THE LONDON SCHOOL OF ECONOMICS AND POLITICAL SCIENCE

Machine Learning Methods for Fantasy Football

[redacted]

A Dissertation submitted to the Department of Mathematics
of the London School of Economics and Political Science
for the degree of Master of Science

September 7, 2020

Summary

This dissertation looks at machine learning methods to tackle the sequential decision making problem that is the game of Fantasy Football or more specifically, Fantasy Premier League (FPL). It takes a look at the landscape of the decision making problem, one of which can be broken down into two core subproblems: (i) player performance prediction or points prediction, and (ii) team formation optimisation. The current best benchmark for an automated FPL manager was presented in the paper by Matthews et al. [25]. In this dissertation we shall explore the theory that underpins their approach, much of which is based on reinforcement learning and bayesian inference, whilst also presenting our own alternative supervised learning approach to the problem.

We take a look at the first subproblem by exploring current methods for modelling football matches, much of which has stemmed from the work of Dixon and Coles [11]. We also look at an extension of the Dixon and Coles paper, namely the Dixon-Robinson model [10] and look at how this can be utilised for points prediction in FPL. We show that one can use supervised learning methods to predict player points directly without any simulations involved, potentially circumventing the computational expensiveness that comes with the Monte Carlo simulations Matthews et al. used. The second subproblem can be solved efficiently once player performance is forecasted, more precisely we can choose an optimal team once we have forecasted individual player points within FPL. This can be done by using a Mixed Integer Programming formulation by framing the team selection process as a Multidimensional Knapsack Problem. This was shown to work well in paper by Matthews et al. [25] and we present an overview of how it works. Furthermore, we then combine the theory for the two subproblems to tackle the grand problem of sequentially choosing an optimal team, and evaluate the results of our supervised learning model in the last section. We then swiftly conclude and wrap up the dissertation with some advice on possible directions for future work.

Contents

1. Introduction

Fantasy sports are a hugely popular collection of online games whereby humans select teams on a weekly or even daily basis to reflect what they believe will happen in real life sporting events. By coming up with good predictions a player can land themselves some pretty lucrative cash prizes. To give a sense of scale of the industry, in the US and Canada fantasy sports are forecasted to be worth $33 billion in 2025 [29]. In this dissertation we will be primarily focused on association football (soccer)[1]. The game of fantasy football can be traced back to 1971 when Bernie Donnelly made an early version of the game [12]. The game took off and gained immense popularity when Andrew Wainstein took inspiration from popular fantasy sports games in the United States, and then formed his company *Fantasy League* in 1991. He later partnered with The Telegraph newspaper in 1993 to form an early popular version of the game where players had to phone in or use post to make their weekly transfers, as this was before the internet was widely used by consumers [15]. The boom of fantasy football can also be accredited to the popularity of a British television programme called *Fantasy Football League* which started in 1994[2]. The official Fantasy Premier League (FPL) game[3] (the version we shall focus on for the rest of the paper) started up for 2002-03 English Premier League season. Initially it attracted just 76,000 players from around the world and in 2020 attracts 7.6 million players from 275 regions of the world (includes minor territories) [33].

1.1 Background

In essence, all fantasy sports games work in the following way: The player must select an optimal team from a vast group of players available based on what they think

[1]We will refer to the game as football for the rest of the paper.

[2]The programme originated from a BBC Radio 5 show with David Baddiel as one of the hosts where he later teamed up with Frank Skinner for the televised version.

[3]`https://fantasy.premierleague.com/`

will happen in real life to maximise their points across the season. We can therefore break down the problem of FPL into a two step problem: (i) **Player performance prediction or points prediction**, and (ii) **Team formation optimisation**.

In their own right these are two interesting, difficult problems. The first problem of player performance prediction can be handled with either statistical modelling [11] or supervised learning approaches [1]. When thinking about the wider scope of the ability to predict the outcomes of matches outside of fantasy football, there are a number of possible areas this could be used, for example, exploiting the betting market, which was the purpose goal of the Dixon-Coles model [11]. A strict betting strategy developed as a result of accurate statistical models could allow someone to make a profit. The reality is however that football and in particular the English Premier League is notoriously difficult to predict and the current best model accuracy is 56.7% demonstrated in the paper by Baboota and Kaur (2019) [1]. An excellent overview of current models that exist are summarised in the survey by Beal et al. [3], though a brief search online shows that this is a very active area of research with authors claiming varying accuracies based on different sample sizes and competitions. For instance, in the paper by Hucaljuk and Rakipovi (2011) [17] they manage to achieve a 68.8% accuracy on Champions League matches, a different competition to the English Premier League. This highlights the main factors one should consider when evaluating the prediction accuracy of models. In the Champions League during the group (early) stages, there are many teams in the competition of lesser quality and you could claim that the games are more predictable than say the average English Premier League match.

Due to the vast amounts of data becoming available and the rising popularity of machine learning research, it is expected that model accuracy will improve over time. Researchers are now able to engineer synthetic features from avenues such as scraping social media sites and using Natural Language Processing tools to extract vital, game changing information [2], that was not available in 1997 when Dixon and Coles set out to exploit the betting market. Though progress will be slow and the best model accuracies have crawled up since the work of Dixon and Coles. Potentially what is needed are new sources of data that are richer than we have available now. Areas which could be of interest are physical condition of players, weather conditions, team morale and changes in personnel.

Lastly, it is also important to highlight the importance of a teams *form* in football. A good example[4] of this is when Southampton lost to Leicester City a record 9-0. This fixture much inflated Leicester's attacking statistics whilst also making Southampton look incredibly poor defensively on paper. After this result Southampton managed

[4]`https://www.bbc.co.uk/sport/football/50092694`

to slowly bounce back from this and garnered impressive victories over strong teams such as Chelsea and Tottenham Hotspur and had an impressive campaign thereafter. On the other hand, Leicester did not manage to replicate this kind of performance going forward and underperformed for the rest of the season. Had a decision maker some weeks after this result considered the entire seasons statistics only, they may undervalue Southampton's ability as a team and overvalue Leicester's. By assessing shorter time frames rather than the entire season, team performance fluctuations can be detected. This highlights the importance of time series data as it is essential for detecting how a team is performing at the time.

Moving on to the second topic of team formation optimisation, Matthews et al. (2012) [25] showed that team optimisation selection can be formulated as a Multidimensional Knapsack Problem (MKP). This is an efficient way to view the problem as opposed to the brute force and filtered brute force methods tried by Landers and Duperrouzel (2018) [21]. Essentially the MKP is a formulation of the team selection process where we want to select a team which maximises the total expected points (where the expected points are derived from some model), with respect to certain constraints posed by FPL. Matthews et al. also showed the importance of considering the long term effects of decision making. They employed a reinforcement learning approach to the sequential decision making problem of choosing a team each week, and show the benefits of considering the long term effect of decisions rather than just myopic ones (singular gameweek), yield significantly better results. Specifically, they take advantage of the Q-learning [39] and Bayesian Q-learning algorithms [9] to help quantify the long term effects of ones actions. In this dissertation we shall look at alternative ways of considering the long term effects of decisions outside of the reinforcement learning framework, employing supervised learning methods to forecast points in future gameweeks. When thinking about the wider scope of sequential team optimisation, there are a number of areas in which these methods can also be applied to such as, selecting the optimal team for a set task in a workplace. This is also common topic in the Multi Agent Systems literature under the subfield of coalition formation. Examples of this can be seen in [7] and [30].

Tieing both of these separate tasks together and we have our sequential optimal team selection problem. The reinforcement learning framework Matthews et al. [25] used set the benchmark for an automated FPL manager and achieved a rank in the 1.1st percentile of players. To briefly explain the procedure, they frame the problem as a *Partially observable Markov decision process* (POMDP) and cast this as a special formulation of a *Markov decision process* (MDP) called a *Belief MDP*.[18]. Markov decision processes are an ideal framework for sequential decision making problems. Within this framework we have *states*, *actions*, and *rewards* that are all encountered by an agent (the decision maker), the *environment* is what encapsulates all of these things. The agent's objective is to therefore discover an optimal way of behaving called a *policy*, through interactions with the environment. Where in a given state, they are presented with a selection of choices (actions) by which they must select the best action such that they receive the most reward. The term 'partial observability' is concerned with the fact that a decision maker may not know what state it is currently in and we must make necessary adjustments to the standard MDP framework to incorporate this uncertainty. This will all become more clear shortly and we will give a focused, more formal introduction to reinforcement learning in the next chapter. Now that we have briefly discussed MDPs we can further discuss the rest of the procedure, Matthews et al. take advantage of the Dixon-Robinson model [10], a model designed to improve upon the Dixon-Coles model [11], to simulate match outcomes and allow them to forecast player points. Once they have amassed all of the predicted point totals for relevant matches according the the FPL point scoring system (detailed in the next sub-section), they formulate team selection as an MKP and solve using IBM ILOG's CPLEX (a multi-purpose mathematical programming solver) thus forming the optimal team based on averages of simulations. They then repeat this process over multiple gameweeks therefore creating an automated FPL manager. In this dissertation we will also consider the approach of Landers and Duperrouzel [21]. The key differences between their approach and the approach of Matthews et al. [25] is essentially using supervised learning to form a team as opposed to simulations. They are also only concerned with a different fantasy sports game and thus we will look at applying their approach to FPL instead.

1.2 Fantasy Premier League

We now give a detailed explanation of how to play the game, outlining the rules and objectives. We will also refer to the player of the game as the *manager*.

Each English Premier League season usually begins sometime in August, prior to the season start a player must select their team of 15 players consisting of exactly:

- 2 Goalkeepers
- 5 Defenders
- 5 Midfielders
- 3 Forwards

Each of these players has a value of £V million and the manager has a budget of £100 million to begin with to select his team. Another restriction in which the manager has to abide by is that an FPL team can have a maximum of 3 players from one EPL team. In the EPL there are 20 teams to select players from and over 500 players in total. A player may use any prior knowledge of teams and players to make informed decisions on who to select, such as the previous seasons' results.

In the EPL there are 38 fixtures/matches for each team which translates to 38 gameweeks (GWs) in FPL. In every gameweek the manager has the ability to alter his team from the previous gameweek in the form of player transfers. Each week a manager is allowed 1 free transfer to swap out one of his current players for a new player whom they do not currently own. You can make extra transfers, but they come at a cost of -4 points. You may choose to swap out a player because they are not playing well or it may be that you need to make adjustments to fit in a player who is in form. All changes to the team must be confirmed 1 hour before the first match of the gameweek (the gameweek deadline). Something to note is that players prices can change by £0.1 million, for example if a player played well in the gameweek it is likely his price will go up once or twice, or maybe even three times. The same goes for players playing poorly, they are likely to see a price drop. This highlights the importance of timing transfers correctly as a manager might be priced out of certain moves or lose value in their squad.

In each gameweek you are also allowed to field 11/15 players in your squad. The remainder are placed on the bench and will score 0 points unless one of your starting 11 does not play and your bench players will be automatically substituted in their place (putting importance of the order of players on your bench). You are also given the choice of a captain, where the chosen player will receive double the points he would normally receive in that gameweek. In addition you must also select a vice-captain such that if the captain does not play for whatever reason you have a backup option.

Furthermore, managers also have access to 4 special *chips*. When played correctly they can garner many extra points. The chips are the following: a *Wildcard*, where you can make unlimited transfers without penalty. *Bench Boost* is when you can gain points from your entire 15 players including those on the bench and we also have the *Free Hit* chip where for one gameweek only you can make unlimited changes to your team for one week only. This is different to a wildcard as the changes are only temporary and your team will revert back to it's old state after the gameweek finishes whereas a wildcard is permanent. The last chip is *Triple Captain* and when played your captain will receive triple points instead of the usual double. Wildcards are unique in the sense that you are given two a season, one for the first half of the season (GW1-19) and one for the second half of the season (GW20-38).

Lastly we detail the FPL point scoring system. Players may obtain points according to Table 1.1. For clarification, an assist is awarded to the player who makes the final pass to the goalscorer. A clean sheet is awarded to a player if their team concedes 0 goals throughout the match. In addition, there is also a bonus point scoring system

Action	Points
Play up to 60 mins	1
Play more than 60 mins	2
Goal scored by goalkeeper/defender	6
Goal scored by midfielder	5
Goal scored by forward	4
Assist	3
Clean sheet by goalkeeper/defender	4
Clean sheet by a midfielder	1
For every 3 shots saved by a goalkeeper	1
Penalty save by a goalkeeper	5
Penalty miss	-2
For every 2 goals conceded by a goalkeeper/defender	-1
Yellow card	-1
Red card	-3
For each own goal	-2

Table 1.1: FPL Point Scoring System

where a player may earn themselves an additional 1-3 points based on set criteria. We won't detail this here as it is not of high importance. For further information or any additional clarification one should consult the official FPL rules[5]. In summary, an FPL manager must choose his squad of 15 before the start of the season and in each of the following 37 gameweeks must choose a team which they think is going to score the most points. Each week they can change their team accordingly, swapping out underperforming players for ones who they think will get them more points. They must also choose their starting 11 and their captain/vice-captain each week. Lastly, they must also consider the optimal gameweek to execute chips for maximised point rewards. All of the team entries are ranked weekly on the basis of who has the most points. The winner is the manager who has accumulated the most points at the end of the season.

1.3 Outline of the Dissertation

We will start out in Chapter 2 with a focused introduction to reinforcement learning. This is by no means an attempt to give a broad and thorough introduction to the topic, we only cover the elements of reinforcement learning necessary to understand the concepts that are introduced thereafter, specifically in the context of FPL. It

[5] `https://fantasy.premierleague.com/help/rules`

will however be good enough such that the reader has adequate and well rounded knowledge of the subject upon completion of the chapter. We cover key concepts such as Markov Decision Processes, the fundamental framework for reinforcement learning algorithms. We then discuss the key problem in reinforcement learning that is balancing exploration and exploitation and this is then followed by introducing an algorithm that can solve or approximately solve a Markov Decision Process called Q-learning. It is important to note that the field of reinforcement learning is very rich and a highly active and dynamic research area, companies such as DeepMind[6] and OpenAI[7] are paving the way for new research and there are many new developments year on year. For a wider scope and a much more thorough introduction to the subject then the textbook 'Reinforcement Learning: An Introduction' by Richard Sutton and Andrew Barto [34] is an excellent place to start.

Once the core key concepts of reinforcement learning have been introduced we move on to Chapter 3 where we explore methods that attempt to model the game of football. The original Poisson model for football scorelines was presented in the seminal work by Maher [24] and then improved with a more accurate model fit by Dixon and Coles [11]. We present and explain the Dixon-Coles model and then move on to an extension of it that is more suitable for modelling football scorelines. These models can be used to simulate football match results and thus can provide insight into player performance as shown in the paper by Matthews et al. [25], therefore it is possible to derive player points predictions from this also. We then proceed with a method in which we can circumvent costly sampling by using a supervised learning method to forecast player points based on various features that are indicative of a players form.

