1 Experiments

The goal is to find player swaps that could lead to the maximum points for each gameweek of the season.There are approximately 650 Premier League Players and close to 8 million FPL Managers playing the FPL.Given each team is of 15 players, there are (15*650) possible 1 player swaps in a given gameweek. This means if we were to start from a single team, there would be (15*650) possible teams in the 2nd gameweek. For a season consisting of 38 gameweek, we have $(15 * 650)^{38}$ possible teams/possible paths for just one FPL Manager starting from gameweek 1. Generating all $(15*650)^{38}$ possible teams and calculating the scores of these teams in each gameweek is computationally infeasible. Secondly, consider even the problem of finding the optimal team given a random team at start and you decide on making possible transfers to get to the optimal solution. The hypothetical dream team is the set of players where the cumulative sum of points per player per game week is maximum. But we could still have greedy mechanism of transfers right off the bat and still beat this dream team. Thus, this problem is not trivial. There in comes the need of finding the optimal team transfer strategy by considering all possible transfer trajectories. We relax this problem statement by approximations using a neural network. Thus, we have chosen the pipeline analysis as an experiment in Section 1.3.

1.1 Naive Heuristic

1.1.1 Implementation

For each FPL manager, get the points of the 15 players in the team in gameweek 1.Using all players available for swap in the current gameweek, swap the player that had the maximum points in the current gameweek with the team player that had the minimum points. After the swap we get a new team for the next gameweek.We now get the total points for the new team using the player points of this week. We store the points of this week and the cumulative points of the FPL manager so far. We then continue the swap process as before until week 11(Current Season has data for only 11 weeks).

Experiment 1 - Without any cost constraint In this experiment we assume that any player swap is possible. Hence, the steps are the same as above.

Experiment 2 - Randomization In this experiment, after calculating all the possible team points by all possible swaps, we apply a bit mask of 0's and 1's(generated randomly with equal probability of getting 0 and 1) to the team points such that each of the possible team points has a 50% chance of happening. That is, only a random subset of the swaps happen to get the team for the next gameweek.

Experiment 3 - Using the Cost Constraint

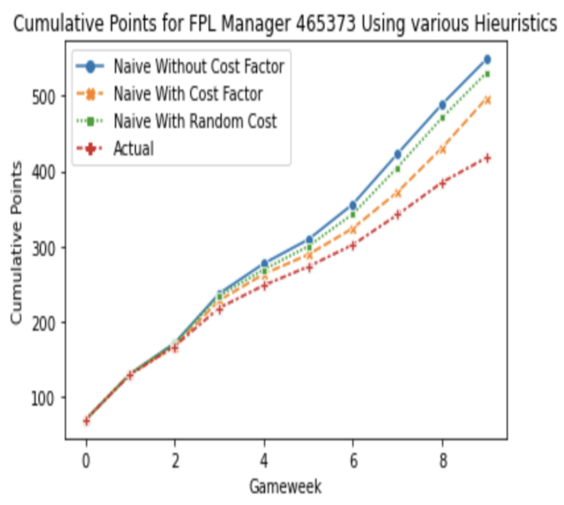
In this experiment, after calculating all the possible team points by all possible swaps, we apply a bit mask of 0's and 1's - formed by using the cost of the current team player to be swapped out, remaining budget of the FPL Manager and the cost of the player to be swapped in.

$$bitmask(i) = \{1,\ if currentBudget + cost(playerOut) >= cost(playerIn)\ otherwise\ 0\} \quad (1)$$

This invalidates the swaps that do that satisfy the budget constraints.

1.1.2 Results

The **naive heuristic** is to swap the player with minimum points in the given week with the best performing player in that week. We tried 3 different experiments(constraints) with the naïve heuristics.



