Fantasy Premier League Project

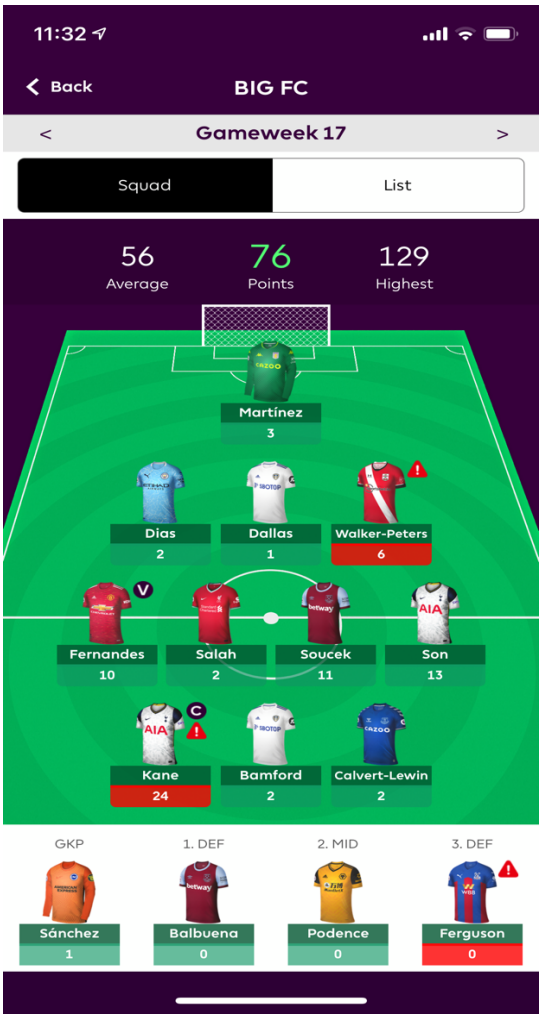
After years of playing the Official Fantasy Premier League with my friends and always relying on intuition and football "Knowledge" when picking my teams, transfers, starting lineup, etc. I decided to start applying things that I have learned throughout my academic journey to my playing strategy in order to boost my performances and improve my ranking over the multiple leagues I'm enrolled in. For those who do not know what Fantasy Premier League is, it is a platform where people create a 15-player team of Premier League players, and gather points based on the real-life performances of these players. Check the image and explanation below for a short example.

The following image is a clear representation of a Fantasy Premier League game week, where I have picked a starting 11, the ones who actually earn me points, and 4 bench players who are only used in the case that a starting player does not play in the game week. A player's points are determined by his real-life performance during the game week:

- Goal: 4 points for forwards, 5 points for midfielders, 6 points for defenders/goalkeepers.
- Assist: 3 points for all players.

- Clean sheets (not conceding a goal): 4 points for goalkeepers and defenders, 1 point for midfielders.

The scoring system is much more detailed and has many other variables, but these are enough to understand how the game works.



There are 38 total game weeks, and my total score is the cumulative sum of each game week's points (76 in this example). Moreover, players are included in public leagues, like the worldwide league (where I currently rank 603,726 out of 7 million) and the Lebanon's local league (where I currently rank 1836 out of 24,000), as well as private leagues that people create with their friends. Building a team is not that easy, since there is a limited budget (100 pounds) that should be spent wisely in order to have the best possible combination of

players, and each user gets to have 1 free transfer each game week, and any additional transfer deducts 4 points from the player's total score. However, each player has 2 wildcards per season, where they could change their whole team without any cost.

After having learned Data Science and analytics for the past year and have started to take interest in the field of sports analytics, I thought it would be interesting to apply those concepts in Fantasy Premier League and try to look for meaningful insights that can potentially boost my performance. My research started by extracting live and updated data from the Fantasy Game API to run statistical analysis on players and teams, in order to finally build 3 Python algorithms that create the best possible team, transfer out the most overvalued player from the team, and transfer in the best players that are not currently in the team.

The main concepts behind the algorithms I built are the following:

- Instead of relying on my personal biases and intuitions, picking players for my next wildcard use will rely on metrics like ROI (return on investment = total points/cost) and not only total points, in order to pick the highest valuable players per pound spent.
- A combination of a player's form and difficulty of upcoming fixtures will determine which players should be transferred out of the team and which players should be transferred into the team.