Then moving on from football modelling, in Chapter 4 we are essentially tieing all of the theory together. We present the reinforcement learning approach of Matthews et al. [25] and explain a little more in depth about the Bayesian inference theory that they use. We then finally get to the Multidimensional Knapsack Problem (MKP) formulation of FPL team selection and give a definition of what it is. Further to this, we will detail our own alternative approach which utilises the supervised learning method from Chapter 3 and combine this with the MKP to select an optimal team. Finally, in Chapter 5 we evaluate our supervised learning approach and summarise our computational implementation. We explore how it performed across a prior FPL season and give some criticism about how well it fared. We then outline some potential areas for improvement and then in Chapter 6 conclude.

[6]`https://deepmind.com/`
[7]`https://openai.com/`

2. Reinforcement Learning

In this section we give a focused introduction necessary to provide the reader with adequate knowledge to understand the Reinforcement learning (RL) [34] approach to FPL. Reinforcement learning can be described as a group of solution methods to the problem of an *agent* learning about the *environment* in which it may or may not have no prior knowledge of. Thus, given any situation or *state* presented to the agent we aim to find a mapping from this state to an ideal action.

A large amount of early work in reinforcement learning was done in the field of animal learning through trial and error. As humans we are often presented scenarios in which we have no prior knowledge about. In this situation, there may be multiple choices or actions that we can make. Provided we choose to make one of these actions, the world around us will respond with feedback. We are then well equipped to determine the utility of making such an action and can remember what happened, therefore when this situation presents itself again, we can make more informed decisions. We may also call the utility here the *reward* associated with performing such an action. We may visualise this process as the feedback loop presented in Figure 2.1.

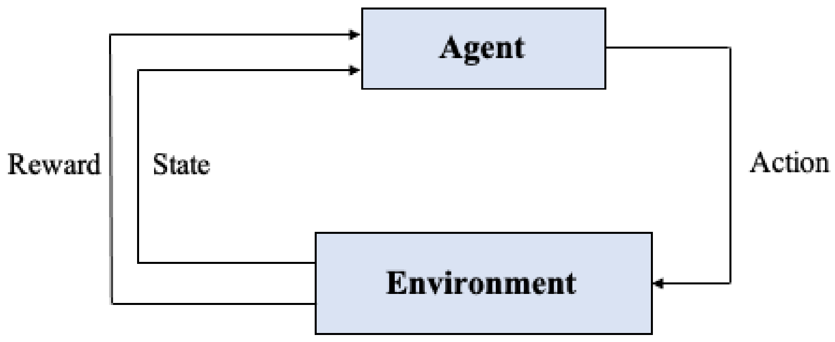


Figure 2.1: The agent-environment feedback loop

In this section we would like to translate this process in a computational sense. How might we go about teaching a computer how to deal with situations it has no idea about? How does a computer determine which actions to make when presented with information about the environment? These are questions that we aim to answer in the following sub-sections. We first start off with some basic terminology needed for reinforcement learning, we then define *Markov Decision Processes* and it's variants, and then explain how they can be used as a framework for modelling decision making. In addition to this, we explain the exploration-exploitation dilemma that plagues reinforcement learning and then present an algorithm that can provide solutions or approximate solutions to these MDPs. In this introduction we shall present various concepts in such a way that is similar to the introduction scheme found in the book by Sutton and Barto [34]. Unless stated or cited otherwise the definitions and theorems are taken verbatim or similarly from there.

2.1 Basic Definitions

Outside of the agent-environment feedback loop there are four main components that allow us to construct a computational reinforcement learning system. These are the following, a *reward signal*, a *policy*, a *value function* and an optional *model* of the environment.

Definition 2.1. A **reward signal** is received by an agent after each time step, where the time step represents where a decision is made. Specifically, it is the immediate numeric value provided by the environment, given that the agent has chosen an action in a given state.

Remark. The primary goal of the agent is to maximise the cumulative rewards that they receive over time.

Definition 2.2. A **policy** is a mapping from a perceived state to an action. An agent will aim to learn an optimal policy by interacting with it's environment, where the optimal policy is the one where an agent makes decisions that maximise it's reward in the long run.

Definition 2.3. A **value function** defines the value of being in any given state. It allows us to determine the total amount of reward an agent can expect to receive in the future when starting at any specific state.

Whilst a reward is defined in the immediate sense we may also want to consider the long-term effects of taking actions. The value function allows us to do this by letting us quantify 'how good is to be in this state?'. The value function is integral to many reinforcement learning algorithms, we will elaborate further on this shortly.

Definition 2.4. An agent can also decide that it wants to learn a **model** of it's environment. A model is something that can simulate the dynamics of an environment. It can provide inference about what may happen in the real environment given certain state-action pairs.

Models are optionally learnt in the sense that we can split all RL algorithms into *model-based* and *model-free* methods. As alluded to in the definition, a model can simulate an environment. If one has a perfect model of the environment then it is easy to perform say Monte Carlo simulations with the model and figure out an optimal policy for the environment without interacting with the true environment much at all. This however is seldom the case and we rarely have good models of the environment, and it can take a tremendous amount of data to approximate them well enough. Nonetheless there are still plenty of model based algorithms for reinforcement learning and there are situations where they can be utilised well. A good example of where we have a model of the environment is in the game of Chess. We know all of the moves that a player can make with the various pieces on the board and we can thus simulate play-outs of the game given the current positions of the pieces. The process of simulating play-outs is called *planning* and there is a vast amount of literature on this topic alone. Such methods can be found in [34], [32] and [20].

2.2 Markov Decision Processes

Markov Decision Processes (MDPs) are the way in which we can frame a sequential decision making problem. This formulation originated in the paper by Richard Bellman [4] in the 1950's. Going back to our original agent-environment feedback loop, MDPs take this behaviour and incorporate it in discrete time steps. For example, for each $t = 0, 1, 2, \dots$ the agent is provided by the environment, a state, and given this state the agent must select one of the actions available to it. In addition to this, the next state given some action is made, is stochastic. Meaning that if we take an action a in a given state s there is uncertainty about what may be the next state s' and there is a probability distribution over what may be the next state. We now formally define the process [42].

Definition 2.5. A **Markov Decision Process** is a 4-tuple $(\mathcal{S}, \mathcal{A}, p, r)$ where:

- $\mathcal{S}$ is the set of states called the state space,
- $\mathcal{A}$ is the set of actions called the action space,
- $p(s', r \mid s, a) = \Pr(S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a)$ is the probability that an action a in the state s at time t will lead to a state s' and yield a reward r at time $t+1$. $S_t, S_{t+1} \in \mathcal{S}, A_t \in A$ and R_{t+1} is from a set of rewards $\mathcal{R} \subset \mathbb{R}$,
- $r(s, a, s')$ is the immediate expected reward that an agent will receive after transitioning from a state s to a state s' due to taking an action a.

Remark. The state and action spaces may be finite or infinite.

For now, we shall only stick to the case of *finite MDPs* where the state and action spaces are finite.

Optimisation objective of the agent

As mentioned previously, the agents goal is to maximise the reward it receives in the long run. This equates to the agent finding a good policy π, where π is a function of a state s, $\pi(s)$. We now outline the tools necessary to find such a policy.

Definition 2.6. The **cumulative reward** G_t is defined in the following way:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

where γ is some discount factor $0 \leq \gamma \leq 1$.

The inclusion of γ is because in many instances we would like to prioritise the rewards that are in the nearer future. This value is usually closer to 1 in most cases, a value of zero would mean that the agent acts myopically and only cares about the immediate reward. The cumulative reward essentially allows us to quantify the reward the agent will receive in the long term.

Remark. The sum in Definition 2.6 can be also be defined recursively. This representation will be useful shortly.

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} \dots \\ &= R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \dots) \\ &= R_{t+1} + G_{t+1} \end{aligned} \tag{2.1}$$

Adding to this, an *episode* is when we can break down the cumulative reward into a finite sequence, i.e. the sum in Definition 2.6 is defined only up to some finite time step T. The concept of episodes will be useful later when we start to outline some RL algorithms. For example, an episode may translate to a single play-out of a game. Moreover, we briefly touched on value functions in Section 2.1 we now define explicitly the type of value functions that we will be dealing with regards to MDPs.

Definition 2.7. For all $s \in \mathcal{S}$, the **state-value function** v with respect to a policy $\pi : \mathcal{S} \mapsto \mathcal{A}$ is defined by the following:

$$v_\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s \right]$$

where $\mathbb{E}_\pi[\cdot]$ denotes the expected value of the cumulative reward given that the agent is in a state s and is following a policy π.

Definition 2.8. For all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$ we can define the **action-value function** q with respect to a policy $\pi : \mathcal{S} \mapsto \mathcal{A}$ in the following way:

$$q_\pi(s, a) = \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a\right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \;\middle|\; S_t = s, A_t = a\right]$$

where $a \in \mathcal{A}(s)$ denotes the set of actions available in a state s.

The action-value function is similar to the state-value function, however it differs in the sense that now we provide a value for each specific action available in a state s and we are not just valuing the state itself. Both are useful tools for solving MDPs and we will explain how momentarily. A simple way to view the effectiveness of say, the state-value function, is that the agent should always want to aim to be in states of high value as these are the positions where it expects to receive a lot of long term reward from. The following theorem will allow us to arrive at an important result called the Bellman equation.

Theorem 2.9 (Tower Rule)**.**

$$\mathbb{E}_\pi[v_\pi(S_{t+1}) \mid S_t] = E_\pi[E_\pi[G_{t+1} \mid S_{t+1}] \mid S_t] = E_\pi[G_{t+1} \mid S_t]$$

Proof. We first simplify the notation by dropping the subscripts and let $s = S_t$, $g' = G_{t+1}$ and $s' = S_{t+1}$.

$$\begin{aligned} \mathbb{E}_\pi[G_{t+1} \mid S_{t+1}] &= \mathbb{E}_\pi[\, g' \mid s' \,] && \text{(simplify notation)} \\ &= \sum_{g'} g \, p(g' \mid s'). && (2.2) \end{aligned}$$

By conditioning on $S_t = s$ and taking expectation of (2.2), we get the following:

$$\begin{aligned} \mathbb{E}_\pi[\,\mathbb{E}_\pi[G_{t+1} \mid S_{t+1}] \,|\, S_t\,] &= \mathbb{E}_\pi[\,\mathbb{E}_\pi[\, g' \mid s' \,] \mid s] \\ &= \sum_{s'} \sum_{g'} g' \, p(g' \mid s', s) \, p(s' \mid s) \\ &= \sum_{s'} \sum_{g'} \frac{g' \, p(g' \mid s', s) \, p(s' \mid s) p(s)}{p(s)} \\ &= \sum_{s'} \sum_{g'} \frac{g' \, p(g' \mid s', s) \, p(s', s)}{p(s)} \\ &= \sum_{s'} \sum_{g'} \frac{g' \, p(g', s', s)}{p(s)} \\ &= \sum_{s'} \sum_{g'} g' \, p(g', s' \mid s) \\ &= \sum_{g'} \sum_{s'} g' \, p(g', s' \mid s) \\ &= \sum_{g'} g' \, p(g' \mid s) \end{aligned}$$

$$= \mathbb{E}_\pi[g' \mid s] = \mathbb{E}_\pi[G_{t+1} \mid S_t].$$

□

Whilst this proof may seem a bit tricky, it is the same as the general case proof presented in [41], just adapted to the context of MDPs. Now that we have introduced the tower rule[1] in context we can now introduce the Bellman equation [4]. For now we stick to the discrete case where the state and action spaces are finite and later we will use a continuous version of it when we adapt this theory to FPL.

Theorem 2.10 (Bellman equation for v)**.**

$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a)[r + \gamma v_\pi(s')], \quad \text{for all } s \in \mathcal{S}$$

Proof. Note that $\pi(a|s)$ is used to denote the probability that $A_t = a$ if $S_t = s$ and the agent if following a policy π.

$$\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi[G_t \mid S_t = s] && (2.3)\\
&= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] && \text{(by (2.1))}\\
&= \mathbb{E}_\pi[R_{t+1} + \gamma \mathbb{E}_\pi[G_{t+1} \mid S_{t+1}] \mid S_t = s] && (2.4)\\
&= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] && \text{(by Tower Rule)}\\
&= \sum_a \pi(a \mid s) \sum_r p(r \mid s, a) r + \gamma \sum_a \pi(a \mid s) \sum_{s'} p(s's, a) v_\pi(s')\\
&= \sum_a \pi(a \mid s) \sum_r \sum_{s'} p(s', r \mid s, a) r + \gamma \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a) v_\pi(s')\\
&= \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a) r + \gamma \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a) v_\pi(s')\\
&= \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a)[r + \gamma v_\pi(s')].
\end{aligned}$$

□

We have included these proofs as it sheds some light on where these equations actually come from. The Bellman equations will be used for the RL approach to FPL that will be introduced later, and we want to avoid any ambiguity with regards to their origin.

Corollary 2.11 (Bellman equation for q)**.**

$$q_\pi(s, a) = \sum_{s'} \sum_r p(s', r \mid s, a)[r + \gamma v_\pi(s')].$$

Proof. Omitted. Very similar to proof for Theorem 2.10. □

[1] Also known as the law of iterated expectations

The Bellman equation is important because it expresses a direct relationship between the values of a state s and the values of its successor states s'. Remember that to solve the MDP we must find an optimal value function, as this will allow us to form an optimal policy. The Bellman equation will form a basis for a number of ways that the agent can actually learn v_π. When you have a finite MDP, value functions in fact define a partial ordering over policies, i.e. we can say that a policy π is better than a policy π' if $v_\pi(s) > v_{\pi'}(s)$ for all $s \in \mathcal{S}$. In this way of defining how one policy is better than another, there will exist an *optimal policy* v_*. It is important to note that there may also be more than one optimal policy such that if $v_\pi(s)$ is optimal then we may find another policy π' where $v_\pi(s) = v_{\pi'}(s)$ for all $s \in \mathcal{S}$. These optimal policies will share the same value functions which we define below. In essence, we are saying that there is always a policy at least as good as all other policies [34].

Definition 2.12. For all $s \in \mathcal{S}$ the **optimal state-value function** v_* is defined by the following:

$$v_*(s) = \max_\pi v_\pi(s).$$

Definition 2.13. For all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$ the **optimal action-value function** q_* is defined by the following:

$$q_*(s, a) = \max_\pi q_\pi(s, a)$$

where $\mathcal{A}(s)$ denotes the set of actions available in a state s.

