



**BRNO UNIVERSITY OF TECHNOLOGY**  
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

**FACULTY OF INFORMATION TECHNOLOGY**  
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

**DEPARTMENT OF INFORMATION SYSTEMS**  
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

## **INTELLIGENT MANAGER OF FANTASY PREMIER LEAGUE GAME**

INTELIGENTNÍ Manažer Hry Fantasy Premier League

**MASTER'S THESIS**  
DIPLOMOVÁ PRÁCE

**AUTHOR**  
AUTOR PRÁCE

**Bc. MAROŠ VASILIŠIN**

**SUPERVISOR**  
VEDOUcí PRÁCE

**Ing. JIŘÍ HYNEK, Ph.D.**

**BRNO 2020**

## Master's Thesis Specification



23053

Student: **Vasilišin Maroš, Bc.**

Programme: Information Technology Field of study: Information Systems

Title: **Intelligent Manager of Fantasy Premier League Game**

Category: Artificial Intelligence

Assignment:

1. Get acquainted with the Fantasy Premier League game. Analyze existing tools for management and recommendation of players, summarize their advantages and disadvantages.
2. Study the principles of supervised learning.
3. Find and analyze features which affect performance of the Premier League's players in the real upcoming league rounds.
4. Design an artificial intelligence which would predict the performance of Premier League players in the upcoming league rounds. Design a web manager of the Fantasy Premier League game which would use the predictions to improve the score of the game.
5. Implement the web manager and the artificial intelligence.
6. Evaluate the usability of the web manager and results of predictions. Analyze the improvement of the game play score. Design improvements of the software.

Recommended literature:

- Machine Learning. *Coursera: Online Courses From Top Universities* [online]. 2019 [cit. 2019-10-13]. Available at: <https://www.coursera.org/learn/machine-learning>
- Premier League. *Fantasy Premier League, Official Fantasy Football Game of the Premier League* [online]. 2019 [cit. 2019-10-13]. Available at: <https://fantasy.premierleague.com/>

Requirements for the semestral defence:

- Items 1 to 4.

Detailed formal requirements can be found at <https://www.fit.vut.cz/study/theses/>

Supervisor: **Hynek Jiří, Ing., Ph.D.**

Head of Department: Kolář Dušan, doc. Dr. Ing.

Beginning of work: November 1, 2019

Submission deadline: May 20, 2020

Approval date: October 23, 2019

## Abstract

Fantasy Premier League online game gives millions of players around the world the chance to become a manager of their club for a while. The results and scores in the game depend on correctly predicting how players will behave in real football matches. If the user had software for predicting and analyzing players' future performance, it would help with making decisions and the outcome of the game could significantly improve. This master's thesis deals with the design and implementation of a prediction model that uses recurrent neural networks for time series prediction throughout the game season. Various methods were used to process player and club data for the last 4 seasons. The results are presented in the form of a web application where users can use the created model on their teams. Performance and accuracy of prediction methods were tested on data from the last season of the Premier League and algorithm predictions were in most cases close to reality. If the user used the prediction model's advice 100% in the game, he would score more points than a regular player who does not use any prediction model.

## Abstrakt

Hra Fantasy Premier League poskytuje miliónom hráčov po celom svete možnosť stať sa na chvíľu manažérom svojho vlastného klubu. Výsledky a bodové ohodnotenie v hre závisia na správnom predvídaní, ako sa budú hráči chovať v skutočných futbalových zápasoch. Ak by pri tomto rozhodovaní pomáhal software na predikciu a analýzu budúcich výkonností hráčov, výsledky v hre sa môžu rapične zlepšiť. Táto diplomová práca sa zaobráva návrhom a implementáciou predikčného modelu, ktorý využíva rekurentné neurónové siete na predikcie časových radov počas celej sezóny v hre. Boli použité metódy na spracovanie dát o hráčoch a kluboch za posledné 4 sezóny. Výkonnosť a presnosť predikčných metód boli testované na dátach z poslednej sezóny Premier League a predikcie algoritmu sa vo väčšine prípadov blížili realite. Ak by sa užívateľ držal predikčného modelu v hre stopercentne, získal by väčší počet bodov ako bežný hráč, ktorý žiadny predikčný model nepoužíva.

## Keywords

Fantasy Premier League, Premier League, fantasy sports, football, prediction, neural networks, recurrent neural network, linear programming, machine learning, supervised learning, time series forecasting, web application

## Klíčová slova

Fantasy Premier League, Premier League, fantasy športy, futbal, predikcie, neurónové siete, rekurentné neurónové siete, lineárne programovanie, strojové učenie, učenie s učiteľom, predikcie časových radov, webová aplikácia

## Reference

VASILIŠIN, Maroš. *Intelligent Manager of Fantasy Premier League Game*. Brno, 2020. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Jiří Hynek, Ph.D.

# **Intelligent Manager of Fantasy Premier League Game**

## **Declaration**

Hereby I declare that this term project was prepared as an original author's work under the supervision of Ing. Jiří Hynek. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....  
.....  
.....

Maroš Vasilišin  
June 3, 2020

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Fantasy sports</b>	<b>5</b>
2.1	Background . . . . .	5
2.2	Types of fantasy sports . . . . .	7
2.3	Fantasy Premier League Dynamics . . . . .	8
<b>3</b>	<b>Analysis of existing predictors</b>	<b>14</b>
3.1	Most popular predictors . . . . .	14
3.2	Most popular features in existing applications . . . . .	20
3.3	Useful features missing in existing applications . . . . .	20
<b>4</b>	<b>Machine learning</b>	<b>21</b>
4.1	Introduction . . . . .	21
4.2	Supervised learning . . . . .	22
4.3	Unsupervised learning . . . . .	23
4.4	Regression algorithms . . . . .	23
4.5	Neural networks . . . . .	25
4.6	Time series models . . . . .	28
4.7	Underfitting vs Overfitting . . . . .	29
4.8	Algorithm selection . . . . .	30
<b>5</b>	<b>Design of the application</b>	<b>34</b>
5.1	Requirements for prediction application . . . . .	34
5.2	System architecture design . . . . .	35
5.3	Data fetching and processing flow . . . . .	36
5.4	Dataset analysis and feature selection . . . . .	37
5.5	Design of the GUI . . . . .	38
<b>6</b>	<b>Implementation</b>	<b>42</b>
6.1	Application server . . . . .	42
6.2	Data preparation . . . . .	46
6.3	Downloading and extraction . . . . .	46
6.4	Filtering and feature selection . . . . .	47
6.5	Forecaster and statistical models . . . . .	48
6.6	Webserver . . . . .	50
<b>7</b>	<b>Experiments and evaluation</b>	<b>57</b>

7.1	Model attributes and feature selection testing . . . . .	57
7.2	Graphical user interface testing . . . . .	58
7.3	Prediction accuracy testing . . . . .	59
<b>8</b>	<b>Conclusion and future work</b>	<b>65</b>
	<b>Bibliography</b>	<b>66</b>

# Chapter 1

## Introduction

Fantasy sports are online games, in which the user takes on the role of a manager of a sports club. The user tries to accurately predict the performance of players in real-life matches. He tries to create a team of players, who he thinks will perform best in real fixtures. Then all of the users are awarded points based on real-life results and on how their selected team of players performed.

These types of games exist for a long time, but they gained the biggest popularity relatively recently, with the invention of the internet. Now people all over the world can participate in these games together. Most popular fantasy sports are baseball, football and American football and tens of millions of users play them every year.

The main problem with fantasy games is that there is a very slim chance of winning anything for a regular user. The biggest prizes are won by merely 0.001% of the users. These top players are either sports experts, who devote a lot of hours to the game every day, or they are using some software to help them. Various statistical models have proven to help with the predictions of the results over the years. For regular users, trying to come up with patterns in players' performance and history in real life seems nearly impossible, but with the help of a computer program, the process can be automated.

For a long time, fantasy sports enthusiasts are trying to find out, what is the exact connection between a player's historical results and possible future results. There are a lot of projects, proprietary and public, where people are using machine learning and artificial intelligence to predict players' performance. Most of them claim that their algorithms are the best available, but they often focus on one or more aspects of the game in detail and neglect the others.

Best fantasy players, however, won't publish their models. Why would they, if their model gives them a competitive edge over the other users? That's the reason why most of the free publicly available tools are not very accurate or precise. There is also not any right solution to the problem, luck and chance play a considerable part in sports, so no tool can be 100% correct.

In this thesis, I will focus on one particular fantasy football application, the Fantasy Premier League application. It is an official application of the English Premier League and is currently the most popular fantasy football application in Europe. My target will be to create a prediction algorithm that will use historical and statistical data of every current Premier League player. It will predict their future performance in various aspects of the game. Part of this thesis will be to determine, which statistics have a bigger or smaller impact on future performance and what are the most useful statistics to use in performance prediction in football matches. This will be used to give bits of advice to application users

about what to change in their team and how to shape their squad in the game. Datasets from previous seasons will be used to train the algorithm, while data from the current 2019/2020 season will be used for testing.

There will be 2 outputs of the thesis. First will be a web application, where users will be able to log in with their Fantasy Premier League accounts and use various features to help them achieve better results in the game. This application would use the second output, which will be a server with a running trained prediction algorithm. This would return the data and the predictions based on the user's squad and requirements.

Web application will offer many useful features for users to analyze their squad, make transfers and substitutions. It will provide some of the most popular features on existing websites. I will also aim to add some features, that are not available in existing solutions, but I think they can add to the user experience and help them in the game. The web application will also aim to help users understand how the prediction works and why it predicted the scores they got.

Fantasy sports, their history, and rules of Fantasy Premier League application are described in Chapter 2. Existing predictor applications and what they offer to the user are described in Chapter 3. In the next Chapter 4, there is a detailed information about what machine learning techniques were used for prediction. Chapter 5 consists of the design of the web application with user requirements. There is also a comparison of different machine learning libraries. Later I describe the dataset I used for creating the prediction algorithm, and how I chose which features are more useful than the others. The Chapter 6 describes the process of implementation of both the predictor and the graphical user interface. There is also more information about how the prediction model determines the predictions. The Chapter 7 contains information about experiments with the model and the web application. It also contains evaluation of the predictions and compares their success rate to results that I got without using the predictor during current season.

# Chapter 2

## Fantasy sports

Fantasy sports are games, now mostly online, where users create teams of real-world players of a professional sport. The most common are football, baseball, American football, basketball and cricket fantasy sports. Players are assigned points based on the performance of chosen players in real-life fixtures. Every fantasy sport has its scoring system, and it awards the user's team the total points of his team's players. Since there are many fantasy leagues and apps, points are awarded differently in each of them. Most sophisticated ones have developed web or mobile application for scoring and team selection. But there are still simple ones where the “league coordinator” awards points manually and controls his local league. In fantasy sports, lots of real-life managerial responsibilities are simulated, such as transferring players, injuries, suspensions and more. There are mainly two kinds of fantasy sports formats, season-long and daily. A Fantasy Sports & Gaming Association<sup>1</sup> research [1] from 2019 stated that in 2017 the industry had around 60 million players in the United States and Canada.

### 2.1 Background

Picking real players and running fantasy sports competitions is not a new concept, it is around since the 1960s. However, then it was still organized sporadically and for fun. The first fantasy football league dates itself to 1962 [8] in Oakland, USA. Around that time, the first baseball fantasy league also started [22]. First, fantasy sports gained more popularity in the 1980s, when some sports journalists started playing together, so it spread some media interest. At this time, a large number of companies started calculating statistics for fantasy leagues and faxing results to their users. In the 1990s [1], around 3 million people were playing. The growth in popularity in these games is plotted in Figure 2.1.

The evolution of technology allowed players to play from virtually anywhere. With smartphones, users can check or change their teams very quickly. Also, live drafting became very popular. With the coming of the internet, many magazines and websites started covering fantasy football, and also new sports were added, for example, hockey and basketball. A website called RotoNews<sup>2</sup> (now renamed to RotoWire) was one of the first [21] to cover fantasy sports. And many others followed. The boom of fantasy sports can be closely connected to the boom of the internet. At this time many companies moved from free to paid models. In 2013 a Forbes article [7] reported that 32 million Americans did spend

---

<sup>1</sup><https://thefsga.org/>

<sup>2</sup><https://rotowire.com/>

467 dollars a person playing fantasy sports. Since 2012 [9] with the emergence of mobile applications, a lot of them were created for fantasy sports, for playing and for predictions as well.