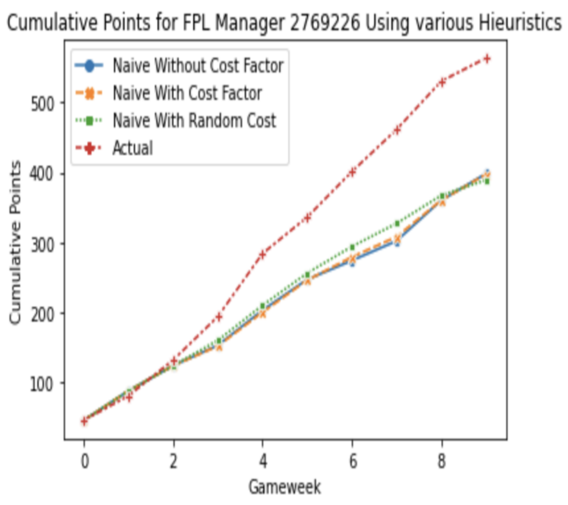


Figure 1: (Left) Cumulative Points of FPL Team where the heuristic generated team performed better than the actual team. (Right) Cumulative Points of FPL Team where the heuristic generated team performed worse than the actual team.

Experiment 1 - Without any Cost Constraint

The cumulative points of the FPL team generated using the naive heuristics is compared with the cumulative points of the actual FPL team chosen by the FPL manager on per gameweek basis.**This experiment gave better results than the actual FPL team for 172/474 teams(i.e. an accuracy of 36%.)**

Experiment 2 - Randomization

The cumulative points of the FPL team generated using the naive heuristics , with the possibility of each player swap being 50%, is compared with the cumulative points of the actual FPL team chosen by the FPL manager on per gameweek basis.**This experiment gave better results than the actual FPL team for 154/474 teams(i.e. Accuracy of 32%.)**

Experiment 3 - With Cost Constraints

The cumulative points of the FPL team generated using the naive heuristics , with the possibility of each player swap depending on FPL manager's budget, player to be swapped in and player to be swapped out, is compared with the cumulative points of the actual FPL team chosen by the FPL manager on per gameweek basis. **This experiment gave better results than the actual FPL team for 114/474 teams(i.e. Accuracy of 24%.)**

1.1.3 Observations

In terms of accuracy, the following is the relative ordering of the experiments :

Experiment 1 > Experiment 2 > Experiment 3

This is expected since players with higher points will be more expensive than the ones with lower points. Due to the cost constraint, it may not always be possible to swap in the player with highest points.Low accuracy in the above experiments suggest that a player who got the maximum points in the given week might not get the maximum points in the next week. Therefore, the naive heuristic may not give the optimal team at any game week or the end of the season. This calls for the need of a sophisticated heuristic/model that consider a lot of different factors apart from the performance in the current week.

1.2 Player Rankings

1.2.1 Rankings Implementation

Experiment 1 - Rankings Table based on points and ICT Index 1. Initially the player data is grouped by the element type (Goalkeeper, Defender, Midfielder and Forward) depending on the *element_type* values. The initial rankings table is based on the total points scored by the player. So the data set of the previous game week is sorted in descending order based on the points of the player. Then the dataset of the next gameweek is sorted in descending order based on points. This gives the actual ranking of the players in the next gameweek. To compare how accurate prediction with points turned out the number of common top 50 players is found. Next the the previous gameweek's dataset is sorted based on the points and ict index and the players are compared in the similar way as before by finding the number of common top 50 players.

Experiment 2 - Player comparison based on predicting next game week's points In this experiment we have used Linear Regression to predict the $total_points$ of the player in the next gameweek based on the features : *bps* , *ict_index*, *clean_sheets* , *goals_scored*, *goals_conceded*. The predicted points dataset is sorted in descending order based on the *total_points* and similarly the actual dataset of the particular week is sorted in descending order based on the *total_points*.The number of common fpl players from the first top 50 was found to compare the accuracy. The similar steps for both experiments are applied for all the available gameweeks data.

1.2.2 Observations from Player Rankings

To choose the next best player to swap we need to compare the players and rank them and for this purpose we have come up with rankings table where the players would be ranked depending on specific features.

Rankings using points
The initial rankings table is based on the total points scored by the player as the points scored by the player is the most important in determining value of a player.

Rankings using points and ICT Index
Then we have used the ICT index along with points to determine the rankings. ICT Index is a combination of the influence, creativity and threat features of the player and hence becomes a significant feature in predicting the player's performance.

Rankings using Predicted points
Linear regression is used to predict the next game week's points and the accuracy of finding the maximum number of top 50 players is shown in the plots. As can be seen from the plots the top 50 common players for Goalkeepers and Defenders have very close to actual results but that isn't the same case with Midfielder or Defender.

