PES Dispenser: Using a neural network to create the best possible soccer team

Richard Cheng[1]

[1]Clayton High School, 1 Mark Twain Cir, Clayton, MO 63105

Motivation

I started working on this project because I'm really invested in a mobile soccer game called PES (Pro Evolution Soccer). It's basically analogous to FIFA, but the engine behind the game reflects real-life soccer gameplay much better. I follow European football very closely, so I was really motivated to use machine learning and data science to try to come up with the best starting 11 from all 12,782 players in the game's database. My principal purpose was to make the best team in order to beat my friends and to climb the rankings in the game. I also wanted to see if I could tackle pretty complex data problems with scientific Python.

Considerations

There were some factors that had to be considered before implementation of my algorithm. The first, and perhaps most important, was the data I would use for my machine to learn from. I decided that since I have so much faith in this game's similarity to real soccer, I could use FootballRanking's ELO scores for all of the real soccer clubs in the world. This way, my machine would be able to see what works in real-life and consider the virtual teams in the same lens.

Another factor that had to be considered was the type of regression I would use for my machine. I landed on neural network regression after some trial-and-error.

One more important consideration was how to evaluate the players in the game. PES already has a database for all of the statistics of their players (accessed on pesdb.net), and I decided that the best metrics to be considered were every attribute except for "overall rating" and any goalkeeping skills. (These attributes include "dribbling," "physical contact," and "defensive prowess.") Very early on, I reasoned that since goalkeepers rarely contribute towards outfield gameplay and don't really interact with the other players, I would use my machine to find the best 10 outfield players and just use the highest rated goalkeeper.

Methods

Here's a link to my github where the project is: `https://github.com/rcheng01/pro_evolution_soccer_data_analysis`.

The first thing I did was build a web crawler with BeautifulSoup to gather all of the URL's of players on the PES database. Since each player was on a different URL, I created a database using my crawler so that I could access each player's page efficiently.

Next, I used that previous database to parse each player's individual pages for their 18 in-game attributes.

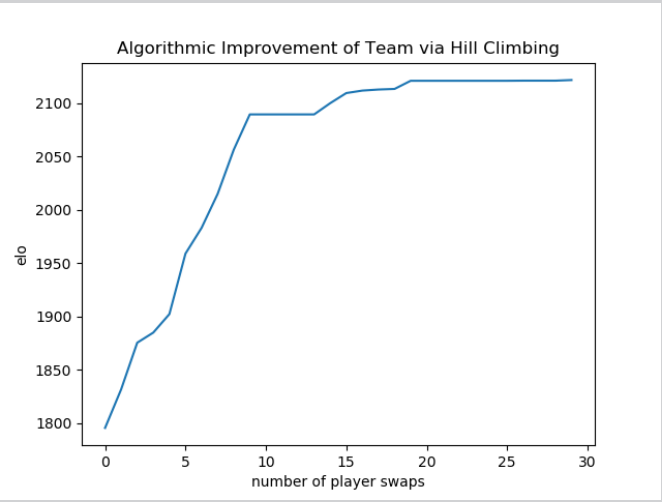
Upon completing this, I converted this into a large pandas dataframe with every player in the game and their stats.

Next, I hardcoded the starting 11 lineups for the top 50 teams in the world according to FootballRanking's world club website. After importing that into my code, I joined this data with my existing dataframe by adding a column linking players in the dataframe to the team that they belong to.

Methods (ctd.)

Next, I used the "sklearn" library to import the neural network regression. I trained my machine using the data from the top 50 teams in the world and their associated ELOs. One consideration I accounted for was the order of the players – since I didn't want this to be a factor, I permuted each of the list of ten players for each team 100 times to enforce symmetry.

Once I trained my machine, I built a hill climbing algorithm to test sets of ten players. Each iteration, I swapped a player in one slot with each of the other players in the entire dataframe. I kept whichever player produced the highest ELO in that spot, kept them there, and moved on to the next slot, doing the same. I did this with each of the 10 outfield positions, and then I repeated this hill climbing for 30 loops. The results of the hill climbing algorithm are depicted below.



Results and Discussion

The seed that produced the highest ELO was the following: [Luka Modric, N'Golo Kante, Thiago, Lionel Messi, Neymar, Paulinho, Marcelo, Antoine Griezmann, David Silva, Jordi Alba]. This was surprising given the clear emphasis on attacking players. I suspect that from the top 50 teams I gave my machine, I didn't submit enough bad performance teams with poor defenses, so it had no reason to penalize the lack of defenders. Still, the team composition is pretty decent, and to be honest, this fits with the high-attacking metagame that's present with the game's professionals. I'm also pretty pleased that the algorithm was able to pick up on "underrated" players like Kante, Thiago, and Paulinho who aren't soccer superstars but are instrumental towards their teams' successes.

I'm going to continue to improve this project, and I think the most straightforward thing I can do is to put more existing teams into my code to improve my machine. Also, I intend to try other forms of regression and optimize from there.