**Final Project Update: Stanford Football Betting as a Statistic and Metric Based Markov Decision Problem**

* Reintroduce problem statement
* The first step we took was to acquire relevant data from collegefootballdata.com on Stanford’s 2023 football season. There’s a lot to choose from, so we had to sift through a lot to find what we’d like to use. We ultimately extracted four datasets: betting data, advanced metrics (offense and defense PPA), general game stats (who won, Elos) , and team statistics (TDs, yds, etc). We want to utilize this data to set our states and inform the decision-making policy. Since these four datasets all had data we’d like to explore, a lot of time has been put into reformatting these datasets into one data matrix for every single statistic, metric, and betting line for each week, resulting in an 11x87 matrix, with a row for each week with data. Stanford played 12 games, so one week was omitted due to lack of betting data
* We then had to formulate the problem as an MDP
  + The states are the relevant statistics, metrics, and betting lines for a given week. We want our model to be used as if it was in the present day, so the algorithm has access to previous weeks as it iterates through the schedule, but not the weeks ahead, even though we know the results. We used a previous season so we can calculate the total profit one would gain by using the model and comparing it to baselines
  + The actions for now are “bet Stanford win” and “no bet” for simplicity. We assume the bettor is a loyal Tree and will not bet Stanford to lose. We ultimately want to explore taking other bets, such as covering the spread or betting on the opponent, but we want to first prove the model works to an extent.
  + The transition model follows the progression of the season. After each week, the states update.
  + The reward function is based on the betting outcome of each week. For now, a bet action equates to a $100 moneyline bet. If the action is bet and Stanford wins, the reward depends on the odds for the game. If the action is bet and Stanford loses, the reward is -100. Reward is 0 if the action is no bet. If we’re feeling fancy, the below code can be put into equation form for *R(s, a)*. Reward function is likely set in stone

A screenshot of a computer code

Description automatically generated

* The meat of the project will be the policy. Since the objective is to maximize cumulative rewards, we hope to use the metrics and statistics available to us to decide on the policy. Below is a graph using a random policy on the data. It’s obviously not good.

**A graph with green line

Description automatically generated**

* My first instinct is to compare the betting odds to the difference of the Elos, and if the risk is worth it, take the bet. We can also integrate other metrics like offensive/defensive PPAs, explosiveness, play success rate, etc. but that is an area of exploration. I’d say we go with a reinforcement learning based policy like Q-learning or Sarsa, but we are open to any suggestions. Honing in on the policy will be our focus going forward.
* We plan for our deliverables to be Profit vs Time plots. Initially, we’ll compare whatever policy(s) with a random policy to see how profitable we are. Since there are only 11 times a bet can be made, we might run many random policies and take an average to set a reliable baseline. Ideally we consistently beat a random policy for the 2023 Stanford season. To further test the robustness of our algorithm, we’ll attempt to extend it to other teams, but that will be a stretch goal, since the more data would have to be acquired, evaluated, and reformatted accordingly, which isn’t really the focus of this class. We might also compare how our algorithm fares for different betting websites, which would have different lines (bar graphs).
* Some things I’d want TA feedback on. Feel free to add anything
  + Do we have enough data if it’s only 11 weeks’ worth and a sequential problem? For example, if we’re betting on Week 2, we only have the outcome of Week 1 to build on. Would it benefit us to go back in time to a bunch of other Stanford seasons and use that data to train the algorithm or something?
  + Would certain methods work best here?