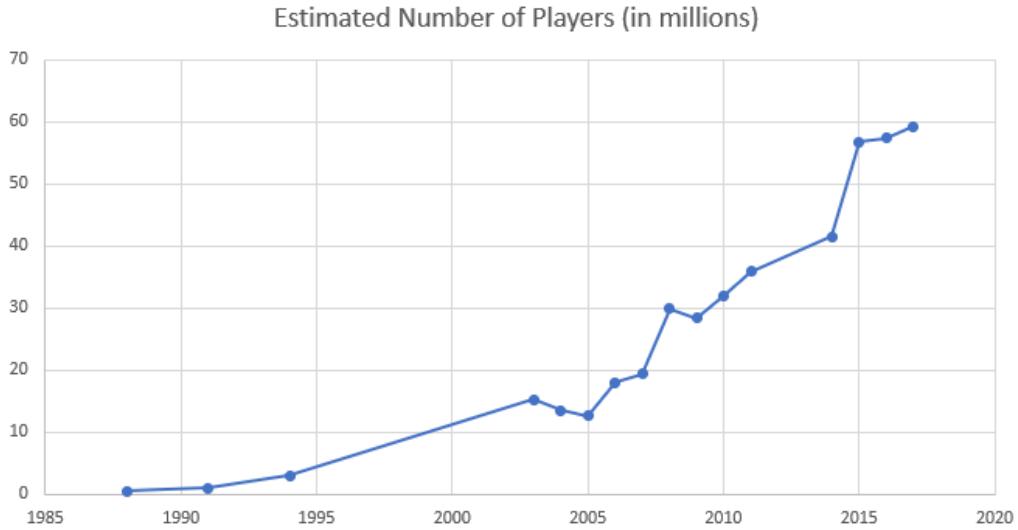


Figure 2.1: Estimated number of fantasy sports players according to 2019 research since year 1988. [1]

It was believed you have to be a sports fan to ever win at fantasy sports games. In 2010 [17] man named John Rozek was named the best fantasy football player in the world, without being a football fan. He was just enthusiastic about statistics and analyzing a particular problem. He had only a moderate knowledge of football at the time. Now when most of the fantasy sports are done by computers, and scoring could be done quickly, more scoring methods were created. All of the player statistics are collected automatically and you don't even need to know the people who you are in a fantasy league with. While it motivated a lot of users to watch sports and other teams, fantasy sports have now become a very successful business.

While fantasy sports are very popular today, one question is asked very frequently. Are fantasy games gambling or not? To answer this question, MIT researchers [6] found out, that fantasy sports games are a game of skill, rather than luck, even though a little luck is needed. Examples of some games that are pure luck or pure skill can be found in Table 2.1. They analyzed thousands of fantasy games over many seasons. To win, players need to have a good knowledge of rules, injuries, stats, weather and other factors. It is not based on pure chance. For companies, which make revenue of fantasy sports, the fantasy games must be classified as a game of skill. In many states, betting on sports is illegal. The factors that influenced this decision are that if it was a game of luck, every player should have the same chance of winning. But this is not true, people who play more games have a statistically bigger chance of getting more points. Also, in skill-based games, a player's performance should be persistent over multiple rounds and his actions should have an impact on a game. Both of which are true for fantasy sports. So the conclusion of the research was, that they are mostly about skill and little about luck. Now, most of the fantasy sports are free, companies make revenue mostly on advertisements.

Table 2.1: Games of luck and skill examples

Games of luck	Games of skill	Luck + Skill
Roll of a dice	Chess	Football
Roulette	Marathon	Baseball
Slot machines		Hockey

Origins of the online fantasy sports game are based in North America, where it was first offered by Yahoo in 1999 [19], other portals like ESPN and CBS followed shortly. Watching and following games through fantasy sports has made many new fans of the respective sports. The concept of outsmarting your opponents using your knowledge and the overall excitement of the game is what brings all those people to play them. As a result of this emergence of new followers, it created interest in stakeholders of the leagues, clubs, media, and sponsors. The possibility of monetizing this interest is what makes these games so attractive.

## 2.2 Types of fantasy sports

There are two main types: season-long and daily fantasy sports games. In LIVE Production research [18], almost 87% of the respondents said they prefer to play free season-long fantasy sports. But it has to be said the research was conducted in Europe, where daily fantasy (DFS) is not as popular as in other parts of the world (United States, Canada, Asia, Africa).

Most popular sports according to website WordAtlas<sup>3</sup>, are soccer – 4 billion fans, cricket 2.5B, field hockey 2B [20]. Interestingly, basketball is 7th, baseball 8th, and American football with ice hockey are not even in the top 10. The whole top 10 can be found in Figure 2.2. It shows that the popularity of sports does not correlate with its popularity in fantasy sports. The most popular fantasy sports are American football, basketball, baseball, hockey, and football. This is true for America, in Europe it is mostly soccer.

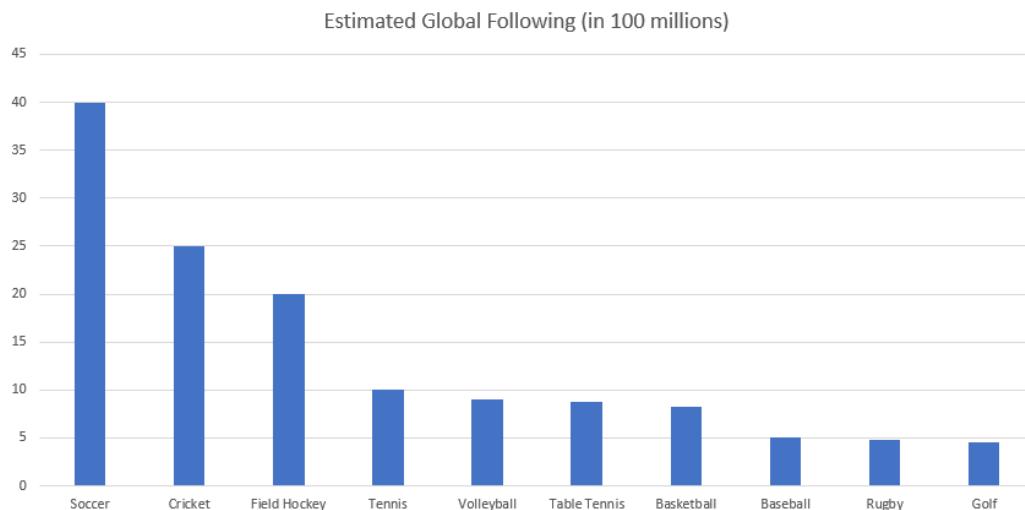


Figure 2.2: Estimated global following of the most popular sports in the world

---

<sup>3</sup><https://www.worldatlas.com/>

## Season-long fantasy sports

In season-long fantasy sports, users choose players at the start of the season. They have to think ahead and take closer attention to form, injuries, fixture difficulty, etc. Many users just select the team at the start and then forget about it until the end. During the season, every week users can usually make 1 or at most 2 trades. Points are awarded each round and prizes at the end of the season. Usually, users have a limited budget to choose players. If that was not the case, they would just pick the best players in the league. This way teams will be more balanced, with some cheaper and some more expensive players. In fantasy football, users pick the strongest 11 players from their squad every week. Users have to examine fixtures, injuries and past statistics to do it. Each website awards points differently but there are many similarities, for example, a lot of points are awarded for goals, assists, clean sheets, also minus points for red cards or penalties conceded. These games also offer leagues to compete with friends. In this thesis, I will focus on season-long fantasy football game from English football league: Fantasy Premier League<sup>4</sup>.

## Daily fantasy sports

The main difference to the season-long fantasy is, that they are played across a shorter period of time, such as a single week, rather than an entire season. Users don't care about long term game, they emphasize most value for the lowest cost right now. Each player will earn points for one fixture only. This format allows more contest types and variations, and it gains more popularity nowadays, mostly in the United States. They are typically played with an entry fee and they are heavily advertised. Prize money is distributed right after the round finishes. Legality of daily fantasy sports was often questioned due to its "cash game" properties. Another difference is a cash pot. While season-long games are more often played for fun, and prize pots are not that big, a lot more money is involved in daily fantasy sports. Over the years, two companies have taken over the daily fantasy sports market, DraftKings<sup>5</sup>, and FanDuel<sup>6</sup>. DraftKings alone hosts over 8 million users. Not long ago, an entirely new segment of the market was created, based on video games — daily fantasy e-sports.

## 2.3 Fantasy Premier League Dynamics

Fantasy Premier League (FPL) is an official fantasy game of the Premier League. At the start of 2019/2020 season, around 7 million people are registered to play. There are a lot of football fantasy leagues but this one is the most popular of them. The fantasy league is organized by Premier League itself, so the level of details, precision and administration is so much better than the rest.

### English Premier League

The Premier League is a top tier of the English football pyramid. It consists of 20 teams from England. There are also some teams from Wales, that are eligible to play in it, but they are currently in the one league below. It is the most-watched league in the world, being broadcasted to 188 countries. It takes place between August and May every year,

---

<sup>4</sup><https://fantasy.premierleague.com/>

<sup>5</sup><https://www.draftkings.com/>

<sup>6</sup><https://www.fanduel.com/>

Every team plays each other twice, at home and away. Total 380 matches a season. Three points are awarded for a win, 1 for draw and 0 for a loss. The team with most points at the end of the season wins. Bottom 3 teams are relegated to the lower league, called the Championship. They are replaced with 3 teams from the lower tier. If teams finish on the same amount of points, the position is determined by goal difference and the number of goals scored.

## Start of play

Rules of the FPL are similar to most of the fantasy football apps. The user has a budget of 100 million pounds to build a squad of 15 players. A squad must consist of 2 goalkeepers, 5 defenders, 5 midfielders and 3 forwards. A maximum of 3 players can be chosen from one club. Only current Premier League players can be selected. Every player has his value. This value depends on player reputation, results, performance, and on market situation. If a lot of users are buying a player, his value rises and if a lot of them are selling the player, his value decreases. The most expensive players are usually the attackers. Example squad from the FPL game can be found in Figure 2.3.

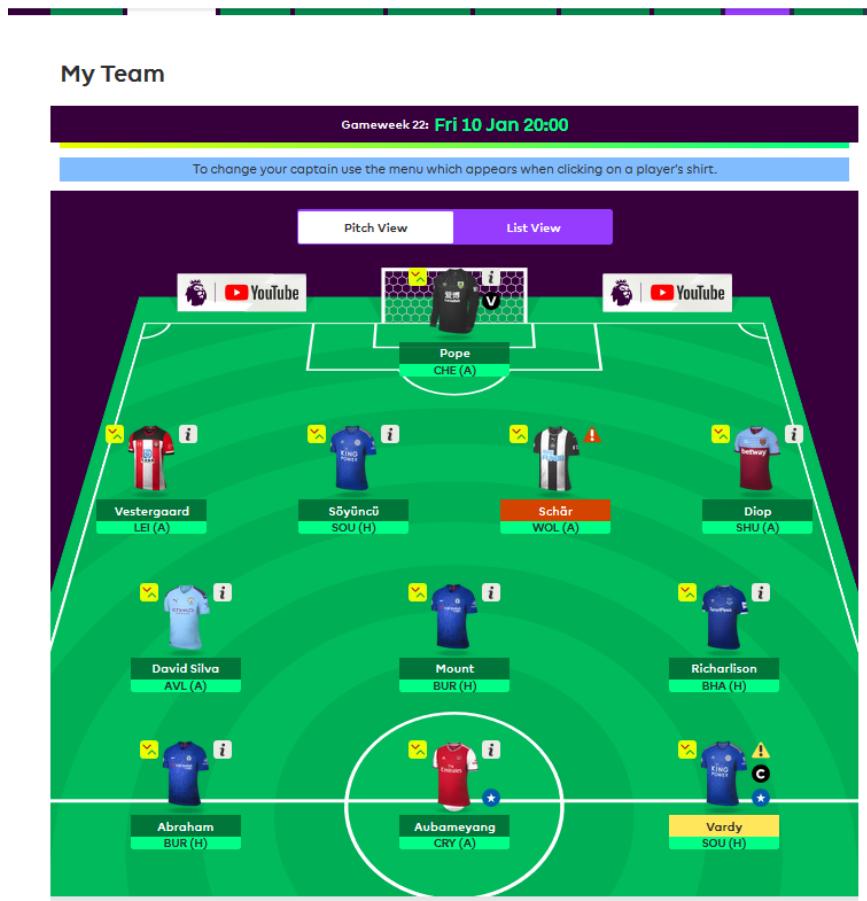


Figure 2.3: My Team section of the Fantasy Premier League Application. It shows user's currently selected squad with some information about the players.

## Game rounds

There are 38 rounds of play, one for each round of the Premier League. Before every round, the user has to choose his starting eleven. He does not need to change anything, if he is happy with his previous selection. There is also a captain selection, this player gets double points for the round. Any formation can be chosen as long as it respects the following rules: 1 goalkeeper, 3 defenders, 2 midfielders, 1 forward at least. The deadline for selection is 1 hour before kickoff of the first match of the week. Then users are waiting for real-life fixtures to finish. Premier League is usually played on weekends with an occasional game on Friday or Monday evening. Once in a while, games are played in midweek.

## Scoring

Players are awarded points for their performance on the pitch. There are different scoring systems for each position of the team. There are also points deducted for any offenses or misconduct in the game, or just for poor performance. In case some of the starting eleven don't play, the reserve players are awarded points. All the total points of 11 players are then added to the user's point tally for the current round. Detailed scoring points can be found in Table 2.2. Example scoring table from the FPL game is shown in Figure 2.4.