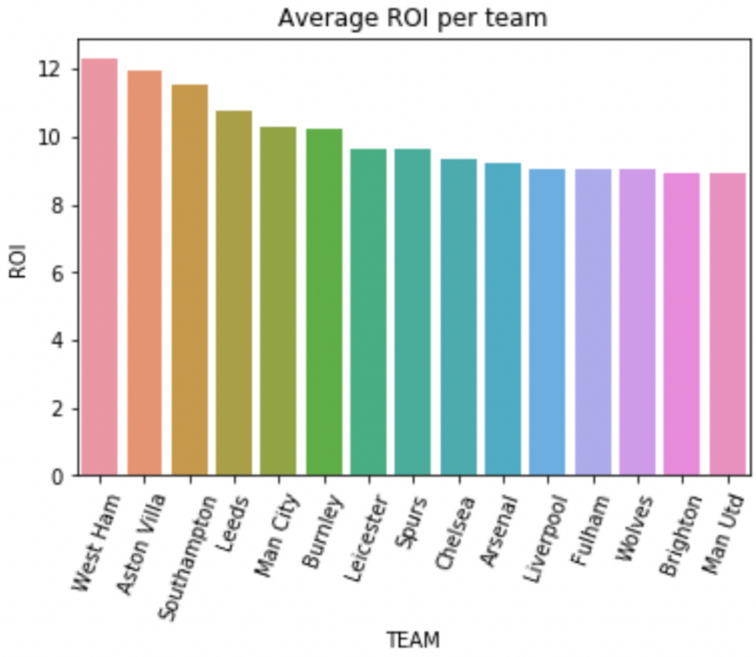
Note: The whole code will be available on my GitHub page, which I will include a link for in the report's conclusion.

1) Importing the Data

Importing the data was an easy step, since the game's website already had a predefined official API that made it easy to import it using a couple of python lines. The data was cleaned, sorted, and filtered, becoming ready to be used for further analysis.

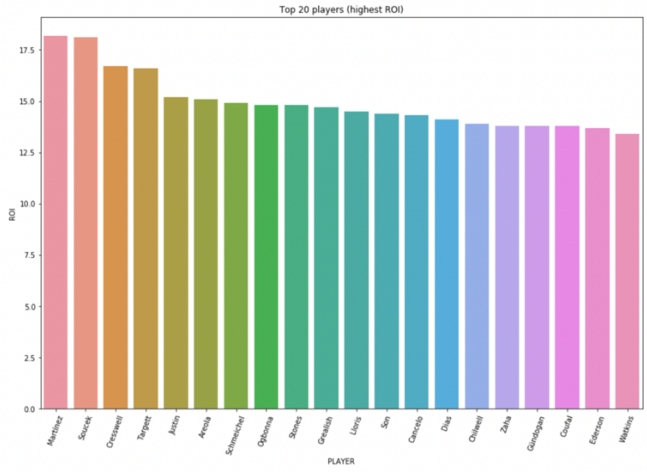
2) Exploratory Analysis

In order to properly understand how to approach the problem, I checked the average ROI per team, since I wanted to build an algorithm that picks the players from teams with the highest average ROI.