Remark. We can also relate v_* and q_* in the following way:

$$q_*(s, a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a]$$

This is due to the fact for each pair (s, a), q_* gives the expected return of taking an action a in a state s and thenceforth given that the agent is following a policy that is optimal.

We now introduce the last piece of the puzzle for solving MDPs, the *Bellman optimality equations.* These equations express that the value of a state under a policy that is optimal, must be equal to the expected return for the best action from that state.

Theorem 2.14 (Bellman optimality equation for v)**.**

$$v_*(s) = \max_a \sum_{s'} \sum_r p(s', r \mid s, a) \left[r + \gamma v_*(s')\right].$$

Proof.

$$\begin{aligned}
v_*(s) &= \max_{a \in A(s)} q_{\pi_*}(s,a) && (2.5)\\
&= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t = s, A_t = a] &&\\
&= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] && \text{(by (2.1))}\\
&= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] && (2.6)\\
&= \max_a \sum_{s'} \sum_r p(s', r \mid s, a)[r + \gamma v_*(s')]. && (2.7)
\end{aligned}$$

□

Corollary 2.15 (Bellman optimality equation for q)**.**

$$\begin{aligned}
q_*(s,a) &= \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \;\middle|\; S_t = s, A_t = a\right] && (2.8)\\
&= \sum_{s'} \sum_r p(s', r \mid s, a)\left[r + \gamma \max_{a'} q_*(s', a')\right]. && (2.9)
\end{aligned}$$

Proof. Omitted. Very similar to the proof for Theorem 2.14. □

Equations (2.6) and (2.9) are both equally useful versions of the Bellman optimality equations for v_* and q_*. They are merely just more concise representations, further steps just involve expanding out the expectation into sigma notation. These forms are much easier to interpret the meaning of these equations. Furthermore, what makes the Bellman optimality equations so important? They are crucial due the fact that if say v_* is obtained it is easy to determine an optimal policy — the goal of reinforcement learning that we keep talking about. Let's assume we have access to v_*, for each $s \in \mathcal{S}$ there will be one (or more) actions where the maximum value can be acquired in the equation in Theorem 2.14. A policy that gives a probability of zero to all actions that do not achieve a maximum, is optimal. This can be best thought of as a one step ahead search. If v_* has been obtained then the actions that appear best after this one step ahead search will be optimal, i.e. given we have v_* we only need to consider the locally optimal action as this is the best action in the long run (as it already factored in the long-term rewards) and a *greedy policy* where we myopically select the best action in each successive state is the optimal policy.

When q_* is obtainable this is an even more powerful tool because we eliminate the one step ahead search element. The action-value function q_* effectively caches the result of all one step ahead searches. It can find an action, for any state s that maximises $q_*(s,a)$. It allows us to simply just select the locally optimal action without having to consider the successor states and their associated values. This is powerful in the sense that we do not need to know anything about the environment dynamics. This will become especially useful when we adapt all this theory to FPL due to the uncertainty in the environment dynamics and a lack of a model of the environment. This does however come at a cost, as we must represent the function as a function of state-action pairs instead of just states and when the state space and action space is large, this can become computationally infeasible. In many cases it is not actually possible to achieve the optimal value functions and we must settle for an *approximate solution* in any non-trivial domain. This is not to say that approximate solutions can't be good, there are many examples of where approximate solutions have produced amazing results like in [37].

2.2.1 Partial Observability

In many real world applications there is uncertainty in the state of the agent. The state may only be partially observable in the sense that we may have only partial information about the current state of the environment.

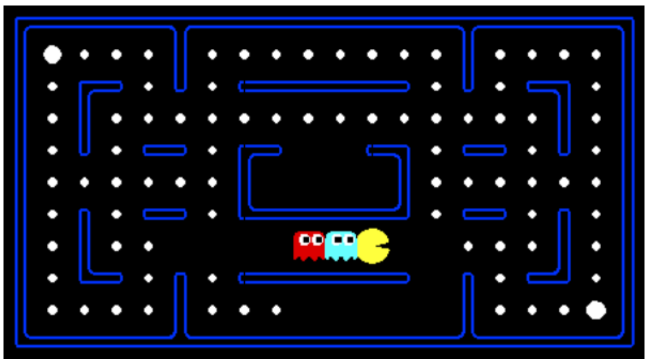


Figure 2.2: Pac-Man. Source: [28].

Example 2.16 (Partially observable Pac-Man)**.** Imagine an agent is trying to play a game of Pac-Man[2]. However, in this case the agent can only see the state of the game through the vision of the Pac-Man itself. Meaning that he can not see around walls, the same way humans can't. The rewards are the white dots and the states are the different positions on the 2D Grid that the Pac-Man can be. The actions are that it can go left, right or turn around as opposed to the up, down, left, or right that a normal player would have. Since the Pac-Man can no longer see where the ghosts are on the grid (unless they are in the agents view), or the full layout of the grid, the agent only has a subset of the information regarding the environment dynamics. Instead of seeing a state, what the agent receives each time it moves, is an observation of the environment, a limited view of what is truly happening. The observation could be many different scenarios based on the behaviour of the ghosts. However if one could compile a list of observations for each state, then we could define

[2]https://en.wikipedia.org/wiki/Pac-Man

a probability distribution for each of these based on successive play-outs of the game. The more times we play the game the more accurate these probabilities would be. These observations and probability distributions over each set for each state therefore become crucial for the Pac-Man's success in the game. The environment state is now partially observable as opposed to the fully observable state that an agent playing the game normally would have access to.

Now that it is clear what we mean by partially observable we are now ready to define the generalised MDP framework [23] to deal with applications where the state of our environment is not fully observable. The following definition is taken verbatim from [43].

Definition 2.17. A **Partially observable Markov decision process** (POMDP) is a 6-tuple (S, A, T, R, Ω, O) where:

- S is a set of states,
- A is a set of actions,
- T is a set of conditional transition probabilities between states,
- $R : S \times A \to \mathbb{R}$ is the reward function,
- Ω is a set of observations,
- O is a set of conditional observation probabilities.

With the POMDP we have introduced some extra characteristics to the traditional MDP framework. When an agent is in a state s and takes an action a, the environment will now transition to a new state s' with probability $T(s' \mid s, a)$ and the agent receives an observation $o \in \Omega$ that depends on the new state of the environment s', and the action a with a probability of $O(o \mid s', a)$. This then yields a reward and the same process then reoccurs at the next time step. This is different to the traditional MDP only in the sense that we can no longer fully observe the current state of the process. In summary we now get *observations* from the environment that depend on the transitioned state and action made to get there. We do not observe the state itself!

Belief MDPs

Solving POMDPs exactly is notoriously very hard. This is primarily due to the uncertainty in the underlying state. In most cases if the action and state spaces are large then it is generally impossible to solve a POMDP exactly, and we must look for approximate solutions instead. One way in which we may go about finding an approximate solution to a POMDP is to reframe the problem as a different class of Markov decision process called a *Belief MDP* posed in the paper by Kaelbling et al. [18]. This will be the formulation that Matthews et al. [25] used to tackle the game of FPL and we will speak more about this in Chapter 4.

Definition 2.18. The **Belief MDP** is a 4-tuple (B, A, τ, r) where:

- B is the set of belief states over the POMDP states,
- A is the action set we had in the original POMDP,
- τ is the belief state transition function,

$$\tau(b, a, b') = \Pr(b' \mid a, b) = \sum_{o \in \Omega} \Pr(b' \mid b, a, o) \Pr(o \mid a, b)$$

$$\text{where } \Pr(b' \mid b, a, o) = \begin{cases} 1 & \text{if the belief update with args } b, a, o \text{ returns } b' \\ 0 & \text{otherwise} \end{cases}$$

- $r : B \times A \to \mathbb{R}$ is the reward function on belief states,

$$r(b, a) = \sum_{s \in S} b(s) R(s, a).$$

Note: $R(s, a)$ and $o \in \Omega$ refers to reward function and the observation from the original POMDP.

Here we have constructed a *Markovian belief state* which allows us to formulate the POMDP as a regular MDP, where the state is now something we *can* observe called a *belief.* This Belief MDP is also defined on a **continuous state space** as there are an infinite amount of belief states due to the fact that there are an infinite number of probability distributions over the the set of states S. We can view the procedure as the agent now choosing actions in various states, based on their beliefs about the environment dynamics (the current belief state). In the case of FPL for example, we shall think of the beliefs as our views on the ability of a player or more specifically a number of probability distribution over their point scoring potential. We will have a distribution for various point scoring avenues such as goals scored or assists, for each player, based on their previous performances. Something to note is that we assume the belief state to be Markovian, this may not be the case in many real world environments. To recap this in context, being Markovian means that the conditional probability of future states of the process depends only upon the present belief state b_t and not the ones that came before it $b_{t-1}, b_{t-2}, \ldots, b_{t-k}$. Thus we can say the belief state is a sufficient statistic of the past, meaning that the current belief state should perfectly summarise all prior belief states and we needn't consider the ones that came before it. In practice the process may depend on previous states of the process, but for the rest of this dissertation we make the assumption that it has the Markov property. After each time step in the process we receive new observations and update our belief state accordingly to better reflect our beliefs about the environment. Updating the belief state itself requires special consideration, the method which we will use to do so will be detailed in a later chapter. Since the Belief MDP is a POMDP cast as traditional MDP we can use traditional methods to solve them approximately and this is the key benefit of reframing the problem.

2.3 Exploration vs. Exploitation

Before outlining some of the important algorithms that we will use to solve MDPs we first talk about the dilemma of balancing exploration and exploitation in reinforcement learning. This problem is best explained in the case of multi-armed bandit problems. These multi-armed bandit problems can be thought of as stateless MDPs where we can take actions and receive rewards for those actions.

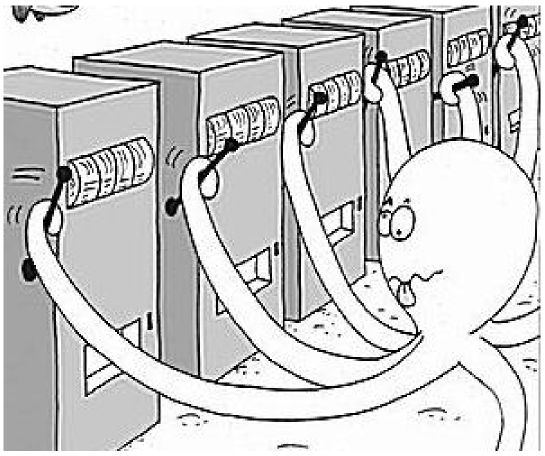


Figure 2.3: The multi-arm bandit problem. Source: [31].

Example 2.19 (Multi-arm bandit problem)**.** Imagine we have a gambler (the agent) who has access to 6 slot machines. The actions available to the gambler are to pull one of the levers to any of these slot machines at each time step. The gambler's goal is to maximise the reward he will receive by pulling a sequence of levers. The gambler has no biases toward any of the slot machines and has never played on any of them before. He pulls the first lever and receives a reward of £100. Here we have our action-value function $q(a)$ (no parameter of s since we are stateless) which keeps an average of the reward that we receive from pulling each lever. Now without any knowledge of the machines beforehand we look to gain an approximation to Q which we will call $\hat{q}$. This approximation will keep an average of the reward received over time. Therefore we can say that $q(a) = \mathbb{E}[R_t \mid A_t = a]$.

Approach #1 — (Greedy strategy)
The gambler will choose an action based on what is the current maximum action-value estimate.

Approach #2 — (ϵ-greedy strategy)
The gambler with will choose a random action with probability ϵ and choose the current maximum action-value estimate otherwise.

Why might the Approach #2 be better than the Approach #1? Remember the gambler pulled the first lever and got £100 reward, therefore his current action-value estimates are $[100, 0, 0, 0, 0, 0]$. He goes with the greedy strategy and selects the first lever again, this time giving him $-$£10. The estimates now look like $[45, 0, 0, 0, 0, 0]$. It is easy to see why this might not turn out well. The gambler may get stuck in a sub-optimal strategy repeatedly pulling the first lever until all of the estimates are back at 0. He never decides to pull any other levers and doesn't know what the other machines have to offer. We can say that this strategy does not allow the gambler to explore his options well. Unbeknownst to the gambler, the second machine is broken and is paying out £10 every time its lever is pulled. Had the gambler not been following an ϵ-greedy strategy he may have never found out this was the case. We have shown that by implementing some randomness in action choices, this is an easy way to ensure the agent explores rather than simply continuing to exploit its current best known option.

There are other more sophisticated methods for ensuring that an agent explores, one of which we will detail shortly. Though in summary, this example shows why it is important we balance exploration and exploitation. Remember the goal of the agent is to find the best policy. This is very unlikely if the agent doesn't explore its options at all throughout time. A very simple way to guarantee exploration is through the ϵ-greedy strategy, this works well in practice and $\hat{q}$ is guaranteed to converge to the optimal action-value function [34]. Efficient and broad exploration turns out to be especially vital in the case of Belief MDPs as you must inform and update all of various distributions that you hold in your belief state.

2.4 Q-learning

Something we are yet to touch on is how we actually go about 'solving' an MDP. As it is not the focus of this dissertation to discuss at length the various methods to find optimal policies, we stick to the ones that we shall use in the later stages of the dissertation. To give a brief idea of the landscape of different reinforcement learning algorithms, they fall into three categories. Monte Carlo methods, Dynamic Programming and lastly Temporal-Difference (TD) methods. TD methods are a mixture of both monte carlo and dynamic programming and the algorithms in which we use in this dissertation fall into this category. TD methods fall into the 'model-free' category that we alluded to earlier in the chapter. They learn by bootstrapping from the current estimate of the value function and like Monte Carlo methods sample from the environment. They also perform updates based on current estimates like dynamic programming [44]. Here, by bootstrapping we mean that we update estimates as we go along, using both old estimates and newer estimates to form an even newer, better one.

Q-learning is a model-free TD reinforcement learning algorithm. It was developed by Chris Watkins in 1989 [39]. Q-learning directly approximates q_* with a learned action-value function Q. It is called an off-policy algorithm due to the fact it can learn Q without following any sort of explicit policy. We outline the algorithm below.