		Pitch View														List View									
		Starters	Pts	MP	GS	A	CS	GC	OG	PS	PM	YC	RC	S	B	BPS	I	C	T	II					
i	V   Pope BUR GKP	0	90	0	0	0	5	0	0	0	0	0	0	2	0	14	16.8	0.0	0.0	0.0	1.7				
A	 Schär NEW DEF	1	45	0	0	0	1	0	0	0	0	0	0	0	0	4	14.2	0.6	0.0	0.0	1.5				
i	 Robertson LIV DEF	6	75	0	0	1	0	0	0	0	0	0	0	0	0	24	6.0	3.6	2.0	1.2					
i	 Söyüncü LEI DEF	2	90	0	0	0	1	0	0	0	0	0	0	0	0	17	23.6	11.0	55.0	9.0					
i	 Bernardo Silva MCI MID	1	64	0	0	0	2	0	0	0	0	1	0	0	0	10	2.0	26.4	50.0	7.8					
i	 David Silva MCI MID	2	90	0	0	0	2	0	0	0	0	0	0	0	0	13	18.8	49.3	62.0	13					

Figure 2.4: Points section of the Fantasy Premier League Application. The points are awarded for a lot of actions on the pitch, more detailed description can be found in the game rules.

There are also bonus points. Three best players in each match get awarded bonus points. They get 3,2,1 points respectively, and when there is a tie, points are split. Bonus points are awarded for various events in the game, detailed view can be found in Table 2.3.

## Transfers

Between rounds, users are allowed one transfer. On some occasions two transfers, if they did not use last round's transfer token. These transfers also need to be within boundaries of the 100 million budget cap. Every other transfer, if needed, costs 4 points. Transfer market from the FPL game is shown in Figure 2.5.

Player Selection

View: All players

Sorted by: Total points

Search: Q

Max cost: Between 3.9 and 12.3

587 players shown

Goalkeepers		£	=
<i>i</i>	Schmeichel LEI GKP	5.4	84
Defenders		£	=
<i>i</i>	Alexander-... LIV DEF	7.5	112
<i>i</i>	Robertson LIV DEF	7.0	98
<i>i</i>	Lundstram SHU DEF	5.1	98
<i>i</i>	Pereira LEI DEF	6.4	92
<i>i</i>	van Dijk LIV DEF	6.4	92
<i>i</i>	Baldock SHU DEF	5.0	91
<i>i</i>	Evans LEI DEF	5.3	86
Midfielders		£	=
<i>i</i>	De Bruyne	10.4	141
<i>i</i>	Son TOT MID	9.8	89
<i>i</i>	Willian CHE MID	7.2	86
<i>i</i>	Cantwell NOR MID	4.9	84
<i>i</i>	Pérez LEI MID	6.0	83
<i>i</i>	Traoré WOL MID	5.6	83
Forwards		£	=
<i>i</i>	Vardy LEI FWD	10.1	144
<i>i</i>	Aubameyang ARS FWD	10.8	123
<i>i</i>	Rashford MUN FWD	9.1	122
<i>i</i>	Ings SOU FWD	6.7	118
<i>i</i>	Abraham CHE FWD	7.8	115
<i>i</i>	Jiménez WOL FWD	7.5	106

Figure 2.5: Transfers section of the Fantasy Premier League Application. It shows every player available for purchase.

Table 2.2: FPL scoring system

Action	Points
For playing up to 60 minutes	1
For playing 60 minutes or more (excluding stoppage time)	2
For each goal scored by a goalkeeper or defender	6
For each goal scored by a midfielder	5
For each goal scored by a forward	4
For each goal assist	3
For a clean sheet by a goalkeeper or defender	4
For a clean sheet by a midfielder	1
For every 3 shot saves by a goalkeeper	1
For each penalty save	5
Bonus points for the best players in a match	1-3
For each penalty miss	-2
For every 2 goals conceded by a goalkeeper or defender	-1
For each yellow card	-1
For each red card	-3
For each own goal	-2

## Mini-leagues

Users can play not only against everyone in the world, but they can also create their own mini-leagues. Usually a group of friends or colleagues will create a league and compete against each other. Apart from user-created leagues, there are also global ones. For example, these could be a fan league of some particular club. There are also different scoring

systems in the leagues, *classic* described above or *head to head*, where players are grouped to pairs each round.

Table 2.3: FPL bonus points scoring system. \*Net successful tackles is the total of all successful tackles minus any unsuccessful tackles. Players will not be awarded negative BPS points for this statistic.

Action	Bonus Points
Playing 1 to 60 minutes	3
Playing over 60 minutes	6
Goalkeepers and defenders scoring a goal	12
Midfielders scoring a goal	18
Forwards scoring a goal	24
Assists	9
Goalkeepers and defenders keeping a clean sheet	12
Saving a penalty	15
Save	2
Successful open play cross	1
Creating a big chance (a chance where the receiving player should score)	3
For every 2 clearances, blocks and interceptions (total)	1
For every 3 recoveries	1
Key pass	1
Successful tackle (net*)	2
Successful dribble	1
Scoring the goal that wins a match	3
70 to 79% pass completion (at least 30 passes attempted)	2
80 to 89% pass completion (at least 30 passes attempted)	4
90%+ pass completion (at least 30 passes attempted)	6
Conceding a penalty	-3
Missing a penalty	-6
Yellow card	-3
Red card	-9
Own goal	-6
Missing a big chance	-3
Making an error which leads to a goal	-3
Making an error which leads to an attempt at goal	-1
Being tackled	-1
Conceding a foul	-1
Being caught offside	-1
Shot off target	-1

## Chips

There are five chips to be played during the season. They can boost team's total points for a current round. Only one chip can be played in one round.

- Wildcard – The user can overhaul his team without having to pay for transfers. This can be used twice a season.

- Triple captain – round where the captain gets 3x points
- Bench boost – round where all 15 players get points
- Free hit – resets the team for one week, it reverts after the round is finished

## Important factors in the game

In this section, I will focus on some factors of the game which may distinguish average and great players. To be a great fantasy football player, only football knowledge is often not enough.

- Good footballers in real life may not be a good option for the fantasy league team. Since FPL scoring system is mainly based on statistics, and not on the overall impact on the game, many great footballers may score low points in the game, despite their impact on the game. This is an example of players who play as deep-lying playmakers or defensive midfielders. Their impact on the game may be huge, but fantasy game does not yet have scores for their contribution. This means their market value is very low and many inexperienced managers would choose them in their squads for a lower price, but they will score low points. On the other hand, there are players of low quality, who score fantasy points very often. Best practice is not to choose players on preferences or real qualities, but purely on fantasy football performance potential.
- The rotation of players is very important. If a team rotates two strikers, there is a good chance that selected player would not score points week in week out. If a player comes from an injury, there is a chance that for a few games he would be only a substitute or not feature at all. Also, new signings tend to have a warming period, for the first few months their performance may not be best.
- Picking players that are good on set pieces. Players who tend to score from free-kicks or corners are very valued because goals and assists have a high point value. A good penalty taker in the team is also a great shout.
- Be patient with some of the players. If the player does not score in two matches, don't transfer him out immediately. Every player has some bad form occasionally.
- Best users use their Bonus Chips wisely and plan their usage upfront.

# Chapter 3

## Analysis of existing predictors

In this chapter, I will write more about existing applications for predicting player's scores in fantasy football, mostly for Fantasy Premier League. There are also a lot of predictors for other sports, and for other fantasy games, such as daily fantasy. But I will focus on the most popular predictors for season-long fantasy games. It is worth mentioning some fantasy games, where players are fictional and made up by users themselves. Users are affecting players' performance and attributes in these games in various ways, so for these games, prediction is not worth it and practically impossible.

Later in this chapter, I will write about the most popular features of these applications and also features that are missing in them. Analyzing existing applications and determining, what makes them great and what are their weak spots should make a base for determining requirements for the application I would create in this thesis. A review of the findings will be in the last section of this chapter.

### 3.1 Most popular predictors

This section contains detailed analysis of two most advanced fantasy football predictors I found. Also some other smaller projects are mentioned in the end.

#### Fantasy Football Fix

Fantasy Football Fix<sup>1</sup> is by far the most popular and sophisticated tool now available for predicting player performance in Fantasy Premier League online game. It contains a lot of useful functions for various aspects of the game. They are claiming to be using what they say is "The most powerful prediction algorithm on the market". Fix, how they are shortly calling it, has also Android and iOS application, apart from the most used web one. These apps have fewer functions, but still, offer most of the functionality. As a part of the app, they have several writers, who are contributing to the blog page, where they explain the algorithm's decisions for the last weeks. This website has a very pleasant graphic design, a good algorithm for prediction and an option for a mobile app is useful.

Fix offers a free version, which contains basic functionality, and also a Premium version, which users can try for free for 7 days in the trial version. The trial version is good for trying out all the advanced functions to determine, if the app helps user in a significant way and whether it is good worth of money.

---

<sup>1</sup><https://www.fantasyfootballfix.com/>

The great thing about Fix is that it uses direct login logic from the Fantasy Premier League (FPL) app, so users don't need another set of credentials just for Fix. Users just log in with their FPL account and password, and the app automatically loads their profile and squad from the FPL website.

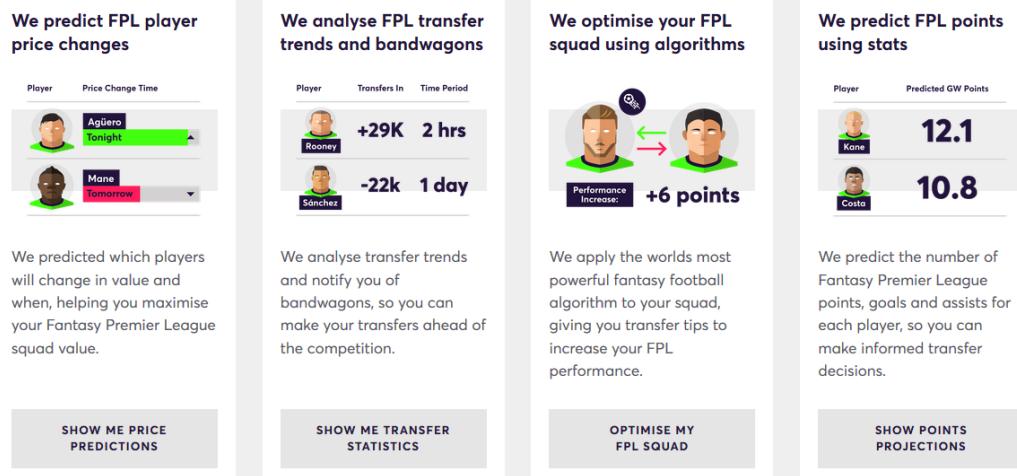


Figure 3.1: Preview of the most popular features in Fantasy Football Fix online prediction application

These are the features of the app, that are available for all the users for free. Most of these free features are used very often by users and should be available in every fantasy sports prediction app.

- Dugout is a unique feature for Fix, it is a dashboard, where user gets comprehensive information about the current status of the game. It contains information about players with the most overall increase in prizes, to see if the user wants to add them to his squad. It also shows some of the recommended transfers with the current market situation. It shows how the squad fared in the last round, who had a great week or bad one. It also shows short information about predicted results in next week and possible suspensions. Parts of the Dugout screen can be found in Figures 3.2 and 3.3.
- The assistant manager is a classic feature, found in most of the predictors now. In this window, the user can see his squad and predicted transfers and changes in starting eleven for the next week. It also shows who should the user select for captaincy. This screen allows the user to browse the transfer market with a lot of filters to find the perfect player for his squad. Some premium features are also available on this page, but I will write about them later.
- Another free and popular feature is the Table of Injured players, where users can see players currently injured, with predicted time out of the game. The table of suspended players is also available.
- On another screen, the user can see in a comprehensive table how the prizes of players are moving up and down, which may help him select who to buy or sell. This proves to be a popular feature for users, who like to find cheap gems, players who deliver

good points for a low cost. Here user can also find some graphs with the top transfers in the game, which is more like a statistical feature, but some users find it helpful.

- There are also next round previews, player statistics, and league table as some of the more general features. Preview of some of the most popular features can be found in Figure 3.1.
- One of the useful features is an option to have notifications delivered to your phone or mailbox, with information about predictions for the next round and some articles in the blog section. This is mostly useful for users, who don't like to spend much time playing, or forgot about the game week.