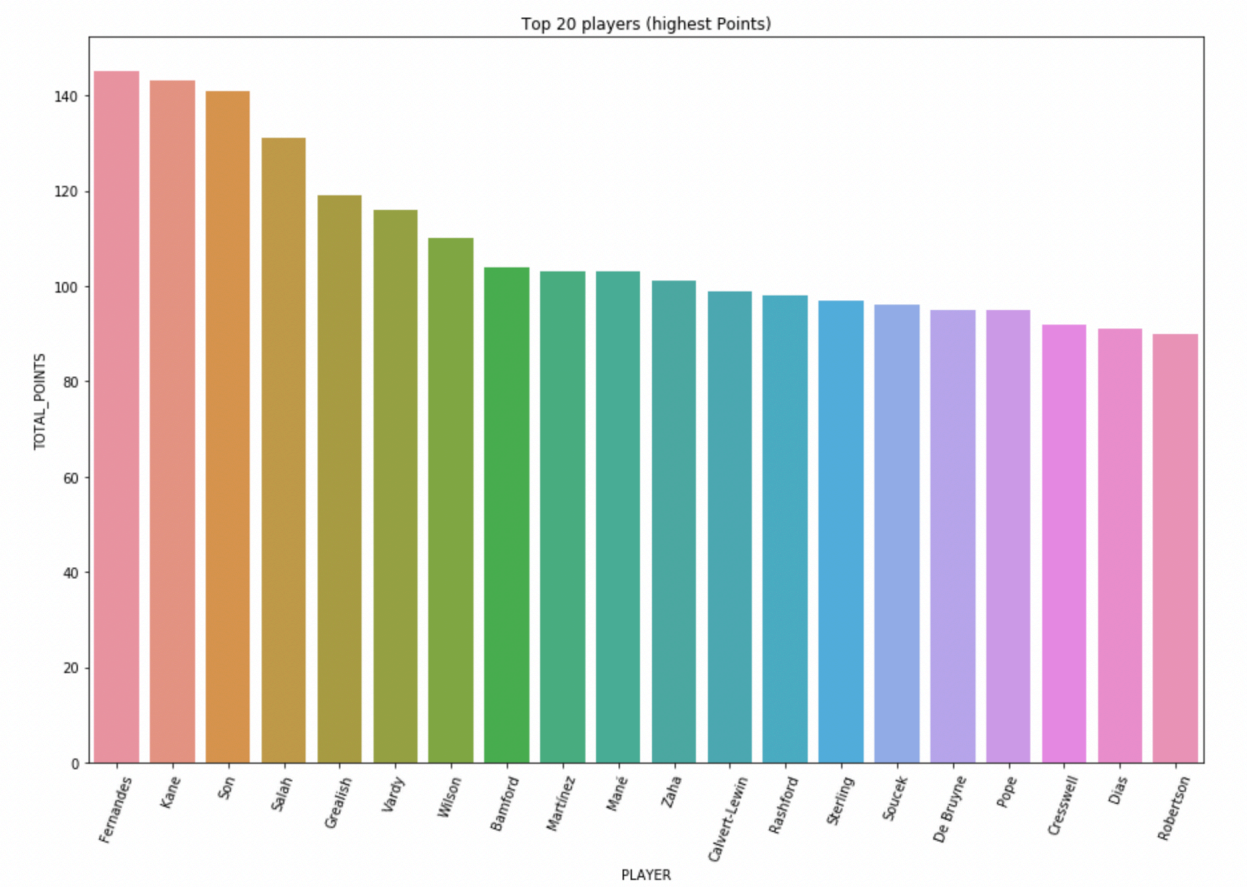
I found it very interesting that from the top 5 teams in terms of average ROI, only 1 team (Man City) is from the top 4 in the Premier League standings, clearly indicating that the other 3 teams (Man Utd, Leicester, Liverpool) have a few numbers of players that are performing well and are expensive, while the rest of the players are underperforming/overvalued. For example, Man Utd is currently 2nd in the league but 15th in average ROI, since most of their points come from Marcus Rashford, Bruno Fernandes and players that are high in cost (above 8 pounds) with the remaining of the players barely getting a reasonable number of points. On the other hand, West ham's players are relatively very cheap and are hugely overperforming (4th in the league currently), leading to this very high average ROI. This is the basic logic behind our team selection, picking more players from teams like West Ham and Aston Villa, and fewer but more expensive players from teams like Liverpool and Man Utd, since they guarantee a high number of points despite a low ROI.

Next, I decided to identify the players with the highest ROI up until this season, making them the most valuable players in the league that our algorithm should keep an eye on.



The best way to understand this bar graph is to take Martinez's case, where his cost at the start of the season was at a very low 4.5 pounds, since the game's creators as well as players never expected a lot from him and his team, considering Aston Villa were one of the worst performing teams last season. To everybody's surprise, Aston Villa are among the best performing teams this season, and Martinez is arguably the best performing goalkeeper, having a total of 103 points, hence the very high ROI value.

Now that ROI and how it works has become clearer, it is time to highlight why relying only on top ROI is not enough for a good team. Below is a look at the top performing players in terms of total points gathered:



Obviously, almost all of the players in the bar graph are not present in the graph of highest ROI, meaning that those players may not be the most valuable in terms of point returned per pound spent, but that is due to a high cost rather than low number of points returned, especially considering that almost all of these players are from the most popular teams, explaining the high cost.

The algorithm's job is to balance the number of players with a high ROI, and number of players with a high total points value, wanting player like Bruno Fernandes in the team to guarantee constant points, but also wanting players like Emiliano Martinez to have players of high value.

3) Building the Wildcard algorithm

Before starting with the algorithm, one should understand the rules and constraints of building a team:

- You start the season with 100 pounds that can go up or down depending on how the players' values increase/decrease based on their performances. For instance, after 18 game weeks my team is now worth 101.4 pounds, which is why I will make the algorithm's starting budget to be user defined.
- The team should consist of 15 players, 11 main players, and 4 subs.
- The players' positions distribution should be the following: 2 goalkeepers, 5 defenders, 5 midfielders, and 3 forwards.
- Only 3 players from the same team are allowed to be in the team.