Algorithm 1: Generic Q-learning

```
initialise all Q values $\forall\ s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$ arbitrarily
set the learning rate $\alpha,\ 0 < \alpha \leq 1$
for each episode e do
    Initialise s, the starting state
    for each step of episode do
        Choose an action $a$ available in a state $s$ using a policy derived from Q
          (e.g. $\epsilon$-greedy)
        Observe a reward $r$ and new state $s'$
        $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_a Q(s',a) - Q(s,a)]$
        $s \leftarrow s'$
    end
end
```

The algorithm is simple by nature. We are essentially learning the *quality* of state-action pairs. Imagine we had a large table for all state-action pairs and we are simply recording and updating our value estimates using the update rule for Q. This update rule is in fact the Bellman optimality equation for q that we introduced earlier. This algorithm is guaranteed to converge provided that all states are continuously visited. A proof of this can be found in [26]. By convergence we mean that we will get closer and closer to the optimal value function q_*. We will see an interesting variant of this algorithm in one of the following chapters and we also adapt it to handle a Belief MDP with minor adjustments.

In summary we have explored all of the necessary tools to understand the RL approach to FPL [25]. We have introduced the key framework of Belief MDPs and also shown that agents must explore as well as exploit their current options to achieve an optimal policy. Lastly, we introduced the Q-learning algorithm which will lay the foundations for a later variant of the algorithm used in the RL approach to FPL.

3. Modelling Football and Predicting Performance

Now that we have laid the foundations for reinforcement learning, we will now move on to the second piece of crucial material that is predicting player performance and match outcomes. As alluded to in the introduction, FPL can be divided into a 2 step problem, points prediction and team formation optimisation. Predicting player performance is right at the heart of FPL. If one can make good predictions about how a player or team will perform during the gameweek, then they will receive many points in return. Making good predictions in the game of football though is hard work indeed, and as recently stated the current best prediction accuracy is 56.7% for predicting the outcomes of matches (Win, Loss or Draw) [1].

In the paper by Matthews et al. [25] they employ the Dixon-Robinson model as a means of predicting match outcomes. The limitations of this was that it does not directly predict *individual player* performance but predicts *team* performance. Whilst they developed a neat heuristic to still take advantage of this model, it would be beneficial to predict individual player performance directly as opposed to deriving it from an overall team simulated performance. In this section we will present our own way of modelling individual player performance using supervised learning methods [13], which will utilise historical Premier League data. This approach is much inspired by the paper by Landers and Duperrouzel [21], where they applied supervised learning techniques to predict the amount of fantasy points a player will receive in weekly NFL fantasy competitions. We also give an introduction to the Dixon-Robinson model and begin our explanation by building upon the model of which it was based on, the Dixon-Coles model.

3.1 Dixon-Coles Model

The Dixon-Coles model [11] was largely based on the paper by Maher (1982) [24] where he showed the Poisson distribution was an adequate fit for the number of goals scored by both the home and away team in football. Dixon and Coles aspired to improve upon the shortcomings of this model. They envisioned that the model should also take into account the different abilities of both teams (both defensive and offensive), it should incorporate the observed home team advantage[1], the recent performances of teams (form) and also should consider the abilities of a team's recent opponents when determining what a teams form is. For example, we might 'let off' a mediocre team if their recent fixtures have been against tough opposition, once they are through a bad spell of fixtures they might perform well against a more evenly matched opposition. In the same way we can say a team is in truly bad form if they are typically known as high performers and are playing badly against weaker opposition in their most recent games.

To start off, we let the number goals scored by the home team i and away team j be denoted by the Poisson random variables $X_{i,j}$ and $Y_{i,j}$ respectively. Then

$$\begin{aligned} X_{i,j} &\sim \text{Poisson}(\alpha_i \beta_j \gamma), \\ Y_{i,j} &\sim \text{Poisson}(\alpha_j \beta_i \gamma), \end{aligned} \tag{3.1}$$

where α_i, $\beta_i > 0$, $\forall i$. The parameter α_i measures the 'attack' rate of teams and β_i measure the 'defence' rates. $\gamma > 0$ is a parameter which represents the home advantage referred to in the last paragraph. Dixon and Coles extended this basic representation and proposed the following model:

$$\Pr(X_{i,j} = x, Y_{i,j} = y) = \tau_{\lambda,\mu}(x,y) \frac{\lambda^x \exp(-\lambda)}{x!} \frac{\mu^y \exp(-\mu)}{y!} \tag{3.2}$$

where

$$\begin{aligned} \lambda &= \alpha_i \beta_j \gamma, \\ \mu &= \alpha_j \beta_i, \end{aligned}$$

and

$$\tau_{\lambda,\mu}(x,y) = \begin{cases} 1 - \lambda\mu\rho & \text{if } x = y = 0, \\ 1 + \lambda\rho & \text{if } x = 0, y = 1, \\ 1 + \mu\rho & \text{if } x = 1, y = 0, \\ 1 - \rho & \text{if } x = y = 1, \\ 1 & \text{otherwise.} \end{cases}$$

[1]Teams tend to win more when playing at their home stadium based on empirical evidence in the paper by Dixon and Coles [11]

After some statistical analysis Dixon and Coles concluded that apart from the scorelines [0-0, 1-0, 0-1, 1-1], the assumption of independence between either team scoring is reasonable. So to factor in this dependence for these specific scorelines we must introduce the τ function and more specifically a dependence parameter ρ, where $\max\left(-\frac{1}{\lambda}, -\frac{1}{\mu}\right) \leq \rho \leq \min\left(\frac{1}{\lambda\mu}, 1\right)$. If the parameter $\rho = 0$ this would correspond to independence between either team scoring their next goal, and the τ function is only put to use when $x \leq 1$ or $y \leq 1$ (goals scored for the home side x, and for the away side y). Otherwise the τ function is equal to 1 and we can assume independence for the rest of the other scorelines.

Parameter Selection

To select attacking and defensive ability parameters α_i and β_i for each team i we select them based upon whichever maximises the likelihood function

$$L(\alpha_i, \beta_i, \rho, \gamma; i = 1, ...n) = \prod_{k=1}^{N} \tau_{\lambda_k,\mu_k}(x_k, y_k) \exp(-\lambda_k)\lambda_k^{x_k} \exp(-\mu_k)\mu_k^{y_k} \tag{3.3}$$

where

$$\lambda_k = \alpha_{i(k)}\beta_{j(k)}\gamma,$$
$$\mu_k = \alpha_{j(k)}\beta_{i(k)},$$

and $i(k)$ and $j(k)$ denote the indices of the home and away teams playing in a match k respectively, where there are a total of $k = N$ matches to be played between teams. To be clear this is when we are trying to find the parameters for n teams in the league, in our case $n = 20$ as there are 20 teams in the Premier League. Therefore after we have estimated these parameters we should be left with the two sets of estimators for all teams, $\{\alpha_1, \ldots, \alpha_{20}\}$ and $\{\beta_1, \ldots, \beta_{20}\}$. In addition to this, to avoid the model becoming overparametrised Dixon and Coles also impose the following constraint

$$n^{-1} \sum_{i=1}^{n} \alpha_i = 1. \tag{3.4}$$

To incorporate team form changes, they extend (3.3) to construct what they call a 'psuedolikelihood'. It differs in the sense that now parameter estimates are formed up to some time t, for example the last 6 matches. They also downweight more historic results and give more importance to the most recent ones. We define the pseudolikelihood for each time point t,

$$L_t(\alpha_i, \beta_i, \rho, \gamma; i = 1, ...n) = \prod_{k \in A_t} \left[\tau_{\lambda_k,\mu_k}(x_k, y_k) \exp(-\lambda_k)\lambda_k^{x_k} \exp(-\mu_k)\mu_k^{y_k}\right]^{\phi(t-t_k)} \tag{3.5}$$

where t_k is the time that a match k was played, $A_t = \{k : t_k < t\}$ (the set of matches played before time t) and ϕ is a non-increasing function of time. The model now can reflect changes in team performance as (3.5) will calculate a parameter estimate based on games up to time t only. Something to note is that the notation has slightly been abused since α_i, β_i and γ are also time dependent. They also define the weight function ϕ by

$$\phi(t) = \exp(-\xi t). \tag{3.6}$$

The model that uses (3.3) for parameter estimates can be seen when $\xi = 0$, and increasing values of ξ give relatively more weight to more recent team performances. A similar method of likelihood maximisation is used to find an estimate of $\xi = 0.0065$, and it is important to note this was the value which maximised a likelihood function for *match outcomes* and not *match scorelines.* We will touch on how to modify this for scorelines in the Dixon-Robinson model in the next section. We shall not state the rest of the parameters α_i and β_i as they are outdated and different teams now occupy the leagues. However, it was interesting to see that the home advantage parameter γ seemed to stay constant over time in their study, meaning that the home effect is a very real phenomenon. It would be interesting to see how the home advantage parameter has varied in 2020 due to the COVID-19 pandemic not allowing fans to attend matches. This could potentially weaken the effects of the home advantage as teams are without their majority home support.

3.2 Dixon-Robinson Model

We first state two key definitions which outline the main features of the models mechanics.

Definition 3.1. Let $\lambda > 0$ be fixed. The counting process $\{N(t), t \in [0, \infty)\}$ is called a **Poisson Process** with rates λ if all the following conditions hold:

- $N(0) = 0$,
- $N(t)$ has independent increments,
- the number of arrivals in any interval of length $z > 0$ has a Poisson(λz) distribution.

Definition 3.2. If $N(t)$ is a Poisson Process with rate λ, then the **inter-arrival times** $T_1, T_2, T_3, \ldots$ are independent and

$$T_i \sim \text{Exponential}(\lambda), \quad \text{for } i = 1, 2, 3, \ldots$$

The key purpose of the Dixon-Robinson model was to extend upon the work of Dixon and Coles, specifically through the means of modelling *goal timings* in a match. This model allows us to give accurate probabilities to specific scorelines in a match as opposed to the match outcome prediction of a Win, Loss or Draw in the Dixon-Coles model.

The model can be described as a two-dimensional birth process where the home and away scores are thought of as two different species. This means that both teams scoring abilities throughout a match are captured by *two evolving poisson processes*, where the rates or intensities that define a teams scoring ability, change throughout a match depending on events that occur in the game, such as a goal being scored or a 1 minute time increment. The model aims to incorporate the phenomena of continuously increasing scoring rates[2] and the dependence of scoring rates on the current scoreline[3]. Something to note is that time in a match is usually 90 minutes plus additional time, and the convention that Dixon and Robinson use is that goal times are recorded as the integer part of the minute they were scored in. They are then rescaled such that $t \in [0, 1]$. The home and away scoring rates for a match k are defined as follows:

$$\lambda_k(t) = \begin{cases} \rho_1 \lambda_{xy} \lambda_k & \text{for } t \in (44/90, 45/90], \\ \rho_2 \lambda_{xy} \lambda_k & \text{for } t \in (89/90, 90/90], \\ \lambda_{xy} \lambda_k & \text{otherwise}, \end{cases} \tag{3.7}$$

and

$$\mu_k(t) = \begin{cases} \rho_1 \mu_{xy} \mu_k & \text{for } t \in (44/90, 45/90], \\ \rho_2 \mu_{xy} \mu_k & \text{for } t \in (89/90, 90/90], \\ \mu_{xy} \mu_k & \text{otherwise} \end{cases} \tag{3.8}$$

where λ_k and μ_k are defined as in (3.3), λ_{xy} and μ_{xy} determine the scoring rates for the home and away sides respectively when the score of a match is (x, y). Due to the fact that at the time the paper was written, goals scored in extra time (additional minutes added at the end of each half) were recorded as scored in the 45th or 90th minute, in turn they observed empirically that goals in this period before the end of each half, were increased. To be clear, this is because we are taking the 45th or 90th to be the time a goal was scored even though it might have been scored in the 1-5 minutes following (additional time). Thus they make necessary adjustments to incorporate this increased scoring rate, which is the purpose of ρ_1 and ρ_2. We need two of these parameters due to the fact that as mentioned previously the rates at which teams tend to score goals increases throughout the match, therefore it is necessary for ρ_2 to have a stronger effect on the intensities. To incorporate the

[2]It has been shown empirically that the scoring rates of teams tend to increase as a match progresses

[3]It was also shown in the same paper that scoring rates of teams tend to depend on the current scoreline

scoring rates dependence on the current scoreline, we define the following:

$$\lambda_{xy} = \begin{cases} 1 & \text{for } x = 0, y = 0, \\ \theta_{1,0} & \text{for } x = 1, y = 0, \\ \theta_{0,1} & \text{for } x = 0, y = 1, \\ \theta_{1,1} & \text{for } x = 1, y = 1, \\ \theta_{2,2} & \text{for } x - y = 0 \text{ where } x, y \geq 2, \\ \theta_{2,1} & \text{for } x - y \geq 1 \text{ where } x \geq 2, \\ \theta_{1,2} & \text{for } x - y \leq -1 \text{ where } y \geq 2, \end{cases} \tag{3.9}$$

where θ_{xy} is the scoring rate of a the home team when the current scoreline is (x, y). An almost identical definition can be made for μ_{xy}. Finally, the last adjustment that they make is to incorporate the increased goal scoring rates in a match by introducing the parameters κ_1 and κ_2.

$$\begin{aligned} \lambda_k^*(t) &= \lambda_k(t) + \kappa_1 t \\ \mu_k^*(t) &= \lambda_k(t) + \kappa_2 t \end{aligned} \tag{3.10}$$

This simple addition works due to the fact with small increases in t, the intensity rates will increase slightly. In summary, we now have rates which evolve continuously throughout a match whilst also being dependent on the current scoreline, the phenomena that was empirically observed in the paper by Dixon and Robinson. We will omit extensive details about the parameter estimation methods as it is out of the scope of the dissertation. Briefly, Dixon and Robinson use maximum likelihood estimators similar to (3.3) to estimate all of the parameters in their model. For more rigorous detail of these statistical methods it is advisable to consult their paper [10].

Though this is still a team based model, we can make use of it to forecast an individual players points/performance. For example, if a team x scores a goal then we shall award each player the points for a goal, in proportion to the probability distribution defined for each individual player scoring a goal. This distribution would be a Bernoulli distribution where each player would have some probability of scoring a goal in a match (more details on this will follow in the next chapter). If you can simulate for many point scoring avenues such as goals scored, assists, clean sheets etc. then you can essentially derive an expected points value v for each player in an arbitrary gameweek. This was the exact approach of Matthews et al. [25] and in this dissertation we would like to develop individual player point forecasting model rather than derive it from some team based model as we hypothesise this may be able to give better predictions.

3.3 Supervised Learning Approach

In this subsection we will focus on the theoretical side of our player based approach and leave the technical details to Chapters 4 and 5. A similar approach was used in the paper by Landers and Duperrouzel [21], though they used it in the American Football (NFL)[4] domain rather than football. As mentioned in the previous subsection, the Dixon-Robinson model is a team based model. We would like to explore ways of forecasting individual player points directly as opposed to deriving it from a team based one. All supervised learning methods [13] can be split into two types of problems, regression problems and classification problems. As in this scenario we are looking to predict a numerical value (expected player points), we are only concerned with the former type. To put it simply, all supervised learning techniques for regression work in the following way: For some input data $\boldsymbol{X} = (x_1, x_2, \ldots, x_n)$ where each variable x_i represents some description or measurement, we want to learn a function that can map $\boldsymbol{X}$ to some output y, in our case the expected points would be the output. Therefore given a dataset with n rows and however many columns or *features*, we can employ methods to learn a function that does just this.

