**Project Goal:**

I want to use my knowledge of Python, statistics, and predictive modeling to predict which Fantasy Premier League players will earn the most FPL points in their upcoming matches. To do this, I will use available data on Premier League fixtures and players, as well as examples of other machine learning methods used to predict match results and player points.

**Measuring Success:**

I can measure the success of my project by comparing the post-prediction team and avg game week points to my previous FPL season teams and their avg game week points.

I can also measure my success by checking my position in the FPL table relative to all others. If I am > 50%, then statistically my team outperforms the “average team”.

**Game Rules:**

* Each fantasy ‘manager’ starts with a budget of 100 Million and must buy 15 soccer players (11 for the main squad + 4 subs)
* You must have at least 2 Goalkeepers, 5 Defenders, 5 Midfielders, and 3 Forwards
* You can’t have more than 3 players from the same team

**Step 1 – Finding the Right Data Source:**

I have a couple options to use as my source of premier league data.

I can connect directly to the FPL API with the following link: ‘<https://fantasy.premierleague.com/api/bootstrap-static/>’. This is great because the data is constantly updated, but also requires me to reformat the data (check my Jupyter notebook for a previous example. This article explains how to set up a proper data frame when using the API: ‘<https://towardsdatascience.com/fantasy-premier-league-value-analysis-python-tutorial-using-the-fpl-api-8031edfe9910#8ef2>’

The above dataset is available in JSON format.

I could also use this previous code that is *“A FPL library that gets all the basic stats for each player, gw-specific data for each player and season history of each player”. The code can be found at: ‘*[*https://github.com/vaastav/Fantasy-Premier-League*](https://github.com/vaastav/Fantasy-Premier-League)*’.*

*The above dataset is available in csv format.*

**Step 2 – Learning from Others:**

I’m not sure where I saw this from, but in a previous example I saw that someone was trying to accommodate for players being signed and transferred to the Premier League partway through the season. In that example, Avg Points per game was calculated by taking a player’s total points and dividing by the # of games that they played in (minutes played > 0).

Another example solved this exact same problem by just removing players who played in less than 10 games.

‘<https://github.com/alsgregory/Fantasy-Football>’

“*The team model is trained initially on the previous three seasons of the Premier league (2016/2017, 2017/2018 and 2018/2019). The player model is trained initially on the previous two seasons of the Premier League (2017/2018 and 2018/2019). In both models, each season is time-weighted with respect to how recent it was.*”

The weighting in the above example is done by the following: 2 years ago season is weighted 1/3, last year’s season is weighted 2/3, this year is weighted 3/3 or 1.

‘<https://towardsdatascience.com/a-simple-method-to-predict-player-performance-using-fantasy-football-data-8b2d3adb3a1a>’ & ‘<https://github.com/d-t/fantacalcio>’

The above example looked at Serie A data and only used the following parameters:

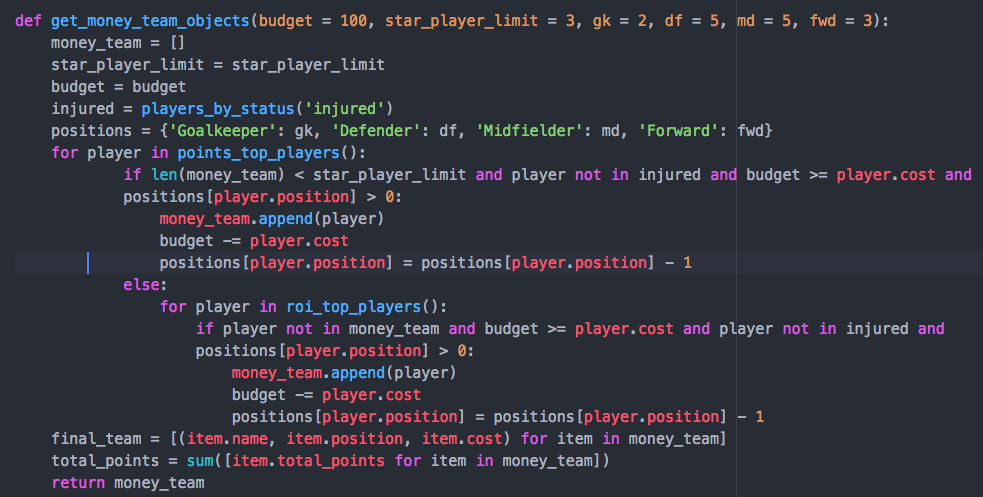
* Score difference between the 1st & last season
* Career score (calculated by taking the total career points divided by the total career games played)
* Presence of a constant positive or negative trend
* # of years playing in the Serie A
* Last season’s avg score

‘<https://towardsdatascience.com/beating-the-fantasy-premier-league-game-with-python-and-data-science-cf62961281be>’ & ‘<https://github.com/Botafogo1894/Project1>’

“*Most optimal Fantasy squad will be measured in terms of the total amount of Fantasy points returned per Fantasy dollars spent = ROI.*”

The above example looked at the following conditions:

* Don’t pick injured, suspended, or unavailable players
* Pick the top three star players with the most cumulative league points in the league first. (We will test the outcome of this condition with different numbers of star-players and pick the version that generates the biggest return on investment, while still allowing enough left-over budget to fill our squad with a lot of the Top 50 ROI players)
* Every time we pick a player and add them to our team, we subtract their cost from our 100MM budget and we add their position and team-name to a list
* Once the optimal number of expensive superstar players are picked, the Algorithm starts going through the list of players with the highest ROI and tries to get us as many of the top names as it can, until we get close to depleting our budget and filling all of the team positions.
* Algorithm prints a list of the players it picked at the end and gives us the remaining budget and the total fantasy points of the team.

Example Python script: 

**Important Notes:**

‘<http://goodsciencebadscience.nl/?p=424>’

Stepwise Regression should be used during the exploratory stage. We want to use the backward method of stepwise regression to help understand what variables have the most importance in predicting future scores, but we don’t want to use stepwise as a predictor because of its limitations. Limitations include overfitting the training data and the importance of the order in which the variables were added to or removed from the model. The *Forced Entry Method* is often used as a substitute for stepwise regression in confirmatory research.

**Program of Choice:**

For this project I’m going to be running Python on a Jupyter Notebook. I’m choosing to use Jupyter Notebook because I want to focus on readability, both for myself and others as this is going to be a long-term project and one that I hope to improve upon in multiple future iterations.