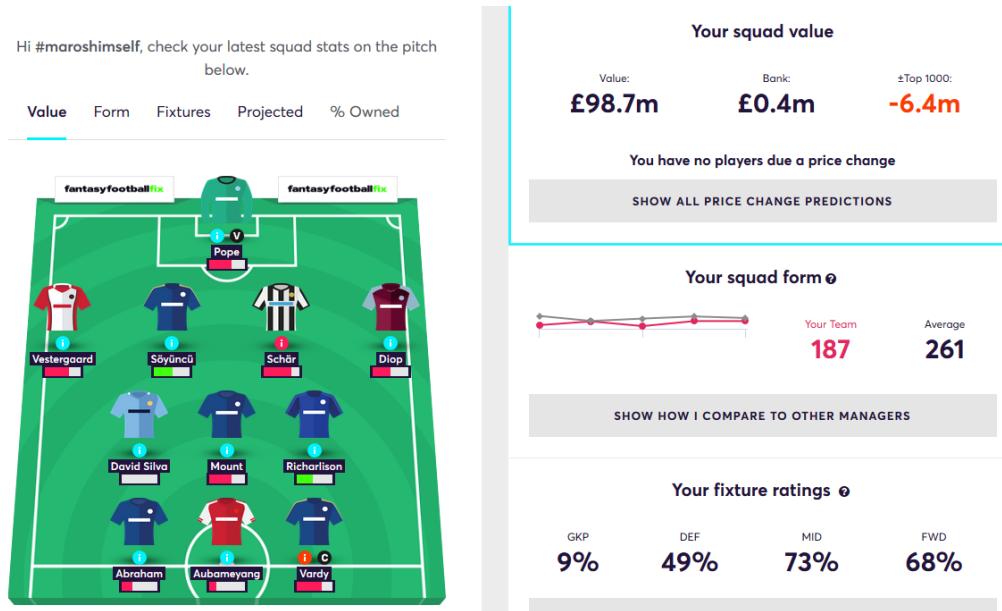


Figure 3.2: Part of the Dugout section in Fantasy Football Fix application. Contains information about squad value and form. Also option to optimize the squad.

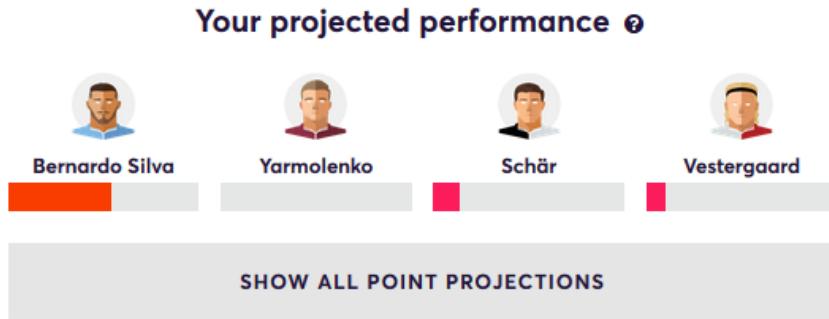


Figure 3.3: Part of the Dugout section in Fantasy Football Fix application. Contains preview of projected performance for some of the players in user's squad.

Fix offers a lot more features for premium users, and while the free version is enough for most players, some of the premium features are powerful. One of the great additions is

an option to ask Fix about rotating players in the next few fixtures. This is useful when counting for players match fatigue, fixture congestion or bad run of games. It also offers a more detailed point prediction for players, where users can see how many points are predicted for every category.

One of the great premium features is also predicting the opponent's starting eleven. This can have a big impact on the user's predicted lineup, as he can adjust it based on the opponent's weaknesses. There is also an analytical tool for analyzing fixture difficulty for the next game weeks.

Some premium features don't necessarily help users with the game but can be useful. For example, the option to export some subset of players statistics, comparing the players in a detailed way, whole game statistics like most used players, captains, formations, and many more. Option to see most user players is shown in Figure 3.4.

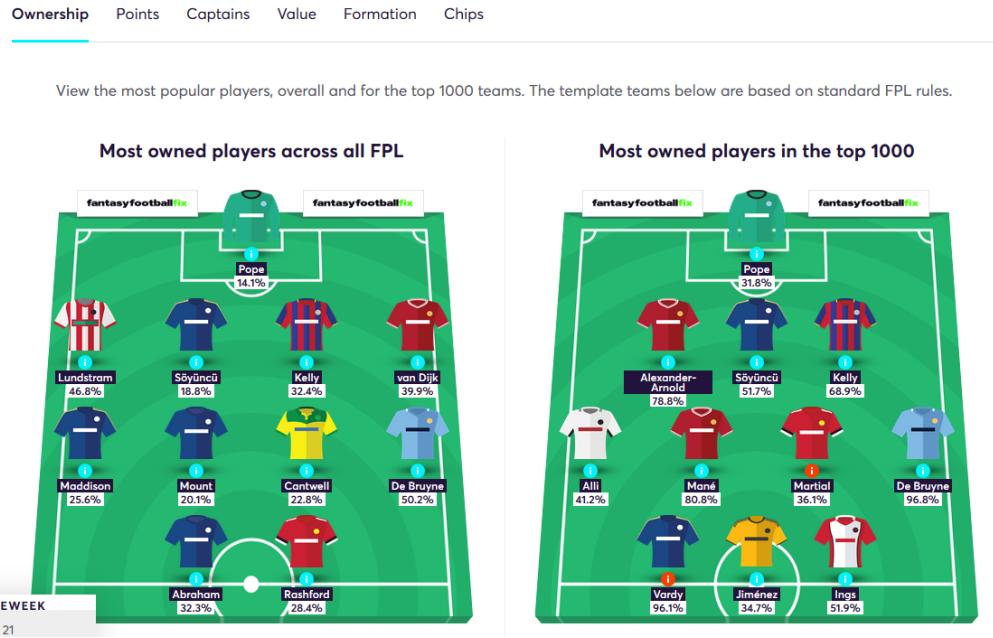


Figure 3.4: One of the Premium features in Fantasy Football Fix application: most used players among other users.

Fix is by far the most advanced tool currently available. While it has many great features, its main disadvantages are that it can be a little complicated. There are a lot of features in it and some can be quite difficult to use or understand. It offers only basic functions for free which makes it hard to use for more advanced predictions, should the user want them.

## Fantasy Overlord

Fantasy Overlord<sup>2</sup> is a simple website based on an article [26] about the theory, that Artificial Intelligence can beat people in fantasy football. It offers some prediction functionality as well as statistics. It has only a web version. In Overlord, users can import teams from

<sup>2</sup><https://fantasyoverlord.com/FPL>

FPL by using their FPL ID, so it does not have the FPL connection as Fix does. Users can predict their team's performance in any number of weeks in advance.

It offers all the basic functions like team changes and transfer recommendations. It also offers Overlord's picks for a Dream Team for a current round, which is just for statistical purposes as users cannot overhaul their whole team anyway (apart from using bonus chips). It also shows all the upcoming fixtures in the next game week, trends for player prizes and popularity changes, and also injury and suspensions table. Predictions for team and player points can be found in Forecasts part of the application. This is shown in Figure 3.5 and Figure 3.6.

Hover over the projected points to see the opponent

Club	22	23	24	25	26
Arsenal	32	35	32	29	42
Aston Villa	24	22	36	24	34
Brighton	30	43	30	31	39
Bournemouth	35	28	32	39	28
Burnley	25	29	25	33	32
Chelsea	32	33	31	28	35
Crystal Palace	41	30	36	41	33
Everton	32	25	37	23	34
Leicester	52	54	63	56	55
Liverpool	55	66	113	54	58
Man City	43	51	45	43	48
Man Utd	36	24	40	39	28

Figure 3.5: Team Point Forecast section of Fantasy Overlord online prediction application

Forwards	Team	22	23	24	25	26	Average	Price	Value
Rashford, Marcus	MUN	8.7	5.5	8.5	8.2	7.0	7.6	£9.1	0.835
Abraham, Tammy	CHE	7.7	6.5	7.8	6.8	8.4	7.4	£7.8	0.949
Iheanacho, Kelechi	LEI	7.6	7.6	7.9	7.9	7.4	7.7	£5.7	1.351
Aubameyang, Pierre-Emerick	ARS	7.0	6.6	7.8	7.2	7.7	7.3	£10.8	0.676
Ings, Danny	SOU	6.7	6.7	7.2	6.6	7.1	6.9	£6.7	1.030
Agüero, Sergio	MCI	6.6	7.4	5.6	6.9	7.0	6.7	£11.7	0.573
Jiménez, Raúl	WOL	6.2	6.8	5.4	5.8	5.6	6.0	£7.5	0.800
Firmino, Roberto	LIV	5.5	6.4	9.9	5.3	5.7	6.6	£9.3	0.710
Jesus,	MCI	4.8	5.5	4.0	5.3	5.1	4.9	£9.5	0.516

Figure 3.6: Player Point Forecast section of Fantasy Overlord online prediction application

One unique feature is called Overlord's picks, which shows top players predicted for a few upcoming weeks in every position. It is very useful for planning transfers ahead. It also offers an option to download historical data, presumably for users who want to create their prediction algorithms. Part of the Overlord's picks screen is shown in Figure 3.7.

Comparing to Fix, Overlord is much simpler and is free to use. It offers all the basic functionality for a standard user, who want to see predictions and some bits of advice

## Goalkeepers

for the 5 upcoming GWs

Top Point Scorers		Avg Points	Top Differentials (<10% Selected)		Avg Points
Fabianski	WHU	6.3	Fabianski	WHU	6.3
Ryan	BHA	4.6	Foster	WAT	4.5
Foster	WAT	4.5	Guita	CRY	4.4
Guita	CRY	4.4	Henderson	SHU	4.4
Henderson	SHU	4.4	Ramsdale	BOU	4.3
Ramsdale	BOU	4.3	Alisson	LIV	4.3
Alisson	LIV	4.3	Leno	ARS	3.8
Schmeichel	LEI	4.1	Dubravka	NEW	3.8
Patrício	WOL	3.9	Pickford	EVE	3.5
Leno	ARS	3.8	Krul	NOR	2.9

Top Values		Avg Points/Cost	Top for GW 21		Points
Fabianski	WHU	1.29	Leno	ARS	9
Ryan	BHA	0.96	McCarthy	SOU	8
Foster	WAT	0.94	Fabianski	WHU	6

Figure 3.7: Overlord's Picks section of Fantasy Overlord online prediction application

for his squad, and also some graphs. For a regular user, it is more useful than Fix for its simplicity, but it lacks some of the useful functions of Fix and its general style and graphical appearance.

## Other applications

Here I will just mention some of the other websites that are predicting players' performance but are less useful or focus on just one part of the game. These websites fall behind the previous two, but are still somewhat useful.

- FPL Statistics<sup>3</sup> focuses mostly on the prediction of players' prizes. It is not directly made for FPL as the previous two, but more for general purpose. It also has some tables with player statistics and trend graphs.
- There are other sources of great FPL predictions, such as some blog or Youtube channels, for example Fantasy Premier League Hints<sup>4</sup>, Fantasy Football Geek<sup>5</sup> or Fantasy Football Scout<sup>6</sup>.

<sup>3</sup><http://www.fplstatistics.co.uk/>

<sup>4</sup><https://fantasyPremierLeagueHints.blogspot.com/>

<sup>5</sup><https://www.fantasyfootballgeek.co.uk/>

<sup>6</sup><https://www.fantasyfootballsout.co.uk/>

## 3.2 Most popular features in existing applications

Here are the most popular features in current existing applications. Most used features are options for:

- transfer predictions, who to sign and who to release,
- captain choice predictions,
- option to see current squad and to switch players around,
- player statistics, injury and suspension tables,
- predicted points tally for players in the next round,
- transfer market, prize changes of the players,
- top transfers in some graphs or tables,
- next round preview.

## 3.3 Useful features missing in existing applications

This is a list of features which I think are available less or not at all and would help in predictions. Some of these features may help players in a small way but focus on aspects of the game that are not generally targeted.

- Long term prediction: while some pages offer a long-term prediction, they mostly do not offer comprehensive information, about how they are factoring some aspects of football season. What I think would improve user experience is to show the best rotation of players for the next rounds and planning their transfers. I would like to have an option to see Cup or European competition games, which most of the Premier League teams would play in. These games can have a big impact on player condition and performance. They can also affect the team's lineup for a league game if some tough cup game is played in a similar time.
- Bench prioritization: users should get information about their bench players, how to sort them in a most useful way, in case some of the starting lineup players would not feature in a game.
- Advice, when to use add-on chips: to let the user know which round is best for using that particular chip, can be crucial, as using those chips correctly can result in massive points increase for that round.
- Improved bargain hunter: the way to select lower prized gems from the transfer market in a more advanced way, offer more option to find them.
- Better Trends: if the user can see what other players are doing, he can also improve his overall game.

There are also some factors that should be included in the prediction algorithm, which I don't think many websites use. Such as counting the probability of BPS score for a player, last matches of that player against an opposition, how many days are between matches, match fatigue, and cup competitions. The goal of this thesis is to replicate the most popular features, and improve upon them with some additional tools.

# Chapter 4

## Machine learning

In this chapter, I will delve more into the theoretical approach to building models, that will predict player performance. For Fantasy Premier League performance prediction application, I require models that can reliably predict how players will perform based on historical statistical data. These models should be able to predict performance for many weeks in the future, so I require models that can take time factor into consideration. Introduction about what these models are and how to use them can be found in this chapter, along with a methodology, which I plan to use.

### 4.1 Introduction

Machine learning [15] is a large field in information technology. It is also presented in many other scientific fields, like statistics, artificial intelligence, neurobiology, and more. Machine learning is the study of the algorithms, that are using data to create models, and these models are used to learn from the data or for predicting new data values. This field is closely connected with statistics, that also can make predictions on the data. It is about creating algorithms, that can find statistical patterns in the data, they are simulating the approach, that a human would take to the particular problem.

The modern definition of the term machine learning is provided by Tom M. Mitchell [14]: “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E”.

The machine learning algorithms are divided into two big categories and a few small ones. The two largest groups are supervised and unsupervised learning.