1.3 Pipeline

We identify the moving parts in the pipeline as shown in Figure 3. The objective is to identify the causal effects of decisions or strategies. In the real world, football team managers consider numerous parameters to buy in or sell players like team balance, fill in an open player type

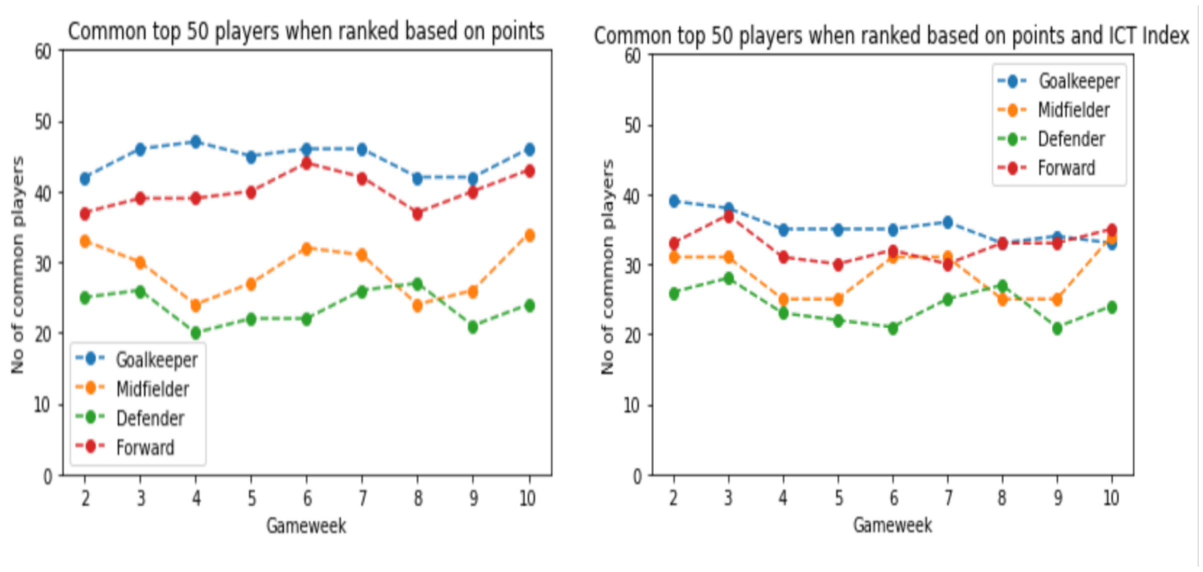


Figure 2: Common top 50 players when ranked based on points and ICT Index

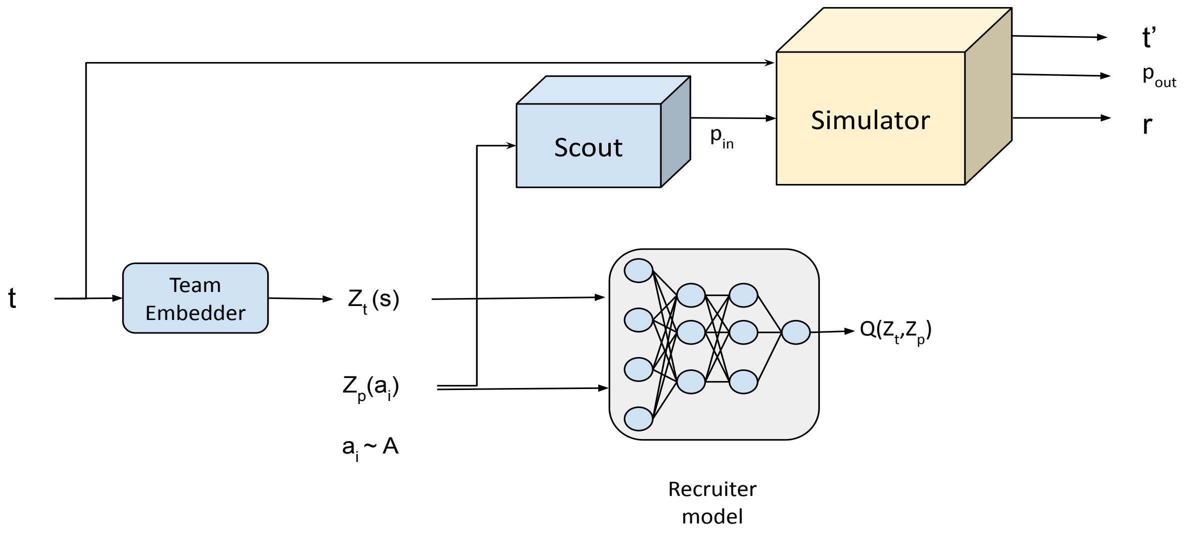


Figure 3: The pipeline for the learning module

position or player attributes. But the driving force is always to maximize gain league points and rank at the top of the leaderboards. The FPL managers are similar in this approach and look at actual gameplays to come up with decisions. Here we define an decision action space $a_i \sim A$. An action vector is a weighted probability distribution over player point features. Player point features are goals, assists, red card, yellow card, goals conceded etc ie; those which factor into a players total points. Thus, an weighted vector like [1.0,0,0,0,0] can be interpreted as a decision to focus on more goals or another like [0.5,0.5,0,0,0] is to have equal focus on goals and assists. We define an discrete action set of such variable actions where each action can be interpreted. We can even go beyond to state each action vector is also a player profile. Thus, each decision or action vector can be studied as the effect of cumulative team points by transferring in a player matching with the profile. This allows us to ask questions like did the manger's strategy of focusing on more goals work better compared to the strategy of having more assists etc. The terms player profile , action, decision or player embedding are used interchangibly from here on out.

1.3.1 Simulator

The simulator is a transfer simulator where given an actual team t and an input player p_{in}, the simulator creates a new team t', by transferring out a player p_{out} and transferring in the input player. The transferred out player p_{out} is chosen based on the least performing player of the same player as p_{in}. This is done for every game week.

1.3.2 Embedding Approach