After defining our constraints, it is time to define the thought process behind the algorithm:

- Avoid picking players with less than a total of 650 minutes, since we only want to select regulars, and players with no chance of playing next round, indicating a current injury.
- Our team should have a total of 3 'star players', which are defined as players with the highest number of total points.
- For every player added, his cost is deducted from the budget, his team and position are added to a list of teams selected till this point and a list of positions selected at this point.
- The algorithm starts picking the number of star players defined, and then goes to selecting players with the highest ROI value.
- Pick players with a form > 2.6, to avoid players on a bad run.

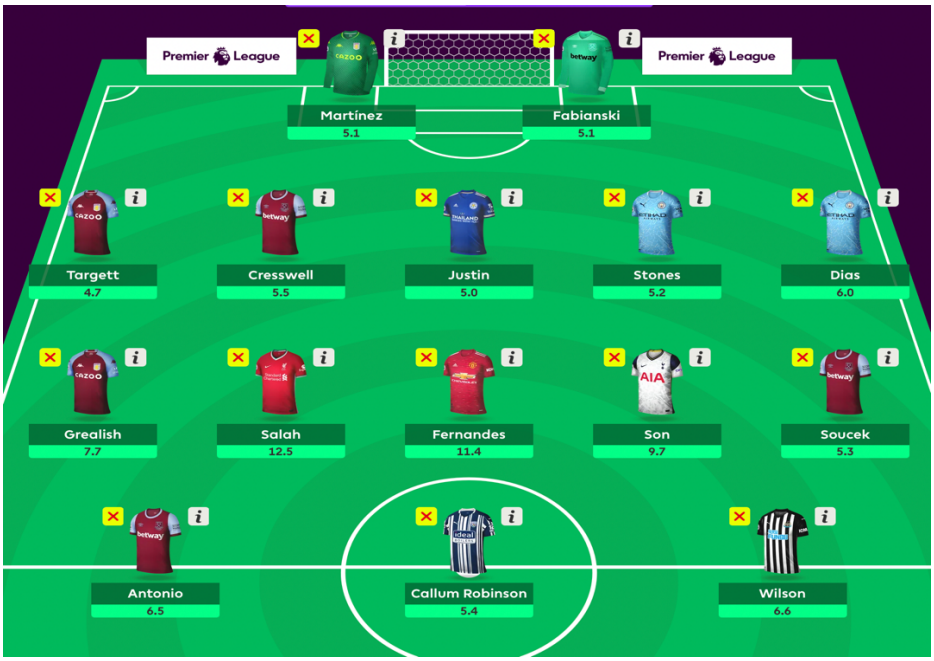
Below are 2 screenshots of my Python algorithm's code:

The Algorithm (Wildcard)

In [15]:
```
# Setting the conditions
gk = 2
mid = 5
fwd = 3
same_team = 3 # max of 3 (<= 3)
star_player = 3 # Total number of star players wanted in the team

my_team = [] # List of names of team created



star_players = top_players_points.sort_values(by = ['FORM'] , ascending = False)


def create_team(same_team = 3 , star_players = 3, gk = 2 ,df = 5,  mid = 5, fwd = 3):


    # Teams of players added to list
    teams = []

    # We need the budget to be user inputted

    budget = float(input('How much money is in your bank ? '))


    # Creating a dictionary that sets position limits

    positions = { 'GKP': gk , 'DEF': df , 'MID': mid, 'FWD': fwd }
```

```
# Creating a dictionary that sets position limits
positions = { 'GKP': gk , 'DEF': df , 'MID': mid, 'FWD': fwd }
for player in top_players_points.index:
    if len(my_team) < star_players and budget >= top_players_points.COST[player] and positions[top_players_point
        my_team.append(top_players_points.PLAYER[player])
        budget = budget - top_players_points.COST[player]
        positions[top_players_points.Position[player]] = positions[top_players_points.Position[player]] - 1
        teams.append(top_players_points.TEAM[player])

    else:
        for player in top_players_roi.index:
            if len(my_team) < 15 and budget >= top_players_roi.COST[player] and positions[top_players_roi.Positi
                my_team.append(top_players_roi.PLAYER[player])
                budget = budget - top_players_roi.COST[player]
                positions[top_players_roi.Position[player]] = positions[top_players_roi.Position[player]] - 1
                teams.append(top_players_roi.TEAM[player])


return my_team
```

