rate 0.423
sklearn LogisticRegression(C=100) weights: [[1.277 0.264]] bias -2.728  (compare with LogRegGD)
```

$$z = 1.277 \times (\text{talent, standardized}) + 0.263 \times (\text{games, standardized}) - 2.728$$

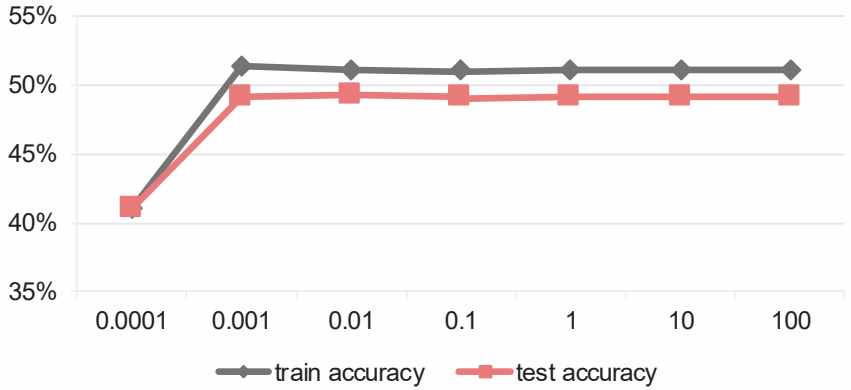
Result: my team has the best chance



My team is the favorite but is not likely to win: about a 66% chance it does not finish first.

More outcomes, and the role of C

3 classes (LogReg_Iris_3cls pattern): accuracy vs C




```
Accuracy: 0.449   (always-middle baseline 0.400)
Balanced accuracy: 0.346   (guessing = 0.250)
Confusion Matrix (rows actual, cols predicted):
[[  56   74  216   14]
 [  45   63  552   60]
 [  36   66 1052  286]
 [   9   10  616  445]]
Mean predicted P(champion) 0.099 vs actual champion rate 0.100
Share of test leagues where the team with the highest P(champion) really won: 0.353
```

Tiny C underfits (41%). From C = 0.01 up, accuracy is steady (train 51%, test 49%), so no overfitting.

My team, four outcomes
champion 34% · contender 35%
middle 26% · bottom 5%
predict_proba rows add to 100%.

What I added, and what it means

Beyond the class

- Built my own dataset: last season's real stats for every player, 12,000 simulated teams with known results
- Learned that accuracy can mislead when 1 in 10 win

Machine intelligence and people

- The model sees stats, not injuries, trades or my decisions
- It supports a manager; it doesn't replace one
- A 34% answer shows how much risk I carry, and that availability matters

For my goals (product management)

- Product decisions are bets: the question is how likely, not yes or no
- The same method estimates the chance a feature or launch succeeds
- Pair a model's probability with human judgment

A model that says 34% and is right about 34% of the time is more useful than one that says “win” and is wrong half the time.

Thank you

Questions?

Somony Pou