We create an embedding for a team and for a player. The team embedding Z_t is just a vector of top correlated team attributes. These features are chosen so that no two teams at a given game week have the same embedding. Covariance of features between two teams must be high. As for the player embedding Z_p, we have a set of weighted probability vectors as stated above. In the action set, we also add in an extra action for no transfers since we don't want to maybe transfer out a good performing player. An analysis study is needed to understand the set of actual players which matches with a profile. If there is a high variance in the types of players, then a ranking for all players is required. Thus, we seek to either reduce variance in our candidate players by modifying the action set. We can also add player physical attributes like speed, ICT index to reduce to a player type. Having a good player profile set is important since we make trajectories based on this discrete set and should cover most of the player space set. We interpret certain embeddings to player types as follows:
1. More goals scorers are forwards and secondly midfielders
2. Least red cards and yellow cards are goals keepers
3. If clean sheets are highly correlated to points then they are probably goals keepers or defenders since forwards gets 0 points and midfielders get 1 point.

1.3.3 Scout

The scout model is a player profile. The scout model takes a player profile, matches it with the player rankings table and splits out a candidate.

1.3.4 Model

We use a 3 layer feed forward fully connected neural network. The network takes in two inputs which reinforcement learning terms are the state and action. The state is Z_t which is just the team embedding, and action is the player profile. Before training , we assume the following:
1. Teams formed through player transfers follow a Markov chain. 2. We use a Monte Carlo approach to evaluate expected gains at each state. We simulate episodes where each episode is a set of transfers happening until termination (budget runs out or end of game week). Each transfer is deemed to be good or bad in terms of a reward $r = 0 \; if \; p_{in}.points >= p_{out}.points \; else \; -1$. The expected gain is $E[G_s|S = s]$ and $G_s = R_s + \gamma R_{s+1} + \gamma^2 R_{s+2} + \gamma^3 R_{s+3} + ... + \gamma^W R_W$. W is the number of game weeks and γ is a dampening factor which can tell neural network to focus on short term or long term rewards. The output of the NN is the $Q(s, a)$ value which tells how good a state is. While training, we use an loss $L = MSE(Q(s, a), E[G_s|S = s]$. This will allow us to estimate goodness of states by learning from experience. We just show enough simulations using player transfers so that the NN has a good enough understanding of the team formation space.