WeHateFantasyFootball

Christian Gerber (cgerber3@illinois.edu, Captain), Ido Tamir (itamir2@illinois.edu)

Project Description

Overview

We are both fantasy football players, and one thing we know about fantasy football is that it is an emotional game. Fantasy football team managers get upset when their fantasy players aren't doing well and are often overly optimistic when their players are doing well. We want to analyze how sentiment about football players correlates to their actual performance in the game. Are people's feelings for players valid? We want to find out.

If you are unfamiliar with fantasy football, fantasy football is a game where you pick real football players and put them on a 'Fantasy' team. This team could be your dream team, of your favorite players all from different teams, or it could just be a team that you think is good. Then each week, you face against another fantasy football team. Points are determined by how well your player plays. If your player runs a yard down the field, you get 0.1 points, if they score a touchdown, they get 6 points, if they fumble the ball, they get -2 points. A lot of it is based on luck, you never know how a game is going to go, but statistics and projections are a big part of it.

Why is it important or interesting?

This is interesting because of the growing game of Fantasy Football. A quick Google search says that over 40 million people in the USA only play fantasy football, and it grows year by year. We believe that this analysis will provide useful feedback, and levelheadedness when choosing who to play, trade, or sit on your fantasy football team.

What tools, systems or datasets are involved?

We will be using an ESPN Fantasy Football API to get player data and projections. This will provide the dataset for the players.

The dataset for sentiment will come from reddit NFL posts and fantasy football posts and other forums we deem useful for this project.

For the actual sentiment analysis model, we will find a pre-existing one online so that we don't have to find training data for it and can just immediately plug it in to evaluate the documents.

What is the expected outcome?

We will evaluate fantasy players sentiment over NFL players as compared to the projected fantasy points of the player. We hypothesize that projected points will be a better predictor of future player performance than fantasy player sentiment. Team manager sentiment is a messy function that has many factors unrelated to overall player performance such as NFL player's character, political positions, and

single performances as well as just the character of the team manager themselves. We hypothesize that projected statistics will be much more accurate and alongside with team manager sentiment will allow us to find players that are undervalued that we can trade for.

How are you going to evaluate?

To evaluate the team manager sentiment, we will use a pre-built sentiment analysis model and use it to analyze fantasy related posts from reddit and other forums. We will then find the players that the posts are about by checking for names of real players in the posts. Once we have the sentiment analysis of individual players, we can evaluate their projected points and then their actual performance. Once we have all those metrics, we can regress on these variables and see which factor is a better predictor for the actual performance of a player.

What programming language do you plan to use?

For both the sentiment analysis, the web-crawler, and the script to interface with the ESPN-API we plan on using python. It is the language we are most comfortable with and there are plenty of tools available for it to achieve what we want.

Workload Justification

- Figure out how to interface and fetch data from ESPN API (3 hours or more hours)
- Find and utilize sentiment analysis model (5 or more hours)
- Build web-crawler (15 hours)
- Build models to evaluate sentiment analysis vs projected points (10 hours)
- Working on best way to display results (5 hours)
- Final presentation (3 hours)
- Optional if we have time: Build web interface for results (10 hours)
- Debugging (5 hours)

Total: 41 hours + 15 hours for debugging and optional work