- Supervised learning: the algorithm is presented with the inputs and desired outputs. It is also called learning with a teacher. The algorithm is asked to find rules and patterns between inputs and outputs.
- Unsupervised learning: the algorithm is given unlabeled data, and its goal is to find patterns in the data on its own.
- Semi-supervised learning: combines both approaches, part of the data is labeled and the rest is not.
- Active learning: this process requires a user to provide labels to a given data set.

## 4.2 Supervised learning

Supervised learning is a machine learning method, that takes labeled training data (these data have predetermined output values), and creates the input-output information of the system [13, 14]. This information is then used to provide ways to learn about the system or to predict new data values. The goal of the supervised learning algorithm is to create a function, that can provide output information for any new input. Supervised learning algorithms are often used for predicting sports results because the past statistical data can be easily transformed into the labeled training data for the prediction model. For example, a player who scored a lot of goals in the past should be predicted to score a lot of goals in the future too. The concept of solving a supervised learning problem is illustrated in Figure 4.1. This is the optimal workflow to solve a supervised learning problem:

- Developer needs to determine, what types of data he will use in the dataset.
- Then, the training set needs to be created, it needs to be from the real world and it should represent a large variety of data.
- Features, to be used in the prediction model, need to be determined. They can be directly taken from the dataset, or created as a function of multiple data values. The feature selection is one of the most complicated tasks.
- Then, the learning function needs to be determined. The selection may vary based on the particular problem, as every algorithm works best for different kinds of problems.
- The algorithm then must run on the training set. Developers may need to tweak some model parameters to improve performance.
- The accuracy of the algorithm is then evaluated on the test set. This set must be separate from the training set.

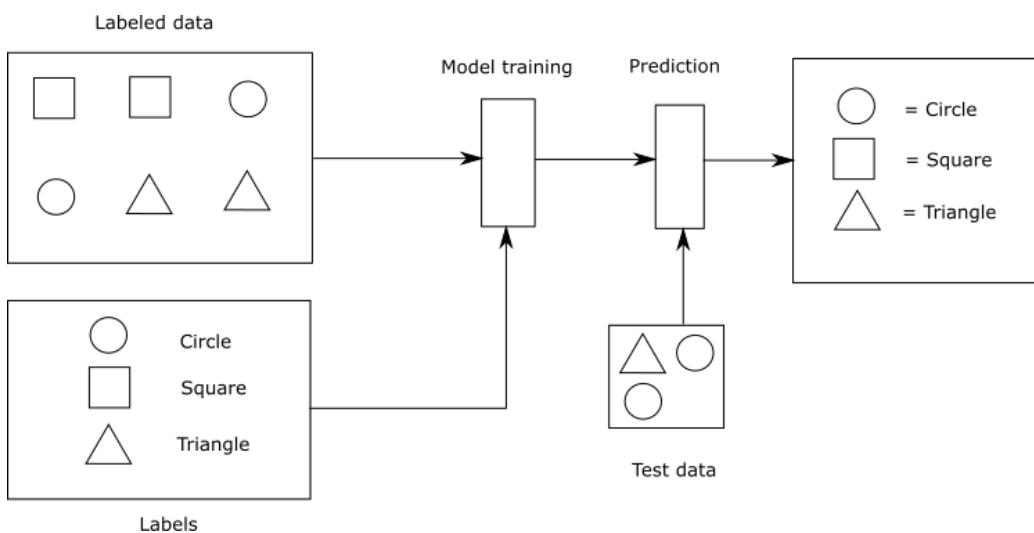


Figure 4.1: Concept of solving a supervised learning problem. The model is trained on the labeled data, tested on the test data, and then it makes predictions based on learned information.

The main advantage of supervised learning is that the output classes or values are usually made for humans to understand. That's one of the reasons, why it is often used. On the other hand, the biggest disadvantage is sometimes the difficulty to obtain correct labels for all the available data. The reason for this may be the cost or time. Also, not every data can be labeled, some abstract concepts can have ambiguous values for humans.

There are two types of supervised learning problems:

- Classification: the model is predicting a discrete value. For example, it is used to categorize inputs into subsets (image classification – is the animal on the picture a cat?, email classification – should the email be put in the spam folder?). A typical example is a decision tree.
- Regression: the model is predicting a continuous value, typically how the value will evolve in time (based on past data, what will be the weather in the next few days?). A typical example is a neural network.

In this thesis, supervised learning will be used to train the model to predict the future performance of a player. If the model predicts the excellent performance of a player and he will perform badly, in the next iteration of learning it will be taught to adjust the way it calculates the prediction. The regression models will be used, as the algorithm will predict the expected points for every player, given his past statistical data.

### 4.3 Unsupervised learning

Unsupervised learning is another concept of how to approach the machine learning problems. This approach is useful when there is little or no information about how the results should look like. There is no feedback about the observations because there is no teacher to provide error flags. There are many ways to solve an unsupervised learning problem, such as clustering, expectation-maximization algorithm, or method of moments.

### 4.4 Regression algorithms

The regression algorithms are used to predict the output value based on the input value and learned input-output relationships. The result is represented as a continuous function, typically a real number. Here are few examples of the regression algorithms, for example linear, polynomial, or logistic regression.

#### Linear regression

Linear regression [24, 25] was the first type of regression to be studied in-depth because the models that depend linearly on their parameters were the easiest to visualize. In the linear regression model, the data is represented as a linear function. There can be one or multiple variables in the linear regression model. If there is only one variable in the model, it is called *simple linear regression* or *univariate linear regression*. The variable, that the model is predicting is called *dependent* variable and the others are called *independent* variables. A graphical example can be found in Figure 4.2. Despite its simplicity, it has many practical uses, for example in computer science, medicine and many more.

Consider that there is a univariate linear regression model:

$$y = \beta_0 + \beta_1 x + \epsilon \quad (4.1)$$

where  $y$  is the dependent variable,  $x$  is the independent variable and  $\beta_0$  and  $\beta_1$  are model parameters. These parameters are called *regression coefficients*.  $\epsilon$  is the unobservable error. This function is then fitted to the training data. Based on how the line correlates with the data, the quality of the function can be determined. If the line fits well, the coefficients were picked well.

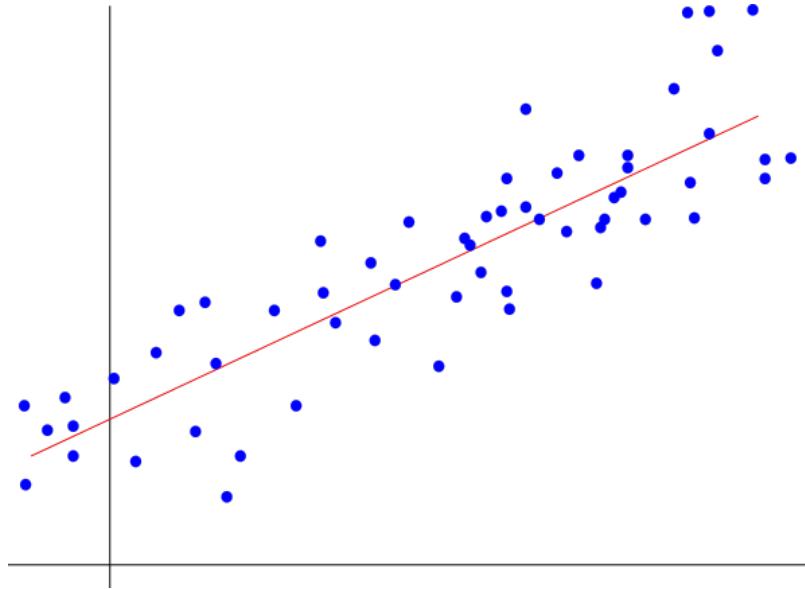


Figure 4.2: Example of an univariate linear regression.

When for every datapoint, there is more than one feature variable, it is called *multiple linear regression*. This is useful, when the output is dependent on more variables, for example in medical examination, the outcome depends on age, genetical information, diagnosis etc. The function that represents the multiple linear regression looks like this:

$$y = \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p + \epsilon \quad (4.2)$$

where vector  $(x_1, \dots, x_p)$  represents the features for each datapoint and as before,  $y$  is the dependent variable,  $(\beta_1, \dots, \beta_p)$  are model parameters and  $\epsilon$  is the unobservable error.

The values of  $(\beta_1, \dots, \beta_p)$  are generally unknown in practice and they need to be determined as a part of the model creation. To know these parameters,  $n$  pairs of observations need to be collected and then used to describe the unknown parameters. There are various methods to estimate these parameters, the most popular are the method of least squares and maximum likelihood.

## Polynomial regression

For some problems, the data cannot be easily interpolated using a straight line, but the relationship between the variables is curvilinear. The regression algorithm that solves these problems is called *polynomial regression* [23]. The polynomial regression function looks like this:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_k x^k + \epsilon \quad (4.3)$$

where the notation is the same as in the linear regression. Linear regression is therefore a special case of polynomial regression, where the degree of a polynomial (largest exponent) is 1. The difference between the polynomial of degree 1 (linear regression) and larger polynomial can be found in Figure 4.3.

The polynomial models are used to describe complex relationships between the variables. The main disadvantage of them is that for very large problems, the polynomial function can be very difficult to find, and the model can be difficult to train very well.

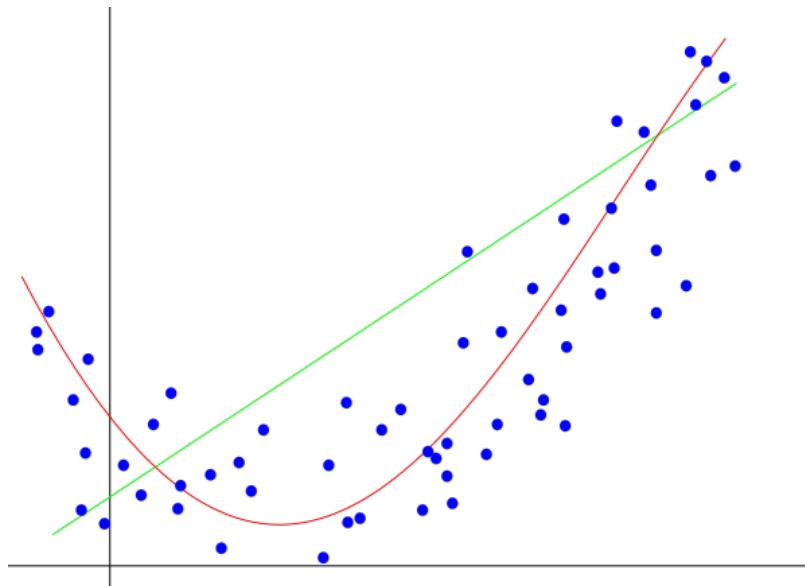


Figure 4.3: The difference between the linear and polynomial regression function. The green line is the linear function and the red line is the polynomial function. The blue dots represent the data variables.

## 4.5 Neural networks

Neural networks [4, 16] are an example of another concept of how to tackle machine learning problems. They come from the field of biology where every brain cell (neuron) takes input data, processes them, and sends them to the output (another neuron). All the brain cells are interconnected and they pass information between each other. Artificial neural networks in computer science work on a similar principle, they represent the software implementation of the brain structure. Neural networks are now applied to a large number of problems in many scientific areas, engineering, medicine, or financing.

## Artificial neural networks

Artificial neural networks are used to mimic the human brain inside the computer, they are used to process the information in the same way that a human would. The human neuron receives information from other neurons via *synapses* on its *dendrites*, and sends the output information to other neuron via the *axon* fibre. In artificial neural networks (ANNs), this behavior is defined differently [11]. The neurons are called *nodes* or *units*. All the synapses (inputs) are modeled with *weights*, so each input can be multiplied by a weight before processing. All the weighted input signals are then added together to supply a node *activation*. When the activation value is higher than the threshold value, the unit produces the signal that means that the unit is activated. Every unit can produce one of the two values: high or low value, 1 or 0, True or False. High value can only be outputted, when the activation function returns value higher than the threshold. This threshold can be different for every unit. The unit can be represented as a function (4.4), where  $K$  is an activation function,  $i$  is the number of inputs and  $w_i$  is a weight of a input with an index  $i$ .  $f(x)$  represents the output of the neuron. The example unit is also illustrated in Figure 4.4.

$$f(x) = K \left( \sum_i w_i \right) \quad (4.4)$$

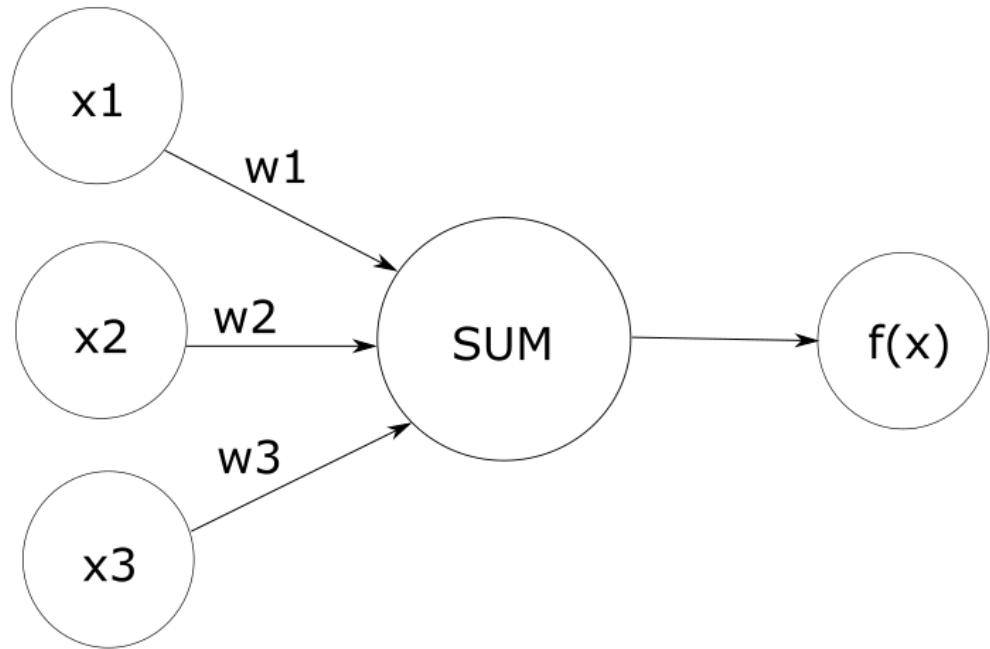


Figure 4.4: Example neural network unit, based on definition in function (4.4).