Functions of this sort can be learned via various techniques, most familiar is the case of linear regression [5] where the method of least squares is used to fit a line to a collection of observations. Then for any given x value we can find an estimated y value associated with it. In the case of FPL, it is likely that we will encounter a lot of categorical data. Usually what is done in the presence of categorical data for regression problems is that the features x_i which contain categorical data are converted to numerical values that are supposed to be representative of the old categorical value. An example of such a categorical value would be the opposition team of a player whose points that you are trying to predict e.g. {Arsenal, Chelsea, Liverpool}. This can be problematic due to the fact that information or predictive power that the feature carries, can be lost, and in regression problems can skew the results. Therefore we look to supervised learning methods that are robust to categorical features. Tree based methods and more specifically ensemble methods such as Random Forests [22] are examples of this.

Due to these benefits, we can use a Random Forest regression model to predict player points. To briefly explain how a Random Forest works one must first consider what a Decision Tree is. A Decision Tree put simply is a series of ‘yes’ or ‘no’ type questions asked, and after we have answered all of these questions, the Decision Tree will provide a recommendation or output. This can be seen in Figure 3.1. Random Forests are a collection or *ensemble* of Decision Trees, where the final output will be the average of the outputs from all of the Decision Trees in the forest. Each tree

[4] `https://en.wikipedia.org/wiki/National_Football_League`

is generally created by a method called *Bagging* (bootstrap aggregating), a method that guarantees to produce diverse trees by training on different random subsets of the training data. A bagged sample is the one where we have sampled the training dataset *with replacement* (bootstrapping) [38]. Once the trees have been constructed, then the ensemble can make a prediction for a new instance and thus give an output, or in our case, predicted points. This concludes a high level overview of how Random Forests work and a more detailed explanation of the various intermediate calculations involved can be found in [13].

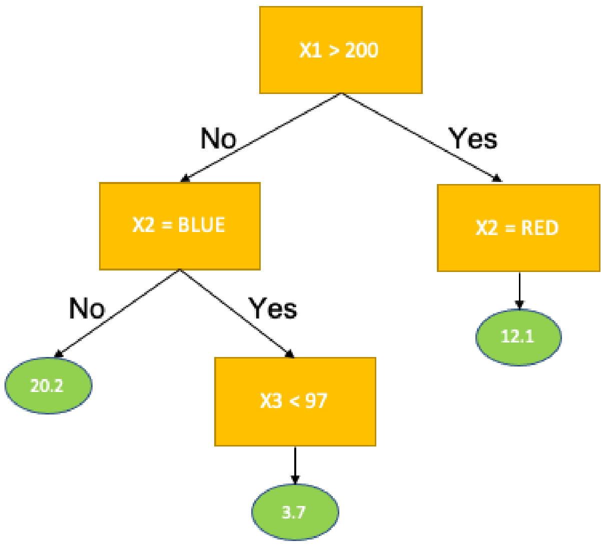


Figure 3.1: An example of a basic Decision Tree used for regression

Points Prediction for FPL

As mentioned earlier on, we believe that a player's form is one of the key factors that indicates whether a player is likely to score well in their next match. In addition to this we must also consider a player's intrinsic value. If a player costs a lot of money in the game, it's for a reason. It means prior to this they have demonstrated serious point scoring potential in the game. Whilst there are some bargains to be found, more often than not the highest scoring players are often the most expensive. Moreover, aside from price, another way to assess a players worth without form is to consider the points that they received last season, as this will be indicative of a good FPL pedigree. Thus, when looking for features to input to our model we would like ones that capture a players form, history, price and lastly information about their opposition. A player facing a weaker team most likely has a better chance of scoring more points. If the top scorer in the league is playing the current worst defence in the league, you would think that it's more likely they are going to garner you more points than if they were playing the best defence in the league for instance.

We therefore propose it would be ideal to construct a dataset of the following nature. For each gameweek, for each player (or training instance) we would like all ‘form features’ to hold a rolling sum over the last N gameweeks where N is small, say less than 5 or 6. This is because a players form can only last so long and we believe that a players most very recent performances are the most informative when considering how many points they will score in the next few weeks. An example of one of these form features would be say, the amount of points a player has scored in the last 4 gameweeks, or the amount of goals or assists they have gotten in their last 4 gameweeks. We would also like to include some sort of difficulty rating of the opponent that they are playing in the weeks following. The benefits of this approach is that we can have the ability to spot patterns in the data where a player looks to be hitting form, and would very much like them to be in our team selection. Another benefit of this approach, in comparison to the team based model is that we can capture much deeper lying player statistics outside of simply just goals scored and assists. Provided we can get access to this sort of data, you could also implement features of say how many shots has the player had in the last 4 gameweeks, something very much out of reach from team based models. This is beneficial due to the fact that these smaller factors could be indicative of player point scoring potential and provide much more realistic estimates than simulations can. The dataset would look something similar to Table 3.1.

	x values or input features					**y value or target**
player	player form features over last N gameweeks		opponent difficulty	player price	times selected	points scored in the next gameweek
1		...				
2						

Table 3.1: Example of the dataset we want to construct to train our random forest model

Something to note is that the ... represents the various form features that we would like to include such as shots on target in the last 4 matches. Once we have constructed a dataset similar to this we would proceed with the standard method of splitting it into training and test sets and therefore using our training set for our random forest model to learn from. We may also check the accuracy of our model by measuring the Mean Squared Error (MSE) on the test set.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Where Y_i is the actual value and $\hat{Y}_i$ is the predicted value. We then look to combine this approach in the next chapter, with a team formation optimisation technique to find for a specific gameweek, the optimal team to select.

4. Modelling FPL

In this chapter we will dig further into the approach of Matthews et al. [25], and we will introduce the MDP framework that captures the sequential team formation problem that is FPL. We first outline the basics of the model whilst also introducing the method to update the belief state as referred to in subsection 2.2.1. We then talk about how we go about choosing an optimal team through the means of a Multi-dimensional Knapsack Problem (MKP). We then talk about the method that we can use to approximately solve the MDP, Bayesian Q-learning, and explain why it is better for our use case than the generic Q-learning algorithm introduced in section 2.4. This procedure therefore unifying the 2-step problem that we outlined in the introduction, where we consider both points prediction and optimal team selection in unison for optimal decision making. Lastly, we will detail our own alternative approach the problem using our own individual player point forecasting method introduced in Chapter 3.

4.1 Conjugate Priors

So far we have been quite vague in what we mean by a belief state, this has been intentional due to the fact it is quite technical and that it makes more sense when presented in a more detailed context. The belief state for the FPL MDP is defined by a group of *conjugate priors.* A *prior distribution* within the realm of Bayesian inference is a distribution for some uncertain quantity before new evidence is taken into account, e.g. if we wanted to estimate some uncertain parameter and roughly knew the region of where it should be based on say, domain knowledge, then we can sample from such a distribution to produce an estimate for this parameter. Ultimately it expresses our beliefs about where this uncertain quantity should be. Now in FPL, with each new gameweek we are exposed to new information, each gameweek we become more informed about the players abilities. Thus, it would be smart to update distributions which represent player abilities gameweek to gameweek. In addition to this, as player ability is uncertain we would like to sample from these distributions to produce parameters for distributions which can outline probabilities for say, a player scoring a goal. We briefly mentioned this in Section 3.2, where a simple Bernoulli distribution could be used to outline the probability whether a player scored a goal in a match given one was scored. The uncertain parameter here would be p in a Bernoulli(p) distribution and can be sampled from the prior, creating an instantiation of this player ability distribution. The benefits of sampling from a prior is that it incorporates the uncertainty in a players output on the pitch. In some weeks a player can have a bad game and in other weeks they can have a good one, here we capture this variation where more consistent, better quality players have more optimistic shaped distributions and it is more likely that the parameters provided will give them more chance of scoring points in a theoretical sense. Upon receiving new information updating these priors to gain a *posterior distribution* is in general a very tough task. However, if the prior distribution and posterior are in the same probability distribution family, the update is done simply using closed-form equations (i.e. using simple, familiar mathematical operations and generally not with any integration or differentiation involved). If they are in the same probability family then they are said to be *conjugate distributions*. The prior is then called a conjugate prior for the associated likelihood function. We relate these with Bayes rule,

$$\underbrace{\Pr(\theta \mid x)}_{\text{posterior}} = \frac{\overbrace{\Pr(x \mid \theta)}^{\text{likelihood}}\overbrace{\Pr(\theta)}^{\text{prior}}}{\Pr(x)} \tag{4.1}$$

where θ is the parameter and x is the data. This can be used as a means for updating priors when new data is received to obtain the posterior. It turns out that if we use Bernoulli or Multinomial distributions to define player abilities then Beta and Dirichlet distributions are considered conjugate priors [14]. This will all come in handy in the next section, due to constraints there is only so much detail we can give on Bayesian statistics but this should be enough to proceed.

4.2 Belief MDP for Fantasy Premier League

We now outline the MDP and related elements introduced in the paper by Matthews et al. [25]. It is important to note that we are only interested in the main point scoring methods in FPL including goals scored, assists, clean sheets and minutes played.

Definition 4.1. The player abilites τ are defined by the following distributions:

- A three state categorical distribution ρ_p which takes values $\{start, sub, unused\}$. This outlines the probability of a player p making some sort of an appearance in a match.

- A Bernoulli distribution ω_p which outlines the probability of a player p scoring a goal.
- A Bernoulli distribution ψ_p which outlines the probability of a player assisting a goal.

Remark. Clean sheet points can directly be derived from the goals scored. If it's predicted that no goals are scored for a team, then points shall be awarded to the opposition's defenders and midfielders only.

These distributions will complement the FPL MDP, which can be defined by the following elements:

- The state space encapsulates for a gameweek i, the upcoming gameweek matches $\mathcal{M}_{i,\ldots,38}$, the set of available players $\mathcal{P}_i$, the outcome of the previous gameweek matches $o \in \mathcal{O}_{i-1}$ and the player abilities τ.
- The action set $\mathcal{A}_i$ for a gameweek i is simply the set of possible team combinations consisting of 15 players.
- The reward function $R(o, a_{i-1}, a)$ where $a \in A_i$ and $a_{i-1} \in \mathcal{A}_{i-1}$ is defined according to the FPL rules in Table 1.1.
- The state transition function is dependent on an unknown distribution $\Pr(o \mid \tau)$. It is unknown due to uncertainty in both the players abilities and the dynamics that effect match outcomes.

To circumvent this issue of not knowing the state transition function, they employ a reinforcement learning approach, operating under uncertainty, and learning a Markovian policy which maximises performance based on it's knowledge of the environment. The knowledge of the environment is therefore captured in a *belief model* which we touched on in the previous section. A group of distributions that we can both sample parameters from to derive player ability distributions held in τ whilst also allowing us to estimate reasonable dynamics for the state transition function.

Belief Model and Sampling Outcomes

Matthews et al. use Beta and Dirichlet conjugate priors to estimate the distributions held in τ,

- $\omega_p \sim \text{Beta}(0, 5)$,
- $\psi_p \sim \text{Beta}(0, 5)$,
- $\rho_p \sim \text{Dirichlet}\left(\frac{1}{4}, \frac{1}{4}, \frac{1}{2}\right)$

these priors are updated with each gameweek using Bayes rule (4.1). All players start with these distributions with the same set parameters unless there is already existing data on them at the start of the season and are adjusted accordingly. Sampling from these distributions allows us to gain instantiations of the player abilities τ. In addition, they add four Multinomial distributions which describe the distribution of minutes for players in each of the four positions in FPL {GK, DEF, MID, FOR}. These will determine how many minutes a player will play given they are in the starting lineup. A value of 70 sampled from these distributions would mean a player in that position has completed 70 minutes of a match and is thus awarded 2 points as per Table 1.1. Using the Dixon-Robinson model described in Section 3.2, one can simulate match outcomes and award points relative the distributions held in τ. To outline the procedure, we have our belief model b_i and we would like to sample from the distribution $\Pr(\tau \mid b_i)$. Combining this with the Dixon-Robinson model we can therefore sample match outcomes using the joint distribution $\Pr(\mathcal{O}_i \mid \tau)\Pr(\tau \mid b_i)$ this is since you can claim that match outcomes are somewhat dependent on the inherent player abilities. At the start of the match we sample according to $\Pr(\rho_p = start)$, to fill the starting lineups for each team and call these P_H and P_A. We also add some minor adjustments to the Dixon-Robinson model. We check in every minute if a player is scheduled to leave the pitch and are replaced with another player according to ρ, remember that all subs are scheduled at the beginning due to the sampling of τ, i.e. their minutes if they start a match are already determined. A replacement is then selected in proportion to $\Pr(\rho_p = sub)$ and this player is swapped out to the unused player set U_H or U_A. Lastly, something that we briefly touched on in Section 3.2 if a match is simulated to be `Arsenal 3 - 0 Newcastle` then we shall award a player in the Arsenal starting lineup P_H for each goal scored in proportion to the sampled Bernoulli distribution ω_p i.e. $\Pr(\omega_p = 1)$ with assists and clean sheets being awarded in a similar way.

Belief MDP for FPL

Now we have all of the elements to define the Belief MDP for FPL. As already stated we maintain prior distributions over the player abilities held in τ, using the belief model. The belief state b_i is thus an instantiation of the belief model, updated with all observations in the last gameweek. The state can be updated from b_i to b_{i+1} according to Bayes rule in (4.1). Then a manager can perform optimally based on it's current beliefs about the players abilities, and maximising the Bellman equations, this time in a *continuous* space as we are dealing with a belief MDP.

$$V(b_i) = \max_{a \in \mathcal{A}_i} Q(b_i, a) \tag{4.2}$$

$$Q(b_i, a) = \int_{\tau} \Pr(\tau \mid b_i) \int_{o \in \mathcal{O}_i} \Pr(o \mid \tau)[r_i + \gamma V_i(b_{i+1})] \, do \, d\tau \tag{4.3}$$

where $\gamma \in [0, 1)$ is a discount factor which dictates how much long term rewards are valued and r_i is the immediate reward taken from the reward function $R(o, a_{i-1}, a)$. We refer to (4.3) as the long term discounted reward performing a, the *Q-value.*

This is now looking similar to the content we covered in Chapter 2. In summary we can perform optimally by going through the following procedure for each gameweek i:

1. Receive the observation tuple $\langle \mathcal{P}_i, \mathcal{M}_i, o \in \mathcal{O}_{i-1}, a_{i-1} \rangle$
2. Update the belief state b_{i-1} to get b_i and $\Pr(\tau \mid b_i)$ using Bayes rule (4.1)
3. Choose an action $a \in \mathcal{A}$ that maximises the Bellman equation (4.3).