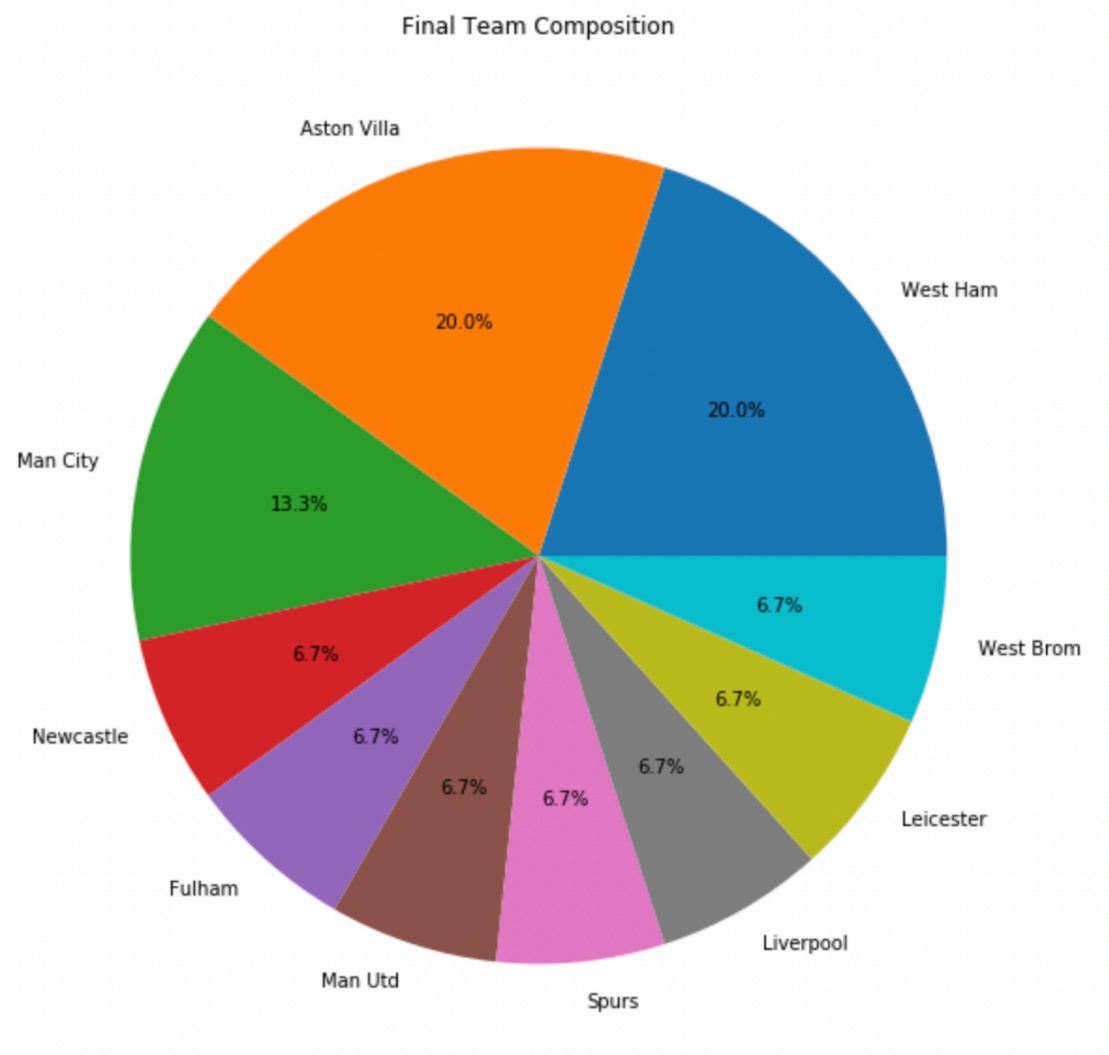
When I call out the function and insert my budget as 101.4, the algorithm gives me this team:



4) Model Performance:

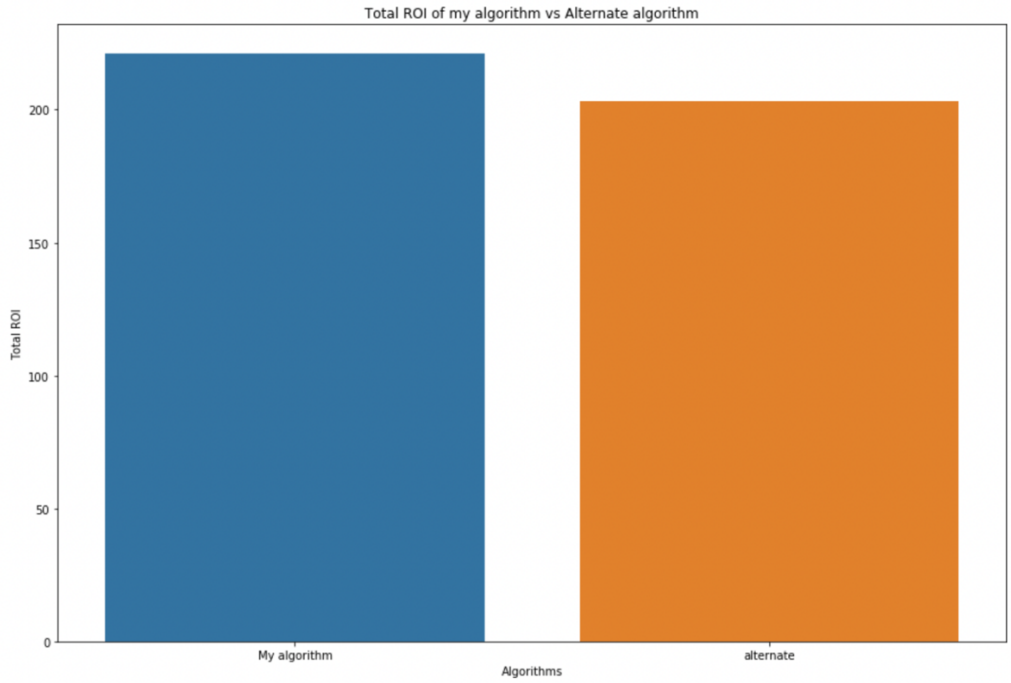
We can notice that our algorithm avoided injured players, like Harry Kane and Kevin De Bruyne, despite having a large number of total points, and picked a perfect mix of three star players and players with high ROI.

The team distribution of our algorithm's picked players is visualized in the following pie chart:



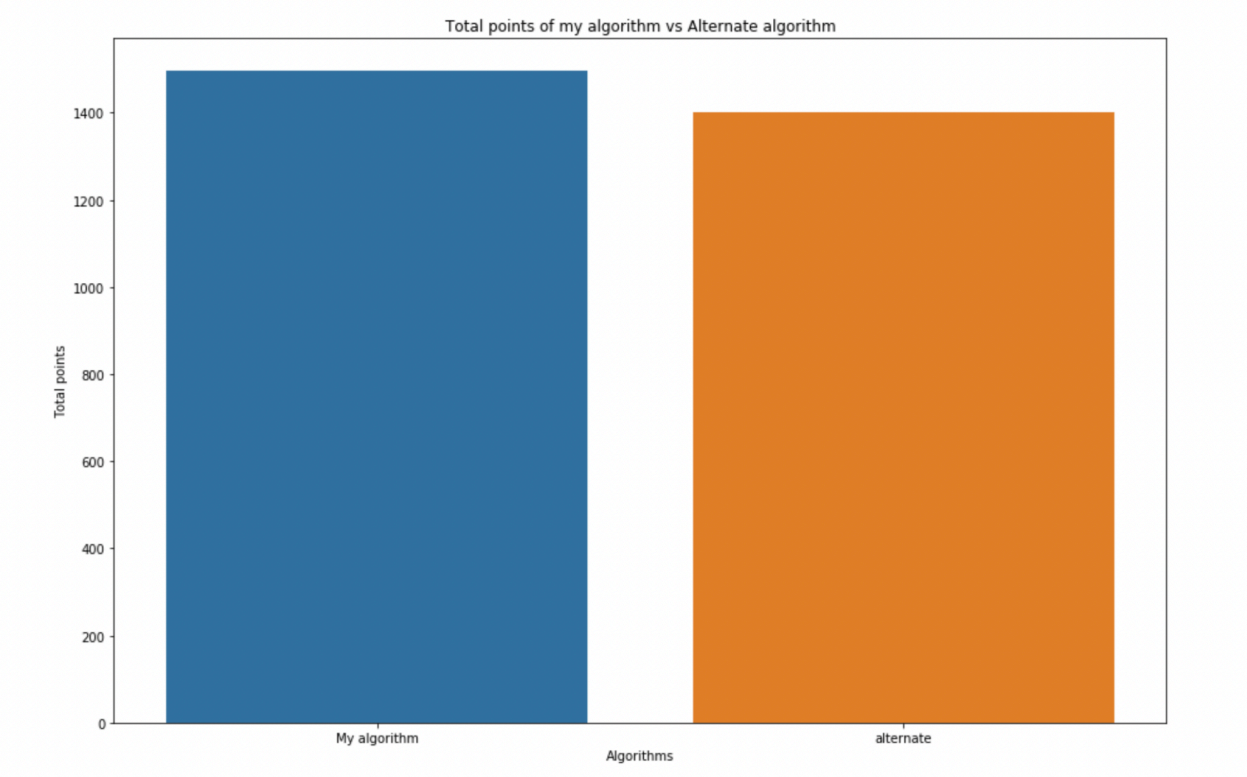
The top 3 teams our algorithm picked players from are West Ham United, Aston Villa, and Manchester City, which are 3 of the top 5 teams in terms of average ROI I discussed in section 2 of the report. Players from Leeds United were avoided despite having a high average ROI, but that makes sense considering the very bad form and results they are having.