The term neural network refers to a system of artificial units. The connections between the units vary for every type of neural network. Not every unit must be connected to all the other units. One example of an artificial neural network can be found in Figure 4.5, where each unit is represented as a circle, with implicit weights. Units are arranged in layers, where the input signal travels from left to right. The input layer represents the signals, that are fed to the network. In the hidden layer, that can be one or more, the inputs are

modified using the weights of the inputs, and the new value is sent to the output layer. Output can be also weighed and modified. This is called a **feedforward** pattern, but there are more available.

For example, with **backpropagation** pattern, the signal can loop back to the previous layers of the network. There is also no limit to how many units can be a part of the network. The number often depends on the type and difficulty of the problem, that the neural network is aiming to solve. The number of input units is often dependent on the training set. Too many units in the hidden layer can lead to very long computations, but too few can prove ineffective in solving the problem.

There is also no right way of how to choose the weights of the input and output layer units. At first, the weight should be chosen randomly, or semi-randomly, and then adjusted based on the results that the neural network provides. After each iteration of the computations, the weight should be adjusted to better represent reality.

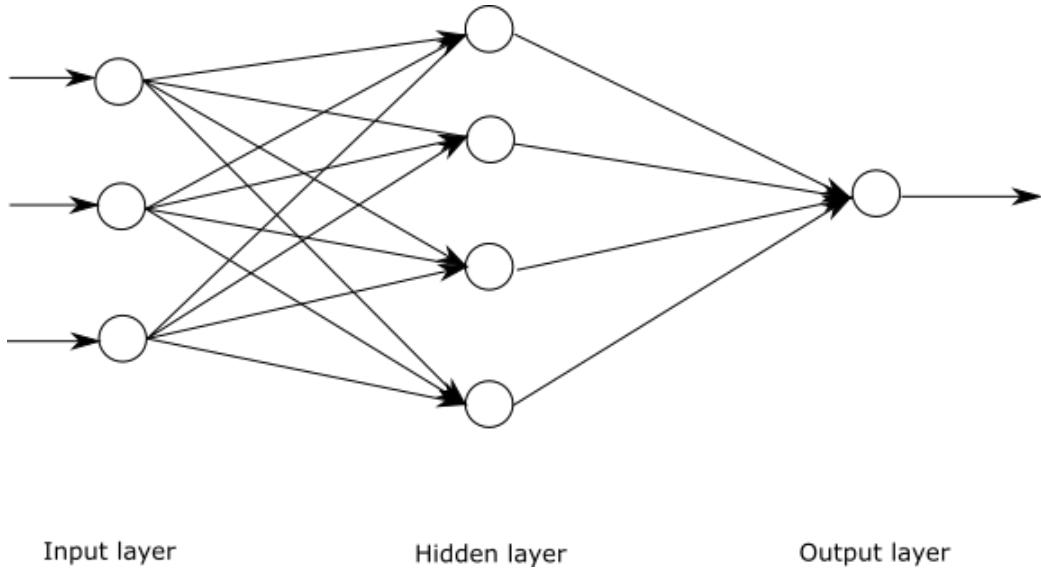


Figure 4.5: Simple artificial neural network with one input layer, one hidden layer and one output layer

## Recurrent Neural Networks

Recurrent neural networks [3] [27] are a subclass of neural networks that have connections between nodes in the hidden layer, that go backwards in layers. This concept is illustrated in Figure 4.6. They allow using computed results mid-processing to calculate more precise results. If the network is given a sequence  $x = (x_1, x_2, \dots, x_T)$ , it updates its hidden state by a function:

$$h_t = \begin{cases} 0, & t = 0 \\ \Phi(h_{t-1}, x_t), & \text{otherwise} \end{cases} \quad (4.5)$$

where  $\Phi$  is a nonlinear function and update of the state is done by function:

$$h_t = g(Wx_t + Uh_{t-1}) \quad (4.6)$$

where  $g$  is a logistic sigmoid function and  $W$ ,  $U$  are weight matrices. The logistic sigmoid function is used as an activation function in this example. For every unit in layers except the first one, the hidden state of the previous layer is also used. These networks are used mainly in speech recognition, but they can be used to spot a pattern in a long continuous stream of data. The concept of recurrent artificial neural networks is useful in making predictions for time-series data, such as statistics of players in a long sequence of game weeks.

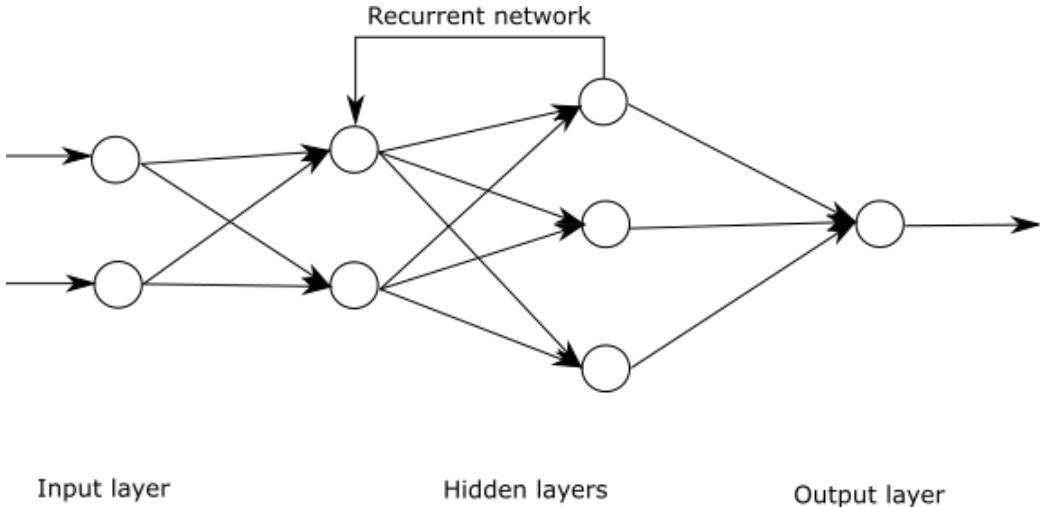


Figure 4.6: Recurrent neural network with one input layer, two hidden layers and one output layer.

The recurrent neural networks are making use of the backpropagation pattern mentioned earlier. It is used to teach the neural network by providing feedback. When the output produced by a particular unit is compared to the output it was meant to produce, the difference can be propagated back to the hidden and input units as a change in weights. So the backpropagation forces the neural network to learn, by reducing the difference between correct and actual outputs. This process is called training a neural network. The bigger the difference between the outputs is, more changes need to be made in the weights in the hidden and input layers.

Once the network is sufficiently trained and produces as few mistakes as possible on the training data, it is presented with the new data – the testing set. This is a completely new set, that it has never seen before, and it should provide similar results as with the training set. Depending on how well it is trained, the results may be different.

## 4.6 Time series models

In regular machine learning datasets, time does not play any role. The training dataset is a collection of observations, either labeled or not. But in the time-series dataset, there is an explicit order of the observations. This order is a time dimension. The number of input nodes corresponds to a number of observations in time [28]. This concept is used to predict outputs in the future, based on past observations. Time series in this thesis is used to order past player data so see how a player performed over time and can be used to spot patterns in player's statistics.

There are two methods [28] that can be performed on the time series data. First is time series analysis [2], this method is used to develop models for describing and understanding the data. This can be useful when developing statistical models about a particular dataset.

The second method is the time-series forecasting, which is used to make predictions from existing past data. Forecasting models are being taught on historical data to make future observations. The very important thing is to not use any data from the future for training the model, it must be completely estimated and predicted by the model. Time series forecasting is best done using the Recurrent neural network described in the previous section. The recurrent neural networks process time series step by step and use they computed state in the next iteration.

## 4.7 Underfitting vs Overfitting

Underfitting and overfitting the model are two very common problems, that can happen while working with prediction models. The model is underfitting when the accuracy is higher on the validation set than the accuracy on the training set. Also, when the whole model is just performing badly, it is called underfitting as well. Underfitting can happen when a model is too simple or too regularized, so it cannot learn much from its training data.

Overfitting is the same thing in reverse, when the model fits well on the training set, but performs badly on the validation set. It happens very often, that the model recognizes specific patterns in the training set, not the general ones.

The term “Goodness of Fit” [12] refers to how good are the model’s predicted values compared to the “true” values. The model that has learned just on the noise on the training set, and cannot recognize noise in general, is considered as overfitted. The recommended way of developing prediction models is always to start with the simple models and adding more complexity later while testing for overfitting all the time. Both overfitting and underfitting are illustrated in Figure 4.7.

Just detecting the overfitting is not enough, there are several ways to reduce overfitting of the model:

- Cross-validation: the idea is to split the training data to generate mini train-test groups, and to use these groups to improve the model.
- Adding more data: when it is possible to collect them, more data can significantly improve the model. But it is not true for every problem. If there is more noise in the data, it can worsen the performance as well. In some problems, getting more data is also not possible.
- Using data augmentation: it is always good to include some data augmentation to the dataset if it is possible. It could include rotating or zooming the images, adding color filters, and more. It is recommended to use augmentation only on the training set.
- Removing features: sometimes some features don’t fit into the model well and removing them can improve its performance.
- Regularization: this one depends on the used learner, but it can be something like using dropout on a neural network or pruning a decision tree.

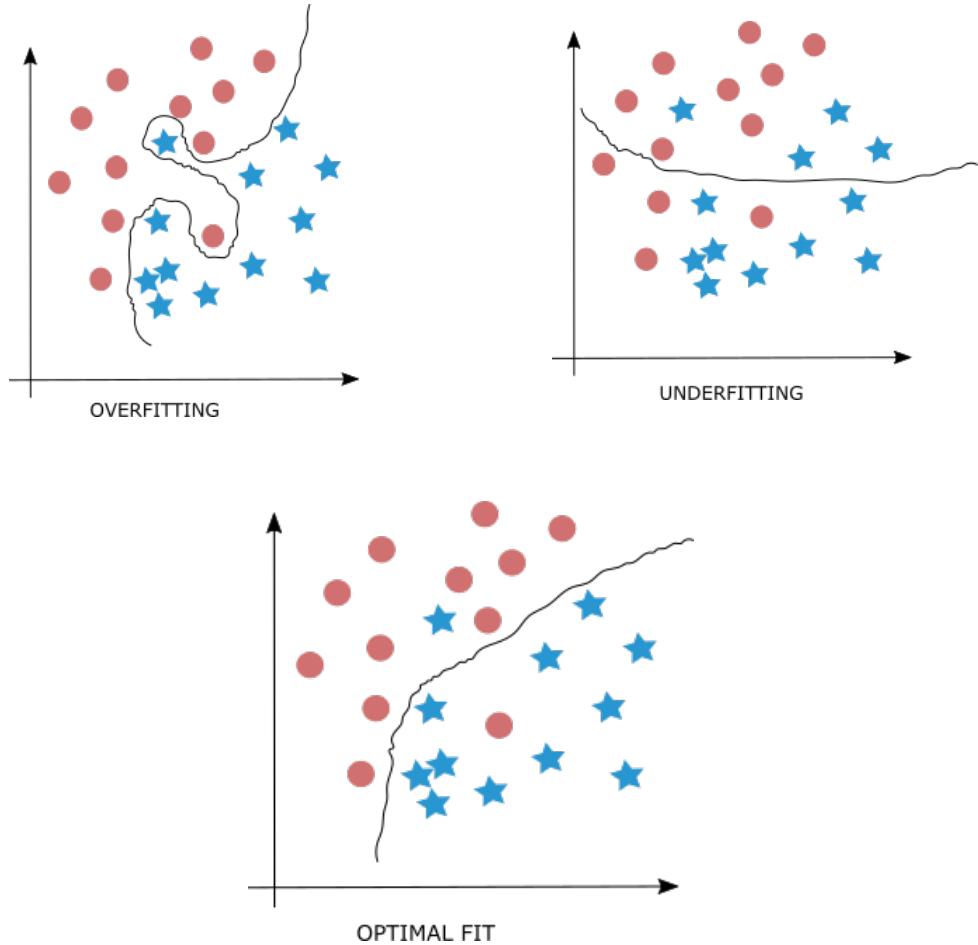


Figure 4.7: Overfitting, underfitting and optimal model examples.