Exact solutions to the Bellman equations in this case are intractable due to size of the action set $|\mathcal{A}_i|$ and the outcome set $|\mathcal{O}_i|$. Matthews et al. alleviate the pains of the size of both with the sampling procedure defined in the last subsection. Where we essentially simulate the environment based on our belief state, such that we only consider the simulated scorelines and not the entire set. This also extends to actions, as if we can simulate matches then we know which players will score the most, and thus can develop a good selection of actions. The pain of estimating Q-values over 38 gameweeks is also a problem but can be alleviated by setting a maximum recursion depth. A depth of $d = 3$ seemed to work well in their paper but they still faced some complexity issues in practice given that they set time constraints for the algorithms to run, and any depth $d \geq 3$ was most likely stopped before it could converge to the correct values, hence they saw no performance benefit with increased depths. A mystery is whether increased depths could in fact prove themselves to be valuable from a point scoring perspective.

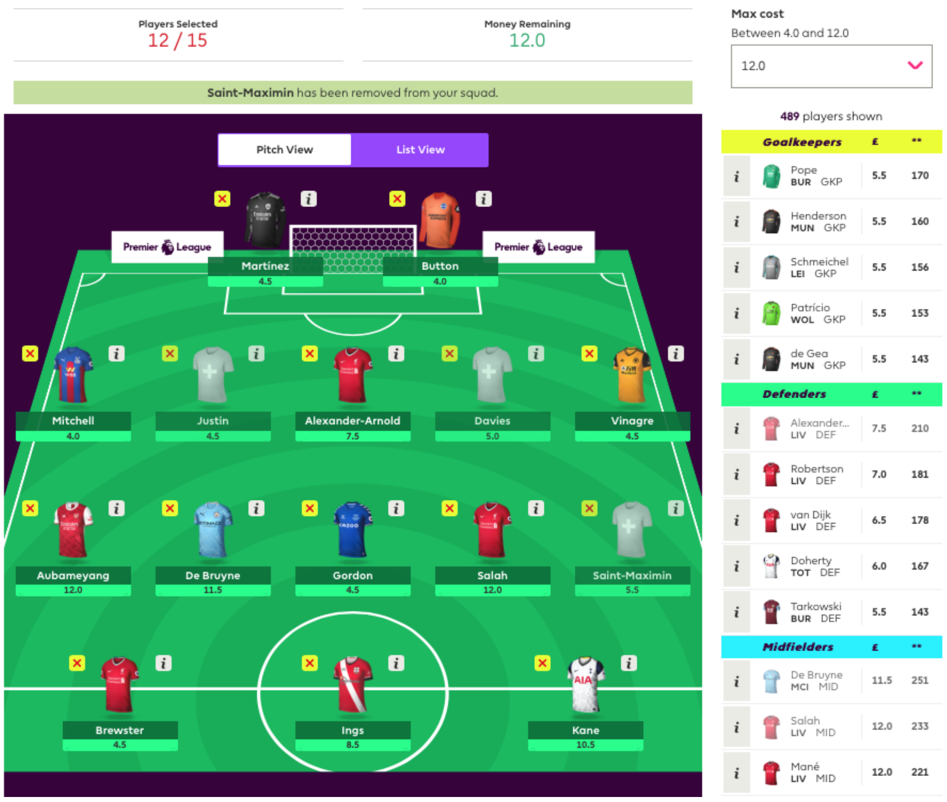
4.3 Team Formation Optimisation

Team formation is the last piece of the puzzle. If we have simulated the environment for a gameweek based on our beliefs, then we now should have the expected points for players in that gameweek. Now all is left to do is fill the team with players expected to score the most points. An efficient way to do this is to frame the team selection problem as the following Multidimensional knapsack problem (MKP) [19]:

$$\begin{aligned} \text{maximise} \quad & \sum_{i=1}^{n} v_i \, x_i, \\ \text{subject to} \quad & \sum_{i=1}^{n} w_{i,j} \, x_i \leq c_j, \quad j = 1, \ldots, m, \\ & x_i \in \{0, 1\}, \qquad i = 1, \ldots, n. \end{aligned}$$

Where for a player i, v_i is the expected number of points, $w_{i,j}$ are the costs where the associated player costs cannot exceed the capacities c_j and x_i simply acts as an indicator to mean that we have chosen one of the n players to be in our selection. We quickly remind ourselves of the selection capacities for FPL.

- Exactly 15 players must be selected consisting of 2 goalkeepers, 5 defenders, 5 midfielders and 3 strikers.
- The total budget is £100 million.
- You are only allowed to select a maximum of 3 players from each Premier League club.
- You are only allowed 1 'free' transfer per gameweek. Each additional transfer is penalised with -4 points. You can also choose to not make any transfers and receive 2 free transfers the next week. This process is capped at 2 however, and the next week you will still have only 2 free transfers if you decide to make no changes to your team again.



Players Selected 12 / 15

Money Remaining 12.0

Saint-Maximin has been removed from your squad.

Max cost

Between 4.0 and 12.0

12.0

489 players shown

Goalkeepers	£	**
Pope BUR GKP	5.5	170
Henderson MUN GKP	5.5	160
Schmeichel LEI GKP	5.5	156
Patrício WOL GKP	5.5	153
de Gea MUN GKP	5.5	143
Defenders	**£**	******
Alexander... LIV DEF	7.5	210
Robertson LIV DEF	7.0	181
van Dijk LIV DEF	6.5	178
Doherty TOT DEF	6.0	167
Tarkowski BUR DEF	5.5	143
Midfielders	**£**	******
De Bruyne MCI MID	11.5	251
Salah LIV MID	12.0	233
Mané LIV MID	12.0	221

Figure 4.1: Screenshot taken of the team selection page on the FPL website

To put this in words, our aim is maximise the expected points of the team by filling it with the best combination of highest scoring players such that this combination does not violate the team selection constraints. Once the team has been selected with a mathematical programming solver like IBM ILOG's CPLEX[1] the *starting lineup*

[1]`https://www.ibm.com/uk-en/analytics/cplex-optimizer`

can be filled by greedily selecting 11 of the 15 players whom have the most expected points. Matthews et al. generate good quality actions for their MDP by generating multiple groups of samples and averaging over each of those from the Dixon-Robinson model, therefore generating a number of candidate teams for that gameweek. The generated team selections are only myopically optimal and are not necessary the best over multiple gameweeks. How might we quantify the long-term effects of making certain decisions? We will now address this in the next section.

4.4 Bayesian Q-learning

Matthews et al. employ multiple techniques to consider the payoffs of each team selection or action including Depth First Search (DFS) [35], Q-learning and finally Bayesian Q-learning. DFS whilst it seemed to yield good results had issues with complexity, as the algorithm has time complexity $O(n^d)$ and seemed to struggle to make team selection decisions even with a search depth of $d = 3$ with 40 candidate teams generated took 40 minutes per decision. They also tried generic Q-learning with slight modifications to accommodate the Belief MDP, this circumvents the complexity issues as the algorithm has time complexity is $O(\eta d)$, where η is the number of episodes or play-outs.

Algorithm 2: Q-learning for the FPL Belief MDP

```
for each episode e do
    Initialise the belief state $b = b_0$
    for each gameweek up to some depth d do
        Choose an action generated by the grouped samples and MKP output
        Sample outcome $o$ for that gameweek from the belief state
        Receive reward $r$ for team selection $a$
        $\hat{Q}(b, a) \leftarrow \hat{Q}(b, a) + \alpha[r + \gamma \max_a \hat{Q}(b', a) - \hat{Q}(b, a)]$
        Update belief state $b$ using Bayes rule and the outcome $o$
    end
end
return $\arg\max_a [\hat{Q}(b_0, a)]$
```

This approach suffered from slow Q-value convergence because the agent did not explore the action space very well. It was possible for the agent to explore outcomes even if it provided the agent with no new information, or for promising actions to be downplayed by ‘unlucky’ outcome sampling. So how did they remedy these shortcomings?

The variant of Q-learning proposed by Dearden et al. [9] proved to be useful in the paper by Matthews et al., and managed to achieve the highest FPL rank among all methods that they tried. This variant is called Bayesian Q-learning (BQL). BQL manages uncertainty in Q-values for each state-action pair by representing them as a normal distribution with unknown mean μ and unknown precision $\tau = 1/\sigma^2$ (reciprocal of the variance). For each state-action pair, since Q-values are defined as the expected total discounted reward we have $Q(s, a) = \mu$. These are updated as rewards arrive, this time using a normal-gamma conjugate prior. The normal-gamma distribution is the conjugate prior of a normal distribution, and we can say that

$$(\mu, \tau) \sim \text{NormalGamma}(m, \lambda, \alpha, \beta)$$

and the posterior can be updated to

$$(\mu, \tau \mid x_1, \ldots, x_n) \sim \text{NormalGamma}(m', \lambda', \alpha', \beta'),$$

where

$$M_1 = \frac{1}{n}\sum_i x_i, \; M_2 = \frac{1}{n}\sum_i x_i^2$$

$$m' = \frac{\lambda m + nM_1}{\lambda + n}, \; \lambda' = \lambda + n, \; \alpha' = \alpha + \frac{1}{2}n \text{ and}$$

$$\beta' = \beta + \frac{1}{2}n(M_2 - M_1^2) + \frac{n\lambda(M_1 - m)^2}{2(\lambda + n)}. \tag{4.4}$$

It works by guiding the agent in directions that have a larger *exploration bonus.* This exploration bonus is defined as something called *Value of Perfect Information* (VPI), a concept originating in the paper by Howard (1966) [16]. VPI allows us to quantify the value in how much information is gained in terms of learning an action a in a belief state b's true Q-value, q_a^*, with our knowledge of $V(b)$ in (4.2). Imagine ranking all of the current best guesses at Q-value estimates in a list where the highest value is at the top of the list. If we perform any action in our current belief state we only gain new information if some other action displaces the current best Q-value. This is because we always want to choose the single most optimal action and we disregard the rest, hence if nothing displaces the current top estimate we have gained no new information. The same can be said about performing it, if it maintains it's position as the best current action then we haven't gained anything. On the other hand, if it drops down in the pecking order we have gained something. We can define this formally by a concept called *gain*, G. We refer to the best action as a_1, the second best as a_2 and so on.

$$G_a(q_a^*) = \begin{cases} \max\left[\hat{q}_{a_2} - q_a^*, 0\right] & \text{if } a = a_1 \\ \max\left[\hat{q}_a^* - \hat{q}_{a_1}, 0\right] & \text{if } a \neq a_1 \end{cases} \tag{4.5}$$

We then define VPI as the expected gain from performing an action a:

$$VPI(a) = \int_{-\infty}^{\infty} G_a(x) \Pr(q_a^* = x)\, dx \tag{4.6}$$

VPI may be calculated in closed-form by following the method of Teacy et al. (2012) [36].

Theorem 4.2. *Given that a_1 and a_2 are the current best actions an agent has according to it's current beliefs with associated rewards r_{a_1} and r_{a_2}, and that the action whose reward is normally distributed with unknown parameters $(\mu, \tau) \sim NormalGamma(m', \lambda', \alpha', \beta')$. The closed form VPI for choosing a is*

$$VPI(a) = \begin{cases} (r_{a_2} - m)\Pr(\mu \mid \mu < r_{a_2}) + \mathcal{B}_p(r_{a_2}) & \textit{for } a = a_1, \\ (m - r_{a_1})\Pr(\mu \mid \mu > r_{a_1}) + \mathcal{B}_p(r_{a_1}) & \textit{otherwise,} \end{cases}$$

where $\mathcal{B}(x) = \frac{\Gamma(\alpha-\frac{1}{2})\sqrt{\beta}\left(1+\frac{\lambda(x-m)^2}{2\beta}\right)^{-\alpha+\frac{1}{2}}}{\Gamma(\alpha)\Gamma(\frac{1}{2})\sqrt{2\lambda}}$ *is called the truncated bias function,*

and Γ represents the gamma function.

For more information about the Gamma function Γ please see the paper by Davis [8]. We can therefore modify Algorithm 2 by changing the action selection step to incorporate VPI such that we now return the highest value of $\hat{Q}(b, a) + VPI(a)$ and change the Q-update step to also incorporate updating the prior as in (4.4). Matthews et al. initialise their normal-gamma parameters as $\alpha = 1, \gamma = 1, \beta = \theta^2 M_2$ where θ is some proportion chosen to trade-off between over-exploration and neglect newly-generated actions. μ is chosen to equal a sampled approximation of the reward obtained by performing the action unchanged up to search depth. They choose to operate on a subset of the action set with only just 3 actions generated per round. This essentially concludes their approach and various experiments. We now move on to the theory of our own approach to the problem.

4.5 An Alternative Approach

Picking up where we left off in Chapter 3, we proposed the method of using a Random Forest regression model [22] to forecast a players points for the next gameweek. The difference to the approach that we take and the approach of Landers and Duperrouzel [21] is that they are choosing one single new team per gameweek. In FPL we must maintain a squad over 38 gameweeks and thus must consider the long term effect of our actions, as there are rules associated with transferring players in and out of teams. Landers and Duperrouzel did not have to factor this into their decisions, and could just choose a new team each week rather than having to obey any set of constraints as to how they can modify their team selection from the previous week. We could choose to act only myopically, however in the paper by Matthews et al. [25] they showed that using BQL or even DFS, where they considered the long term effect of their team selections yielded superior results.

We deal with the problem as follows. Once we have fitted our Random Forest model to some large amount of test data, we are in a position where we can now forecast a players points for a given gameweek. Since we want to do better than acting just myopically, we adapt the MKP approach in the following way. We generate our forecasts for the upcoming gameweek, we feed this data into our MKP and let it solve for the optimal team that gameweek. Here, we do not have the luxury of multiple generated actions or team selections from sampling procedures used in the Belief MDP approach. Therefore to combat this after we have generated an optimal team for that gameweek, to generate actions we start removing players from the item set that were in the optimal team for that gameweek. Each time we remove a player we reinstall the one we removed last time. The MKP will then generate different teams due to the fact the item set is different. This is a simple heuristic for generating actions, once we have generated n teams or actions for one gameweek we can then fast forward to the next gameweek and run the forecasting procedure again. At this stage you have two options, you can either decide to repeat the removal process to generate more actions per round, or you can proceed from this point onwards, with the first optimal team that is given as the optimal solution of the MKP when *all* players are available, i.e. skipping the removal process. The latter would constitute to a brute force approach. The former has time complexity of $O(n^d)$ where d is the depth or the number of gameweeks that you are willing to look ahead. Obviously the latter is not as optimal due to the fact that it does not explore all options, however it might prove to be a decent compromise if we are unhappy with the length of time it takes for the algorithm to make a single decision, as it produces a valid trajectory through the action space.