In order to evaluate the quality my algorithm, I coded another algorithm that focuses only on total points returned (a more standard approach to the game) and disregards the return on investment and actual value each player has in order to compare the total points as well as total ROI of both algorithms.



In terms of total ROI, the algorithm I built outperformed the alternate algorithm, which is kind of expected since my model was built on the concept of ROI being the main metric. However, I should check my model's performance in terms of total

points gathered against the alternate model, since the alternate model focuses solely on points gathered which might give it an advantage.

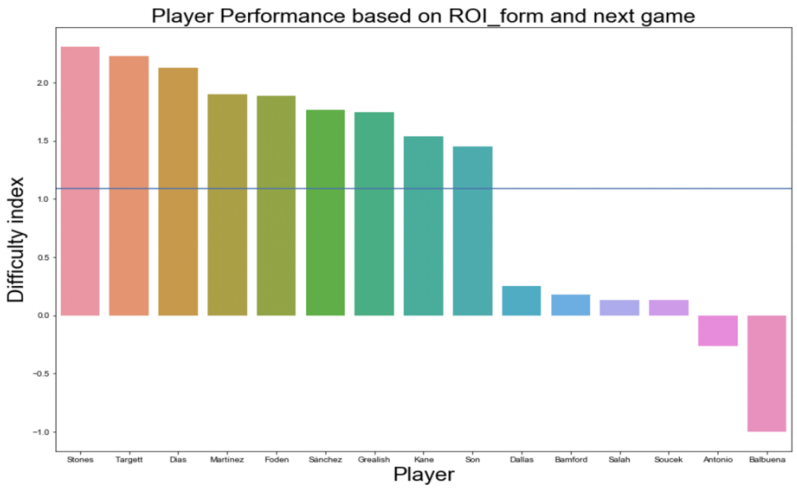


To my surprise, my model even outperformed the alternate model in terms of total points gathered, showing how effective it turned out to be in terms of building a perfect combination of star players and valuable players.

The next step was to create a transfer bot that advises me which players to transfer out and which players to transfer in.

5) Transfer out:

When transferring a player out, we should take several factors into consideration, like his most recent performances, the difficulty of his upcoming games, and his cost. Consequently, I decided to create a new variable similar to the ROI, but instead of dividing total points by cost, we divide form by cost, since form is the average number of points per game the player has accumulated during the last 30 days. This new metric will evaluate which players are currently the most valuable, irrespective of their form during the beginning of the season, in order to identify the current underperforming players which should be transferred. The next step was to account for the difficulty of upcoming games, which was defined as 1 one of the three values -1,0, or 1, all of which were all defined already in the Fantasy Premier League website. Finally, both values were added together to form the so called "ROI_DIFF", which combines current form, cost, as well as difficulty of the upcoming game. The function created prompts the user for their current 15 player team and returns which player should be transferred out as well as a bar graph showing which player has the lowest value corresponding to our new metric. The following is a small example of what the function returns:



```
According to his recent form and overall return on investment, as well as
his upcoming game you should transfer:    Balbuena
```

After identifying which player should be transferred out, I created an algorithm that would identify which player is best positioned to take his place.