## 4.8 Algorithm selection

There are many ways how to utilize machine learning in applications. Many programming languages offer libraries to implement machine learning. Currently, most often used languages for machine learning are Python and R. Other languages used are C/C++, Java, and JavaScript [5]. For purposes of this thesis, the Java language was chosen as the best candidate. There are a lot of Java-based libraries for machine learning. Many of them are general-purpose but some have specific usages. Neural networks with time-series input data were chosen as the best fit to represent the historical data of football players. It is not the purpose of this thesis to implement the neural network from scratch, but rather to utilize existing solutions, and test various combinations of input data and predicted values. Here is a list of available Java libraries, that provide the options to implement what is needed for this thesis, at least to some degree.

## WEKA

WEKA<sup>1</sup> is the most popular library for machine learning in Java. It is mostly used for data mining, data analysis, and prediction models. It is available under GNU General Public licence and it has a graphical user interface to directly import data, and also Java API for development. It also supports data preprocessing, classification, clustering, visualization, regression, and feature selection. It contains over 250 algorithms in total. WEKA also has advanced features to support long-running mining, experimenting, and comparing results.

The idea behind WEKA was to provide a uniform interface to the collection of machine learning algorithms. Its biggest strength lies in classification, so applications that require automatic classification can benefit from it. It is implemented in Java but has packages that enable the use of code in Python or R. It requires Java 7 in the latest version. WEKA supports all common file formats, but also features its own, ARFF, to describe data in attribute-data pairs. The first part is a header, which specifies attributes/features and their type: nominal, numeric, date, string. The second part is data, where lines correspond to instances. The last attributes in the header are the target variable or class. Missing data are marked with a question mark. An example ARFF file can be found below.

```
@RELATION person_dataset

@ATTRIBUTE 'Name' STRING
@ATTRIBUTE 'Height' NUMERIC
@ATTRIBUTE 'Eye color'{blue, brown, green}
@ATTRIBUTE 'Hobbies' STRING

@DATA
'Bob', 185.0, blue, 'climbing, sky diving'
'Anna', 163.0, brown, 'reading'
'Jane', 168.0, ?, ?
```

Since version 3.7.3 WEKA has dedicated time-series environments to develop and evaluate forecasting models. It can be installed as a plugin to the WEKA GUI. This time series framework first transforms the data into a form, that can be processed by a standard propositional learning algorithm. It removes temporal ordering by encoding the time dependency as additional input fields. These fields are called **lagged variables**. After the transformation, any of WEKA's regression algorithms can be applied to learn a model. This approach is often more powerful than classical statistical techniques. An example of the time series plugin in WEKA GUI can be found in Figure 4.8. This library was chosen as the best available and will be used in the implementation.

## MOA

MOA<sup>2</sup> is an open-source software used for data mining and data stream analysis in real-time. For solving a complicated problem it is often combined with WEKA. Its collection of machine learning algorithms are useful for regression, classification, outlier detection, clustering, recommender systems, and concept drift detection. It can be useful for large evolving datasets. It aims for time and memory-efficient processing.

---

<sup>1</sup><https://www.cs.waikato.ac.nz/ml/weka/>

<sup>2</sup><https://moa.cms.waikato.ac.nz/>

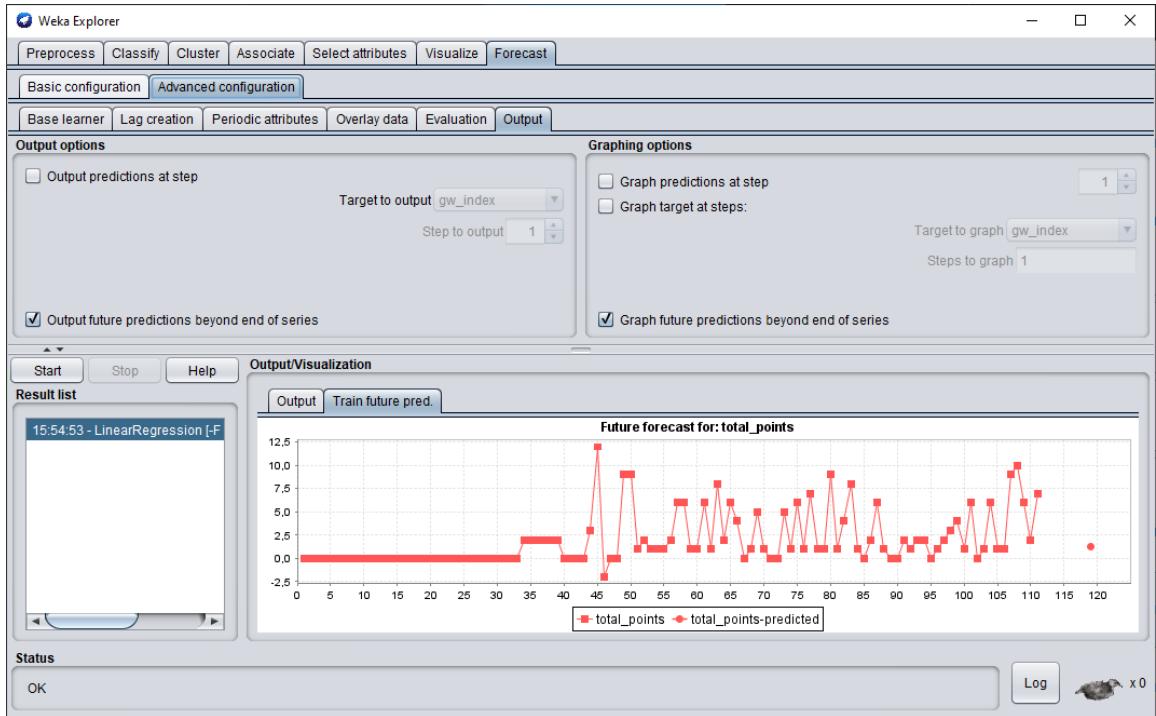


Figure 4.8: Time series package in the WEKA GUI.

## Deeplearning4j

Deeplearning4j<sup>3</sup> is a commercial, open-source deep-learning library in Java and Scala. It mainly serves programmers working on Hadoop, the massively distributed storage system with big processing power and the ability to handle limitless concurrent tasks. Deep neural networks are capable of pattern recognition and goal-oriented machine learning. It is very useful in identifying patterns and sentiment in speech, sound, and text. It is also used for detecting anomalies in financial transactions. It is used more for business than for research.

## Mallet

Mallet<sup>4</sup> is an open-source Java machine learning toolkit, for analyzing text and language. It supports natural language processing, clustering, document classification, information extraction, topic modeling. It is available for free and can be used commercially.

## ELKI

ELKI<sup>5</sup> is another open-source software in java for data mining. It is focused on unsupervised methods in cluster analysis, database indexes, outlier detection. It allows independent evaluation of algorithms and data management tasks. It was designed primarily for researchers and students and provides a lot of configuration parameters. This allows for easy benchmarking.

<sup>3</sup><https://deeplearning4j.org/>

<sup>4</sup><http://mallet.cs.umass.edu/>

<sup>5</sup><https://elki-project.github.io/>

## **Java ML**

Java machine learning library<sup>6</sup> is a collection of algorithms with a common interface. It only features java API. It contains algorithms for data preprocessing, feature selection, classification, and clustering. It features several WEKA bridges to access WEKA algorithms. It offers more consistent interfaces than WEKA and implementations of some algorithms that are not present in other packages. It is available under the GNU GPL license.

## **JSAT**

This tool<sup>7</sup> has one of the largest collection of machine learning algorithms. It is pure Java, has no external dependencies. It supports parallel execution, so it is great for research purposes. It is very fast for small and medium-size problems.

---

<sup>6</sup><http://java-ml.sourceforge.net/>

<sup>7</sup><https://github.com/EdwardRaff/JSAT>

# Chapter 5

## Design of the application

This chapter contains the proposed design of the web application for Fantasy football performance prediction. In the first section, there is a list of requirements for the application. These requirements were created based on findings from existing prediction applications in Chapter 2. Next, there are designs for application architecture, data fetching flow, and graphical user interface.

### 5.1 Requirements for prediction application

These are the minimal high-level requirements for the application. Based on these requirements only, the application could be built on any platform. For this thesis, the application must consist of the following parts:

- information system with prediction model and player API,
- graphical user interface,
- data fetching scripts from Fantasy Premier League API.

#### Functional requirements

- The prediction model will be trained on historical data fetched from Fantasy Premier League (FPL) API. The datasets containing more detailed historical data should be used as well.
- The information system will contain the trained prediction model and provide methods to manipulate and query the model via the API calls.
- There must be a way to retrain the model, when there are newer historical data available.
- The application will provide a way for user to import his current team and optimize it using the prediction algorithm.
- The application will have a graphical user interface (GUI) that will access the information system.
- The GUI will have the option to import user's existing FPL team via his FPL credentials. The application must verify user's identity and download only his team.

- The GUI will offer an option to input the team manually without logging in.
- The GUI will display the user's existing team and offer options to manipulate and save it.
- The GUI will have the option to display predictions created by the prediction model based on the user's existing team. Users will have options to use the predictions to transfer in or out players in his team. They will have options to display predictions for his captain and vice-captain picks. The user will have multiple options on how to adjust predictor settings, such as the number of weeks to the future, prediction technique, etc.
- The system will display predicted points tally for the user's current team for the next round, or multiple rounds if selected.
- The GUI will have sections dedicated to the injury and suspension tables, and a section with detailed player statistics.
- The GUI will show a current transfer market for the FPL application, with statistical data about total points and information about player prizes.
- The GUI will provide overview of historical data for every player, for each of the past seasons.

## 5.2 System architecture design

The architecture of the application will have the following 3 main parts: server, client and a data fetching scripts. The architecture is illustrated in Figure 5.1.

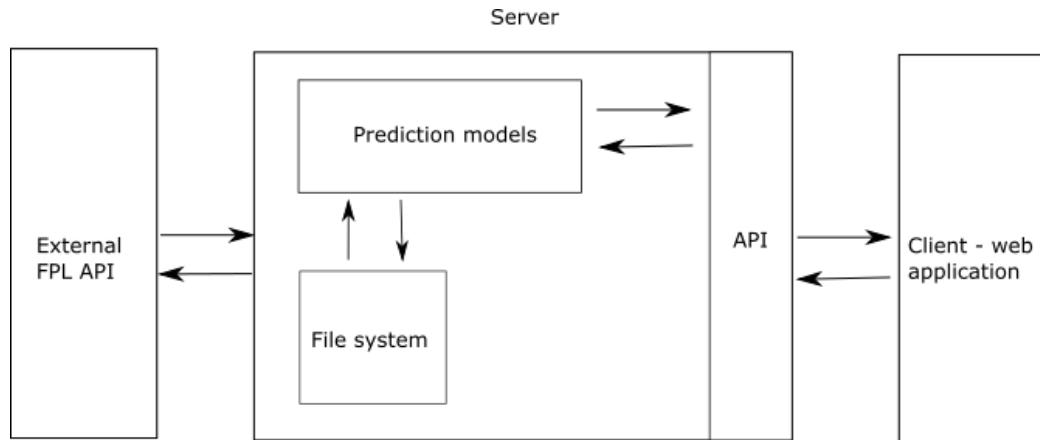


Figure 5.1: System architecture with server, client and data fetching parts. Client and server are communicating through the exposed API.

- Server: backend component, built as a standalone application with API exposed to the client. It will contain prediction models and methods to access them. It will also contain a file system for storing player data. It will contain options for administrator to recreate datasets and create new predictions if needed.

- Data fetching component: it will run on the same server as the main backend component. It will be used to fetch data from the FPL API every game week, format and filter them and to store them in the dedicated file system. This fetching will be triggered manually by the administrator.
- Client: web application, the main GUI part of the application. It will offer functionality to the user using API calls for the server.

### 5.3 Data fetching and processing flow

All the data used in this thesis are collected from two resources. The new data are fetched from the official FPL API<sup>1</sup> by a custom script. This script then formats and filters the data and stores them in a player data file system on the server. From this API the script can get all of the data for the current game week, and also statistical data about teams and players from the current season. The problem with this resource is, that the data are erased before every season. However, there is a project on Github<sup>2</sup> dedicated to collecting historical data from this API, which is the second data resource used. This project also provides scripts for scraping data from API in Python. The script in this thesis is inspired by them. The flow of the data fetching part of the application can be found in Figure 5.2. This data fetching script will be used in the future for getting more data. In this project, it will be just implemented and not used, and it will be enabled before the start of the new season. For this thesis, historical data from the mentioned Github project are enough.