A small caveat of this procedure is how we might go about selecting the team for the first gameweek due to the fact there is no prior gameweek data to base our predictions on. One way to pick a starting team would be to solve the MKP based on last seasons points totals. The logic to this is that players who scored well last season and are available this season, provide the safest of bets in terms of team selection choices when we have no information about this season. Some players might be unavailable this season as they may have transferred to another league (or have been relegated to a lower league), so we should be careful not to include these players. We might also argue that this way alienates new players to league, but in terms of risk, it would be beneficial to wait and see how these new players perform before including them in our team selection.

In addition to this, we also must factor in captaincy selection and our chip playing strategy. We choose to give the captaincy to the player forecasted to score the most points in the next gameweek and the vice-captaincy to second highest. There are many ways in which we can choose to play our chips, a simple way of doing for computational convenience would be to randomly select the weeks that they will be played before the start of the season. As it's not the aim of the dissertation to explore optimal chip playing strategies, we will follow this approach. It's important to remember that only 1 chip can be played in a single gameweek and that only 1 wildcard can be played in the first half of the season and the other one can be played in the second half.

We can summarise our approach in the following steps:

1. Generate the GW1 team based on last seasons points totals using the MKP and randomly select the weeks that you will play the chips.
2. For each successive GW:
   - Forecast points for the GW from the Random Forest model.
   - Solve the MKP initially for all players to get an optimal team.
   - Start the removal process to generate n teams.
   - Select the captain and vice captains based on who are the top 2 expected point scorers.
   - Record the teams total forecasted points in some external array structure such that we can decide the optimal decision once the procedure is over.
3. For each team generated, either repeat step 2 up to some search depth, and look ahead to the next gameweek. Or take the brute force approach and explore all valid trajectories from the initial team generations. Stop once you have looked ahead throughout all trajectories (DFS).

We note that this all designed with computational experiments in mind. If we were trying to use this method to play the actual game, a human player could intervene with certain aspects of the decision making process. For instance, a human player could choose when to play their own chips, as opposed to the computer randomly selecting them before the simulation process starts. It is most likely possible to design some sort of system that can say for example, select an optimal chip playing strategy but this is not the focus here. Unfortunately it is not possible for us to evaluate the full alternative approach due to the lack of data on ranks. However, in the next chapter we will evaluate our points forecasting model and implement it along with the MKP with some example code.

5. Computational Experiments

In this penultimate Chapter we shall evaluate our proposed new model. We implemented this using Python 3[1] and specifically in a Jupyter Notebook. It is recommended that you install it via the Anaconda Distribution[2] as it will include many of the core packages that we will use to implement the model, as well as Jupyter. We performed the bulk of the data cleaning and wrangling using the Pandas Python library [40], and implemented the Multidimensional Knapsack Problem using the PuLP Python library [27]. The code is available in HTML format, which is better for strictly viewing purposes, and the original Jupyter Notebook `.ipynb` file is provided also. The data that we used for the experiments were originally scraped from the FPL website by GitHub user `vaastav`[3].

We now look to explore and evaluate the Random Forest point forecasting model. As pointed out before we were looking for various form features to include in our dataset. Table 5.1 summarise the form features that we managed to acquire. We would have liked to have had many more that capture the players underlying statistics, but this is what we will work with for the remainder of the dissertation. Data of this sort is hard to obtain, especially for free, and we will discuss this matter further shortly.

To clarify, the features marked with an asterisk are FPL's own measurements of performance[4]. We describe roughly what they are in Table 5.1, but FPL do not reveal the formulas that they use to come up with such numbers. Nonetheless, we shall include them in our model due to the meagre amount of information available, and we look to evaluate their effectiveness in the points prediction process. As mentioned in Section 3.3 we wanted to calculate a rolling sum of these features over the past N weeks, and fit these values to the points that the player scored in

[1] `https://www.python.org/`

[2] `https://www.anaconda.com/`

[3] `https://github.com/vaastav/Fantasy-Premier-League`

[4] `https://www.premierleague.com/news/65567`

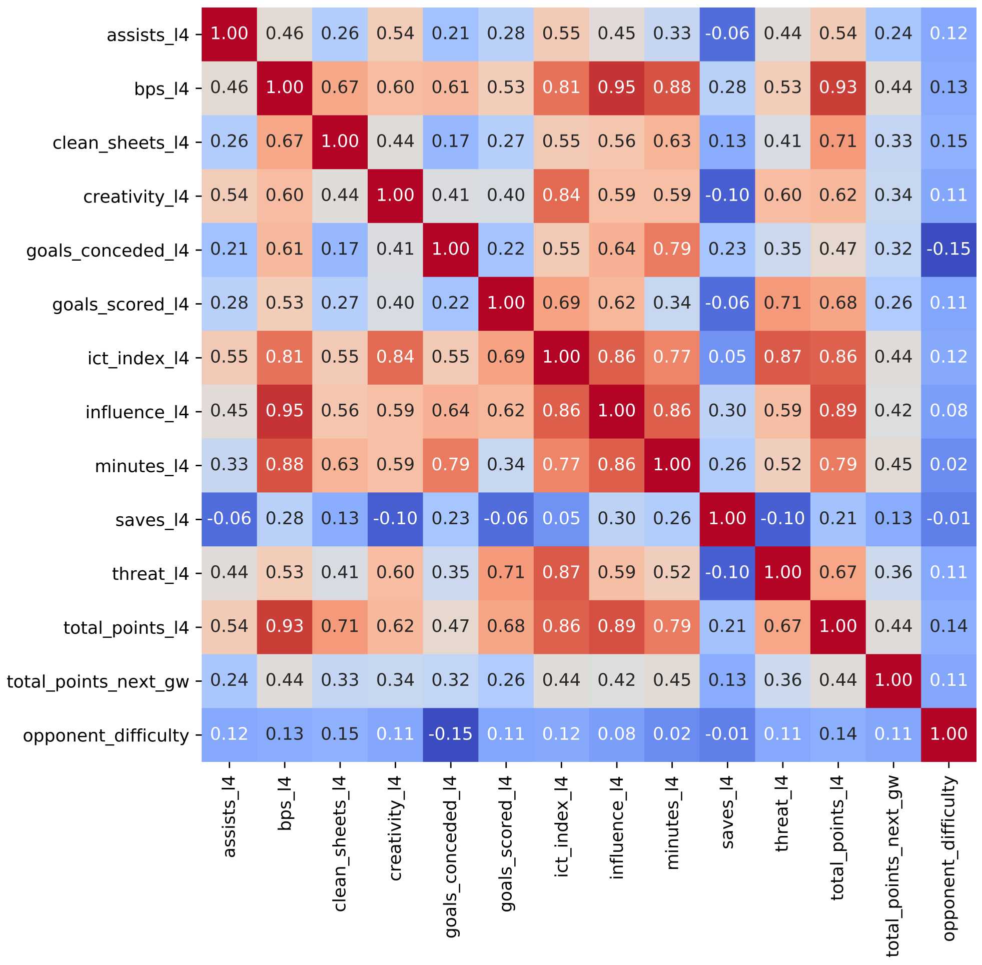
Feature Name	**Description**
Assists	Number of assists
BPS	Number of points in bonus point scoring system
Clean Sheets	Number of clean sheets
Creativity*	An FPL score of how creative the player was
Goals Conceded	Number of goals conceded
Goals Scored	Number of goals scored
ICT Index*	An FPL score of Influence, Creativity and Threat
Influence*	An FPL score of how influential the player was
Minutes	Number of minutes played
Saves	Number of saves
Selected	Number of times selected that gameweek by players
Threat*	An FPL score of how threatening the player was
Total Points	Number of points scored

Table 5.1: Form Features

the following gameweek. Additionally we will add the features of 'was home', 'last season points' and 'opponent difficulty' to help improve the fit, therefore satisfying the requirements of an ideal dataset outlined in Section 3.3.

First we perform some simple preliminary data visualisation to try and get some insight into our dataset. Observe Figure 5.1, the warmer the colour then the higher the correlation between features. Some quick things to point out is that we can see the FPL performance measurements of creativity, threat, and influence are highly correlated with their aggregated feature in the ICT index, as to be expected. Despite this, each one might have it's own merits and due to the meagre amount of data we have access to, we shall retain them all to see how influential each one is in our model. Their definitions on the FPL website are also ambiguous so it might be interesting to see if any single one may pick up on underlying player performance. In addition to this, we would have liked to have seen some stronger correlations with the points scored next gameweek, however a correlation of around 0.45 for BPS, ICT Index, Total Points and Minutes, we expect these to be the most decisive in the Random Forest model. To be clear these correlations were made using Pearson's Moment Correlation Coefficient (PMCC).

Next we look at which value of N turned out to have the lowest value of MSE. Observe Figure 5.2. The model which produces the lowest value of the MSE on the test set, is the model with the best prediction accuracy. The values were all quite close to each other overall, however $N = 2$ did not seem to fare as well. $N = 5$ had the lowest MSE and generally the MSE tended to drop off slightly with increased



	assists_l4	bps_l4	clean_sheets_l4	creativity_l4	goals_conceded_l4	goals_scored_l4	ict_index_l4	influence_l4	minutes_l4	saves_l4	threat_l4	total_points_l4	total_points_next_gw	opponent_difficulty
assists_l4	1.00	0.46	0.26	0.54	0.21	0.28	0.55	0.45	0.33	-0.06	0.44	0.54	0.24	0.12
bps_l4	0.46	1.00	0.67	0.60	0.61	0.53	0.81	0.95	0.88	0.28	0.53	0.93	0.44	0.13
clean_sheets_l4	0.26	0.67	1.00	0.44	0.17	0.27	0.55	0.56	0.63	0.13	0.41	0.71	0.33	0.15
creativity_l4	0.54	0.60	0.44	1.00	0.41	0.40	0.84	0.59	0.59	-0.10	0.60	0.62	0.34	0.11
goals_conceded_l4	0.21	0.61	0.17	0.41	1.00	0.22	0.55	0.64	0.79	0.23	0.35	0.47	0.32	-0.15
goals_scored_l4	0.28	0.53	0.27	0.40	0.22	1.00	0.69	0.62	0.34	-0.06	0.71	0.68	0.26	0.11
ict_index_l4	0.55	0.81	0.55	0.84	0.55	0.69	1.00	0.86	0.77	0.05	0.87	0.86	0.44	0.12
influence_l4	0.45	0.95	0.56	0.59	0.64	0.62	0.86	1.00	0.86	0.30	0.59	0.89	0.42	0.08
minutes_l4	0.33	0.88	0.63	0.59	0.79	0.34	0.77	0.86	1.00	0.26	0.52	0.79	0.45	0.02
saves_l4	-0.06	0.28	0.13	-0.10	0.23	-0.06	0.05	0.30	0.26	1.00	-0.10	0.21	0.13	-0.01
threat_l4	0.44	0.53	0.41	0.60	0.35	0.71	0.87	0.59	0.52	-0.10	1.00	0.67	0.36	0.11
total_points_l4	0.54	0.93	0.71	0.62	0.47	0.68	0.86	0.89	0.79	0.21	0.67	1.00	0.44	0.14
total_points_next_gw	0.24	0.44	0.33	0.34	0.32	0.26	0.44	0.42	0.45	0.13	0.36	0.44	1.00	0.11
opponent_difficulty	0.12	0.13	0.15	0.11	-0.15	0.11	0.12	0.08	0.02	-0.01	0.11	0.14	0.11	1.00

Figure 5.1: Correlations between different features in our dataset

values of N. Since the values are so close, it's not completely fair to say that any one model is better than the other, however it may be possible to claim that $N = 5$ is most likely a slightly better fit than $N = 2$ say.

A slight drawback of this approach, which could also have negative effects on the MSE values, is that in the earliest gameweeks such as GW2, GW3 etc. we have less data on players, and the error in the predictions in these earlier weeks is likely to be higher. A suggested fix for this would be to potentially have a separate model for the first N gameweeks whilst data is collected for those first N weeks, and then rollout our proposed model once you have the appropriate data gathered. This separate model would be more focused on the inherent pedigree of players, based on their previous season performances. Example features in this model would be say, last seasons points total or the number of goals they have scored in the past few seasons. Any measure that would indicate some sort of consistency of the player would be beneficial. After performing cross validation [6] for a chosen value of N we got similar scores for all folds, meaning that we can be safe in saying that our model was not

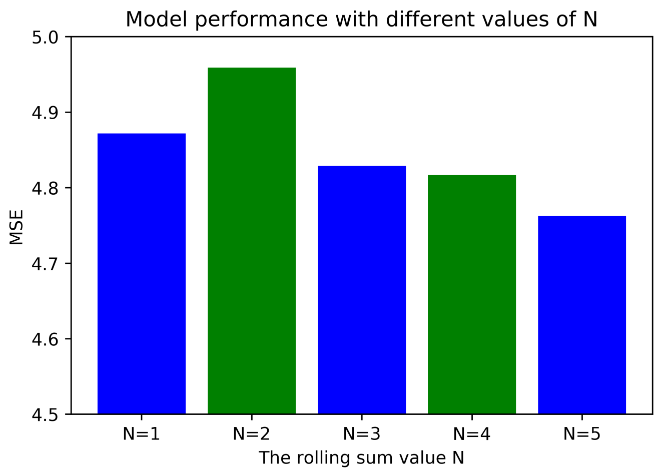


Figure 5.2: The MSE with different values of N

overfitting in any major way. The accuracy of the MSE was also similar for both test and training sets.