6) Transfer in:

When it comes to choosing which players should be transferred in, form is not enough to identify the best possible picks, because there is always a chance that a player is currently scoring or assisting despite having poor performances. For this reason, we should look at the metrics expected goals and expected assists, two metrics that are relatively new and are being widely used in the world of football analytics. "Expected goals" is a metric which assesses every chance, essentially answering the question of whether a player should have scored from a certain opportunity. Put simply, it is a way of assigning a "quality" value (xG) to every attempt based on what we know about it. The higher the xG - with 1 being the maximum - the more likelihood of the opportunity being taken. If a chance is 0.5xG, it should be scored 50% of the time.

The factors taken into account when assessing the quality of a chance include:

- Distance from goal
- Angle of the shot
- Did the chance fall at the player's feet or was it a header?
- Was it a one on one?
- What was the assist like? (eg long ball, cross, through ball, pull-back)

- In what passage of play did it happen? (eg open play, direct free-kick, corner kick)
- Has the player just beaten an opponent?
- Is it a rebound?

Expected assists (xA) measures the likelihood that a given pass will become a goal assist. It considers several factors including the type of pass, pass endpoint and length of pass. Adding up a player or team's expected assists gives us an indication of how many assists a player of team should have had based on their build up and attacking play.

The algorithm I built to identify the best players to transfer in followed 3 main principles:

- A player with goals > expected goals is an indication of an overperforming player who is scoring chances that are hard to score (have a low scoring expectancy), indicating quality performances from the player.

- A player with assists > expected assists is an indication of an underperforming player who is getting assists due to the high conversion rate of the players receiving the passes, and not the quality of the passes themselves.

- Consequently, the best players are those with goals > expected goals and expected assists > assists, which is why I created a metric adding those to values (G - EG) + (EA – A).

Finally, I filtered out players with form less than 4 in order to guarantee the best performing players in the last month and sorted them by the new metric defined. The following 2 screenshots are the code of my algorithm that suggest which player to transfer in:

```
def transfer_in():
    bank = input('How much money do you have in the bank ?')
    player_out = input('Which player are you going to transfer out ?')
    team = (input('Enter your 15 player team ')).split(',  ')


    # Checking avaiable budget
    pout = player_out.upper()
    pout_cost = df[df['PLAYER'].str.upper() == pout].COST
    budget = float(bank) + float(pout_cost)

    # Create a variable that stops/continues iterations

    player_added = 0

    # Checking position of the player being transferred out

    position = df[df['PLAYER'].str.upper() == pout].Position

    # Transferring in the best player based on poistion

    if (position == 'FWD').bool() :

        for i in forwards.index:
            if player_added == 0:

                if ((forwards['PLAYER'][i] not in team) & (forwards['COST'][i] <= budget)) :
                    player_in = forwards['PLAYER'][i]
                    player_added = 1


    if (position == 'MID').bool() :

        for i in midfielders.index:

            if player_added == 0 :

                 if ((midfielders['PLAYER'][i] not in team) & (midfielders['COST'][i] <= budget)) :
                        player_in = midfielders['PLAYER'][i]
                        player_added = 1
```

```
    if (position == 'DEF').bool() :

        for i in defenders.index:

            if player_added == 0 :

                 if ((defenders['PLAYER'][i] not in team) & (defenders['COST'][i] <= budget)) :
                        player_in = defenders['PLAYER'][i]
                        player_added = 1


    if (position == 'GKP').bool() :

        for i in goalkeepers.index:
            if player_added == 0 :

                 if ((goalkeepers['PLAYER'][i] not in team) & (goalkeepers['COST'][i] <= budget)) :
                        player_in = goalkeepers['PLAYER'][i]
                        player_added = 1


    print (''),
    print(''),
    print('You should transfer in ' , player_in)
```

Conclusion:

All in all, the algorithm built for team selection was evaluated against another model inspired by the 'standard' approach to playing Fantasy Football and proved to be better, however the two other algorithms built for transferring players in and out cannot be evaluated currently, and the only way to see if they are in fact reliable models is for me to use them for the remainder of the season and see how my league rankings improve. In terms of limitations, I think a more accurate way to predict players' performances is building a machine learning classification model, however a major issue with this is the lack of data entries, since there are only 38 games per season per team and data from previous seasons is not that useful since most teams had very different teams, coaches, conditions, etc.

This project was a great introduction to the endless possibilities in the world of sports analytics, gave me a new way to look at my favorite sport, and definitely improved my skills as a data analyst and coder. The link to my GitHub account will below with the whole code, included the data scraping, cleaning, as well as all the algorithms.

GitHub link: https://github.com/Anthony-Moubarak/Fantasy-Premier-League-Project/blob/main/FPL_final.ipynb