Furthermore, we now will talk about how important certain features are to the model. After graphing for various values of N, all graphs seemed to value the features very similarly. In Figure 5.2 we show the relative feature importances for $N = 4$. Interestingly Minutes was the feature that topped the chart. After some thought this makes quite a lot of sense given the dataset that we have used. Many players in the dataset could be deemed as 'dead players' that never get any minutes on the pitch or maybe only a small amount. If players are not playing then there is no chance that they will score any points. Recall that in FPL just by playing over 65 minutes a player will receive 2pts. It is likely that the model learning phase latched on to the fact that if a player has played 0 minutes in their last N games, it is likely they will score 0 points and unlikely they will score any more than that. Hence it should come at no surprise that minutes was an important factor for points potential. In second place on the chart we have the ICT Index. This is comforting that it's placed so highly as it validates our hypothesis about underlying player statistics being of use when predicting player points. The ICT index represents a players output on the pitch. Is the player making dangerous passes that lead to goals? Is the player scoring every chance that he gets? Is he always in a dangerous goal scoring area? This is exactly what this statistic is getting at. If we could somehow get access to data which held records of statistics such as number of key passes in the game or number of touches in the opponents penalty area (for each player), then we hypothesise that these would also be useful for predictive performance. Two other non-surprising important features were total points in the last four matches, and the number of times that a player was selected by FPL players in the game. This essentially is saying that players who are popular will most likely score more than players who

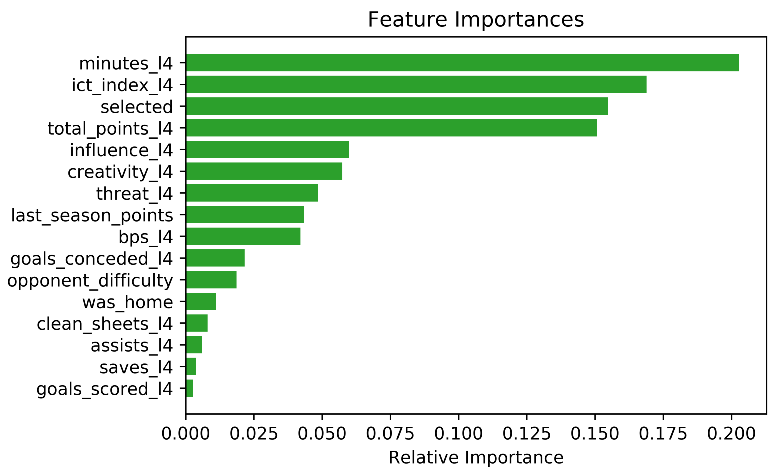


Figure 5.3: Feature importances from the Random Forest model with rolling sum of $N = 4$ for form features

are not selected that much, because why would so many people be selecting a bad player? Wisdom of the crowd comes to mind here. It highlights the fact that when picking players you should always consider their popularity, as if you are not selecting these types of players then you are likely to be missing out on points. A players total points in the last four matches is another straightforward explanation, if a player has been scoring a lot of points recently then it is more than likely that they will continue to do so. The converse can be said about players who are not scoring well in recent matches.

At the bottom of the pile we have most surprisingly goals scored, assists and clean sheets! This is contradictory to our earlier hypotheses. However, we suspect that this is also due to the nature of the data we were using. Considering the fact that were many 'dead players' in the dataset and as well as this, goals and assists are not all that common, e.g. the top goal scorer in the 2019-20 season managed only 23 goals (meaning they would have blanked in 15 different gameweeks). In summary due to the sporadic and infrequent nature of players scoring or assisting goals it is likely that the model never picked up on this as a good signal for points potential. This could be potentially remedied by converting goals scored in the last N to a different metric, say perhaps a rolling sum of all goals or assists that they have scored in *all* games thus far. This could be better for indicating player pedigree and thus would be better for points prediction. Another remedy would be to remove the dead players but this does go against the spirit of a truly 'automated manager'.

In reflection it has become apparent that it may be more useful to have separate models for different player positions, i.e. a predictive model for each of goalkeepers, defenders, midfielders and forwards. This is due to the nature of the different way each player collects their points. Strikers will get points more for scoring goals, so we really should be looking at their attacking output mainly, whilst goalkeepers are only collecting points from clean sheets and saves. It's also possible to take this even further and create separate models for each player, though this is slightly unrealistic at present time as you would need a lot data to create anything meaningful, something which we do not have access to at present time.

Issues with Data

Something we had to be careful of was that due to the COVID-19 pandemic the season was suspended from GW30 to GW38 and FPL have decided to store this data under the labels of GW39 to GW47. Aside from the technical details, this most likely would have adverse effects on our model for point forecasting as it is a form based model. The break was around 3 months long and it most likely had the same effect of starting a new season for players. This would have most definitely disrupted their form, fitness levels and most likely impaired them from performing at their absolute best when matches resumed in June. For direct comparison, the break was even longer than the average players usual summer break. We suspect this could invalidate the approach. An alternative would have been to run experiments on previous seasons that had no disruption but due to gaps in the dataset this was not at all possible. The only consistent data that was available for our experimental needs was for the 2018-19 and 2019-20 seasons. The 2018-19 season was the one that we used to train our model on, and it is now obvious what kind of predicament this leaves us in, 2019-20 is the only remaining option.

Another frustrating problem that we encountered is that the FPL website wipe their leaderboard and rank data each season. There are no datasets or archives online that keep track of this data so therefore it is not possible to see how well the model ranked over the 2019-20 season. Rank data is crucial to measurements. For instance, if we had two automated managers going against each other, if one scored 2080 points and the other scored 2100 (quite close in points) and their ranks were 230,000 and 60,000 respectively. Then it is clear that these extra 20 points that the second manager had proved to be a lot more impressive in that rank band, i.e. those 20 points made a lot of difference. Then pose another scenario where two managers scored 2200 and 2220 (same gap in points) but this time their ranks were 30,000 and 29,000 and that point gap becomes slightly less impressive. Despite this, we still implemented some adaptable exemplar code in the Jupyter Notebook for the alternative approach for an interested party to try provided they had rank data. Which includes the MKP formulation for FPL and a simulation over gameweeks.

6. Conclusion

In this dissertation we have presented multiple approaches for modelling football matches such as the Dixon-Coles model, and we explored their applications and usages for Fantasy Football. Additionally, we provided an in depth look into the current benchmark for an automated FPL manager, that was presented in the paper by Matthews et al. [25]. Throughout this paper there were various techniques and models that were presented deemed to be assumed knowledge of the reader. We have provided informative explanations of these areas. Examples of these areas being the Dixon-Robinson model and the various Bayesian inference techniques that they used. Though most importantly we presented a focused but thorough introduction to reinforcement learning such that reader would be well equipped enough for further study in this area. We explored abstract model formulations such as the Belief MDP and looked at some algorithms to solve them, these being the Q-learning algorithms. Further to this, we presented our own alternative approach to the problem of predicting player performance using a Random Forest model, and combined this with the Multidimensional Knapsack Problem to select an optimal team.

In addition, we also evaluated our player performance model and provided suggestions on how it might be improved upon. We looked to see if there were any truths to our earlier hypotheses and concluded that some were right, namely that *recent* underlying player statistics are somewhat useful for predicting performance. It was also found that these models do not find recent goals and assists very useful for prediction, something of which was very surprising. Though we believe that this is likely due to the dataset and model training mechanics. Although variations of these metrics may prove to be useful in actuality. As detailed in Chapter 5 there is a severe lack of quality free data resources for football. Whilst there is some basic data available online for free, any kind of rich data would require an expensive license from a company such as Opta[1]. We suspect this is due do the vast amount of

[1]`https://www.optasports.com/`

money involved or to be made with access to such data and we do not expect it to be publicly available any time soon. Finding the right data is of the upmost importance for supervised learning methods and whilst datasets are not widely available, then perhaps simulation methods for performance prediction are currently the best way forward in terms of performance prediction.

In terms of areas for future work there are numerous avenues that an interested reader could take. In terms of football modelling and predicting performance, one could look at ways of extending the Dixon-Robinson model and try to incorporate different hypothesised match effecting factors and see if it improves prediction accuracy. An example of this would be putting in parameters which can represent the predicted weather conditions, as weather conditions possibly could have an effect on the outcome of a match. The interested reader could also look at other supervised learning models for points prediction and assess whether or not they can improve the fit or FPL performance. Provided access to the right dataset it would be interesting to see how something like a Neural Network or a Gradient Boosted Machine measures up to the Random Forest model. Additionally, considering the fast moving pace of the reinforcement learning field it could be worth keeping an eye on the new literature coming out for some potential new algorithms or models that would be suitable for modelling FPL.

Outside of the world of Fantasy Football, the interested reader could take a look at the applications of newly found football models in betting markets. This was the original use case for the Dixon-Coles model and a good model may have a chance of yielding a profitable betting strategy, especially in the newer popular betting frameworks such as betting exchanges. Lastly, as the usage of analytics in real world sports is becoming ever more prevalent, it could also be worth looking at a models applications for squad management. For example, if a team has a little chance of winning an upcoming game (as per the model) and in close proximity there is another game that is very important, it may well be worth resting key players for such a game. This will ensure that they are fresh for the following important game that they have more chance of winning, and they have not lost much by potentially forfeiting the other less important match.

Bibliography

[1] Rahul Baboota and Harleen Kaur. Predictive analysis and modelling football results using machine learning approach for English Premier League. *International Journal of Forecasting*, 35(2):741–755, 2019.

[2] Bogdan Batrinca and Philip C Treleaven. Social media analytics: a survey of techniques, tools and platforms. *Ai & Society*, 30(1):89–116, 2015.

[3] Ryan Beal, Timothy J Norman, and Sarvapali D Ramchurn. Artificial intelligence for team sports: a survey. *The Knowledge Engineering Review*, 34, 2019.

[4] Richard Bellman. A Markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

[5] Åke Björck. *Numerical methods for least squares problems.* SIAM, 1996.

[6] Michael W Browne. Cross-validation methods. *Journal of mathematical psychology*, 44(1):108–132, 2000.

[7] Georgios Chalkiadakis and Craig Boutilier. Sequentially optimal repeated coalition formation under uncertainty. *Autonomous Agents and Multi-Agent Systems*, 24(3):441–484, 2012.

[8] Philip J Davis. Leonhard euler's integral: A historical profile of the gamma function: In memoriam: Milton abramowitz. *The American Mathematical Monthly*, 66(10):849–869, 1959.

[9] Richard Dearden, Nir Friedman, and Stuart Russell. Bayesian Q-learning. In *AAAI/IAAI*, pages 761–768, 1998.

[10] Mark Dixon and Michael Robinson. A birth process model for association football matches. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 47(3):523–538, 1998.

[11] Mark J Dixon and Stuart G Coles. Modelling association football scores and inefficiencies in the football betting market. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 46(2):265–280, 1997.

[12] Ffmag. The Big Bang: Where Did Fantasy Premier League Begin. `https://fantasyfootballmag.com/where-did-fantasy-premier-league-begin/`, Jun 2018.

[13] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[14] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. *Bayesian data analysis*. CRC press, 2013.

[15] Hugo Greenhalgh. Meet the man behind bringing Fantasy Football to the UK. `https://www.ft.com/content/8dbc85cc-7eb0-11e7-ab01-a13271d1ee9c`, Sep 2017.

[16] Ronald A Howard. Information value theory. *IEEE Transactions on systems science and cybernetics*, 2(1):22–26, 1966.

[17] Josip Hucaljuk and Alen Rakipović. Predicting football scores using machine learning techniques. In *2011 Proceedings of the 34th International Convention MIPRO*, pages 1623–1627. IEEE, 2011.

[18] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[19] Hans Kellerer, Ulrich Pferschy, and David Pisinger. Multidimensional knapsack problems. In *Knapsack problems*, pages 235–283. Springer, 2004.

[20] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[21] Jonathan Robert Landers and Brian Duperrouzel. Machine learning approaches to competing in fantasy leagues for the NFL. *IEEE Transactions on Games*, 11(2):159–172, 2018.

[22] Andy Liaw, Matthew Wiener, et al. Classification and regression by RandomForest. *R news*, 2(3):18–22, 2002.

[23] William S Lovejoy. A survey of algorithmic methods for partially observed Markov decision processes. *Annals of Operations Research*, 28(1):47–65, 1991.

[24] Michael J Maher. Modelling association football scores. *Statistica Neerlandica*, 36(3):109–118, 1982.

[25] Tim Matthews, Sarvapali D Ramchurn, and Georgios Chalkiadakis. Competing with humans at fantasy football: Team formation in large partially-observable domains. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[26] Francisco S Melo. Convergence of Q-learning: A simple proof. *Institute Of Systems and Robotics, Tech. Rep*, pages 1–4, 2001.

[27] Stuart Mitchell, Michael OSullivan, and Iain Dunning. Pulp: a linear programming toolkit for python. *The University of Auckland, Auckland, New Zealand*, 2011.

[28] Chris Piech. Stanford University: CS221. `https://stanford.edu/~cpiech/cs221/homework/prog/pacman/pacman.html`.

[29] Orbis Research. Fantasy Sports Market Estimated to grow at 13.2by 2025-Key Insights on Revenue (Value), Potential Application, Economic Fluctuations, Growth Rate and Future Trends: Orbis Research. `https://www.globenewswire.com/news-release/2019/08/24/1906227/0/en/Fantasy-Sports-Market-Estimated-to-grow-at-13-2-to-reach-33200-Mn-by-2025-Key-Insights-on-Revenue-Value-Potential-Application-Economic-Fluctuations-Growth-Rate-and-Future-Trends-Or.html`, Aug 2019.

[30] Yara Rizk, Mariette Awad, and Edward W Tunstel. Decision making in multi-agent systems: A survey. *IEEE Transactions on Cognitive and Developmental Systems*, 10(3):514–529, 2018.

[31] Loren Shure. Multi-Armed Bandit Problem and Exploration vs. Exploitation Trade-off. `https://blogs.mathworks.com/loren/2016/10/10/multi-armed-bandit-problem-and-exploration-vs-exploitation-trade-off/`, Oct 2016.

[32] David Silver and Joel Veness. Monte-Carlo planning in large POMDPs. In *Advances in neural information processing systems*, pages 2164–2172, 2010.

[33] Surya. Orgin and History of FPL: When did Fantasy Premier League start? `https://allaboutfpl.com/fpl-history-and-origin-of-fantasy-football/`, May 2020.

[34] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[35] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.

[36] WT Luke Teacy, Georgios Chalkiadakis, Alessandro Farinelli, Alex Rogers, Nick R Jennings, S McClean, and Gerard Parr. Decentralized Bayesian reinforcement learning for online agent collaboration. In *11th International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[37] Gerald Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[38] Robert J Tibshirani and Bradley Efron. An introduction to the bootstrap. *Monographs on statistics and applied probability*, 57:1–436, 1993.

[39] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[40] Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010.

[41] Wikipedia contributors. Law of total expectation — Wikipedia, the free encyclopedia, 2020. [Online; accessed 6-September-2020].

[42] Wikipedia contributors. Markov decision process — Wikipedia, the free encyclopedia, 2020. [Online; accessed 5-September-2020].

[43] Wikipedia contributors. Partially observable markov decision process — Wikipedia, the free encyclopedia, 2020. [Online; accessed 6-September-2020].

[44] Wikipedia contributors. Temporal difference learning — Wikipedia, the free encyclopedia, 2020. [Online; accessed 13-August-2020].