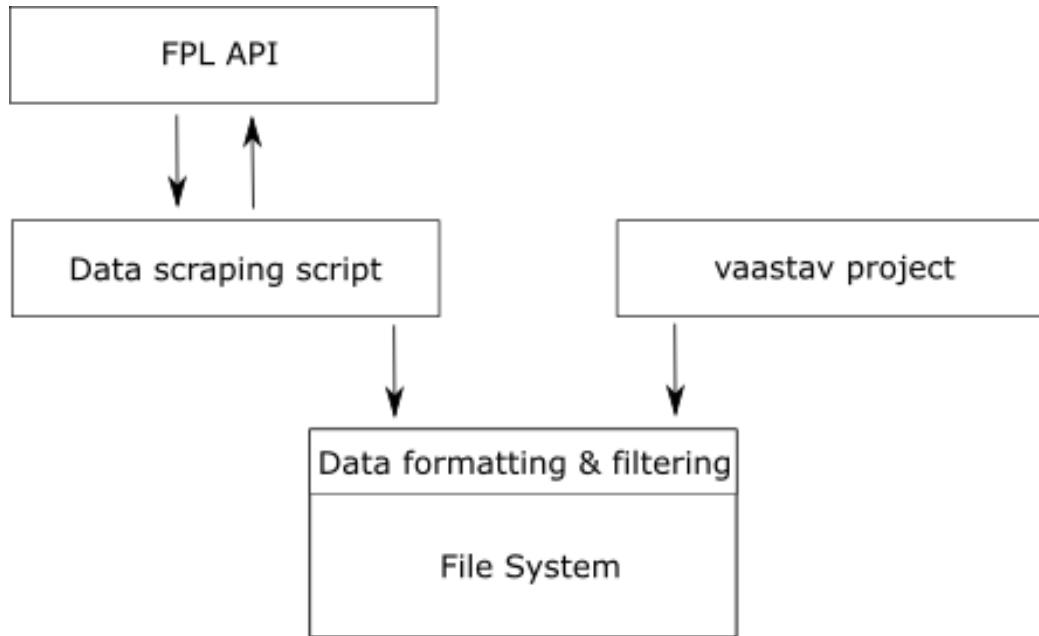


Figure 5.2: Flow of the data fetching from the FPL API. It includes one time import of the historical data from the vaastav Github project mentioned above. After data is fetched from the API, it is filtered and formatted according to the needs of the prediction model.

<sup>1</sup><https://fantasy.premierleague.com/api/bootstrap-static/>

<sup>2</sup><https://github.com/vaastav/Fantasy-Premier-League>

## 5.4 Dataset analysis and feature selection

To accurately predict the performance of every player, their historical statistics are naturally considered. How a player performed in past matches can be a good indicator of future performance. However, there are more things to consider, such as the history of their opponents. Player is more likely to have a good game against a poor defense. Also injuries and suspensions, fixture congestion and weather can have an impact on player performance. Also, teammates need to be considered when predicting a player's performance. When a star defender is injured, the team is likely to concede more goals in a game.

The Fantasy Premier League API returns a lot of data, but there are two main objects: **teams** and **elements**.

**Teams** is a list containing information about the clubs currently competing in the Premier League. There is a club name, id, and values for its strength. Strength was determined before the season concerning the players in the club, their position and results from last season, and the odds of their success. The strength attribute is divided into a few categories, such as strength at home, away, attack at home and away, defense at home and away. The object contains a lot of attributes, this is a summary of the most useful ones:

- There is statistical information about past matches, league position, and obtained points, they are useful to determine the current strength and form of the team.
- Information about the strength of the team, it is divided into multiple values, such as home and away, or attacking and defending strength. This information is also useful to predict how likely the team will win the match.
- The rest of the features are general and will be used just for presentation purposes in the web app, the example is club name or short name.

The second important object is **elements** object, which in this API means players. There are a lot of pieces of information about players here. Some examples are player name and team, position and price. Some interesting attributes that I will use in the prediction are how many people have this player selected, his availability status, his expected points this or next round, etc. The very interesting attribute is an ICT index<sup>3</sup>, which consists of influence, creativity, and threat. This is a list of all the attributes of the object used for prediction:

- The object contains various statistical information about the player's past performance. There is information about how many goals he scored and assisted, for goalkeepers how many saves he made and average points per game. These will be used for predicting future performance.
- There is information about the probability of the player playing the next game, it is based on injuries or past selection in the team. This will be useful when predicting long term selection of players in the model.
- For transfer market prediction and recommending transfers, the object contains information about player prize changes and current values.
- The ICT index attributes, that will be used to calculate the overall impact that the player has on the game.

---

<sup>3</sup><https://www.premierleague.com/news/65567>

- There is also some general information about the players that will not be used for prediction, just to display the values in the web application, such as player names and squad numbers.

These attributes, however, won't be enough for accurate predictions, that's why there is one more endpoint in the API, called element-summary<sup>4</sup>. It offers detailed player analysis based on past performance. It contains statistics for the current round, all the scores from past matches in the season and also historical data from past seasons, grouped into one statistic. This is a list of very useful attributes that the object provides:

- For every past fixture in the season, it provides information on who the opposition was and how many points did the player score against them in that match. This is very useful when making predictions and recommendations directly based on opposition. Some players may perform better against a particular team and these values should be taken into consideration.
- There is also data about whether the fixture was home or away. Many players perform better when playing in a familiar environment and these attributes will be useful for predictions as well.
- The object also offers detailed information about how many goals, assists, and cards the player had in that game, as well as some other statistics like how many minutes he played. These should be also considered when making the prediction model. From all of these statistics, the model will calculate useful predictions in the form of expected goals, non-penalty goals, expected points, expected goal difference, expected bonus points and more. These will be used to finalize the recommendations for player selection.

Lastly, this API has one downside, it does not contain teams that were relegated in previous seasons. That's why the past data have to be precisely merged to avoid confusion and incorrectness.

## 5.5 Design of the GUI

In this section, there are designs of all the main screens in the web application. Designs generally show the most important functionality and layout on the respective pages, they do not show all of the control and UX elements on the pages.

In Figure 5.3 there is a mockup design for a main dashboard screen. In Figure 5.4 there is an “Edit Squad” screen with transfer market and squad view for user to edit. In Figure 5.5 there are designs for two overlays. The overlays are for optimizations options and for actions on squad screen. In Figure 5.6 there is a “Squad Optimization Overview” screen. It shows the recommended team selection and some other options for maximizing the points. In the Figure 5.7 there is a design for a “Statistical Overview” screen, with tables for various player statistics.

These designs are not final, and they can be changed during implementation, when the more obvious and user-friendly design is created. These designs provide all of the functionality that is needed to successfully fulfill the requirements.

---

<sup>4</sup><https://fantasy.premierleague.com/api/element-summary/{playerId}/>

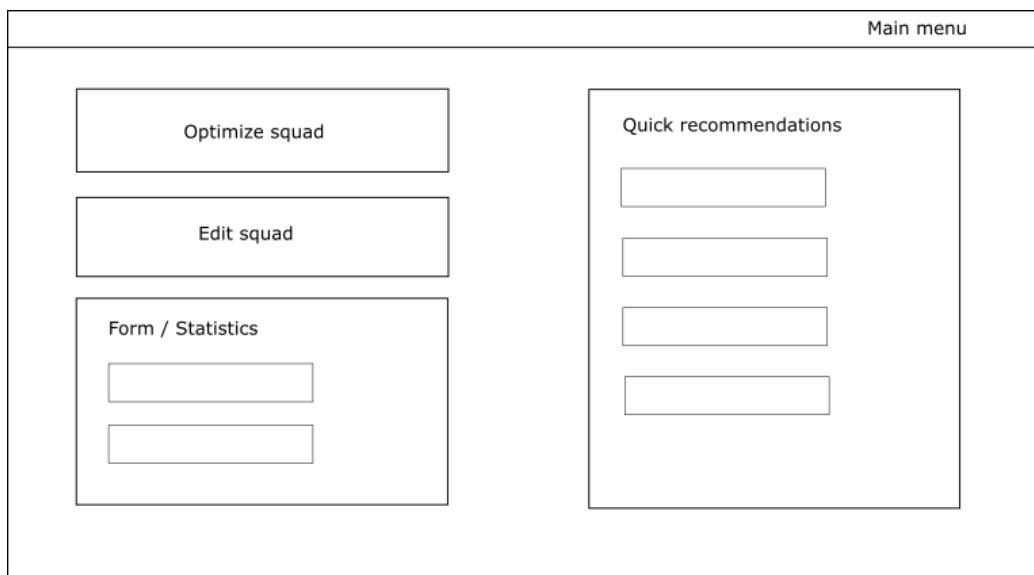


Figure 5.3: Main “Dashboard” screen design. It contains options to enter different subpages of the application. The “Optimize squad” button takes user to the “Optimize” screen where he can select various options to improve his squad by the prediction algorithm. “Edit squad” option offers option to edit the team manually. “Form/Statistics” option shows some of the popular statistics and “Quick Recommendations” windows shows some quick tips about the squad.

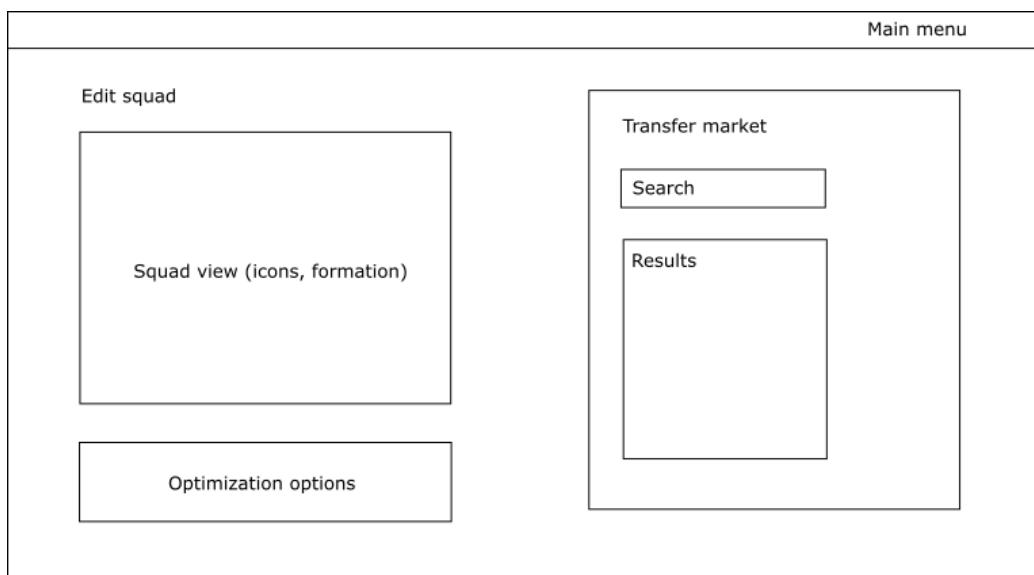


Figure 5.4: “Edit Squad” screen design. It offers options to view the current squad with editing options, some optimization options and a transfer market, where user can manually select and buy players for his team.

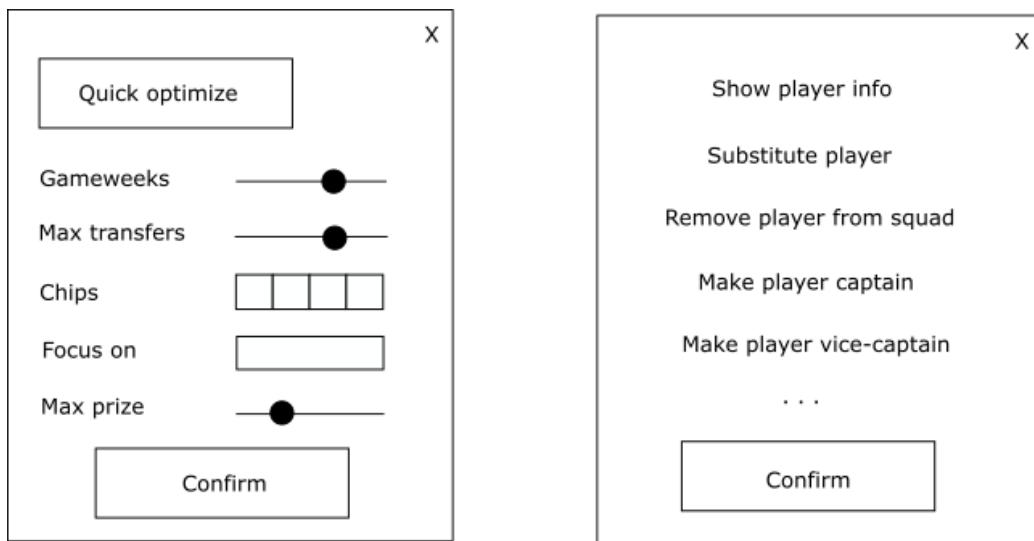


Figure 5.5: Designs for the two option overlays. On the left side, there is an overlay design for extended optimization options. It offers multiple options for a prediction algorithm to optimize user's squad according to his requirements. On the right side, there is an overlay in the squad screen, it offers options related to the player and the current squad. These options could vary for each player.

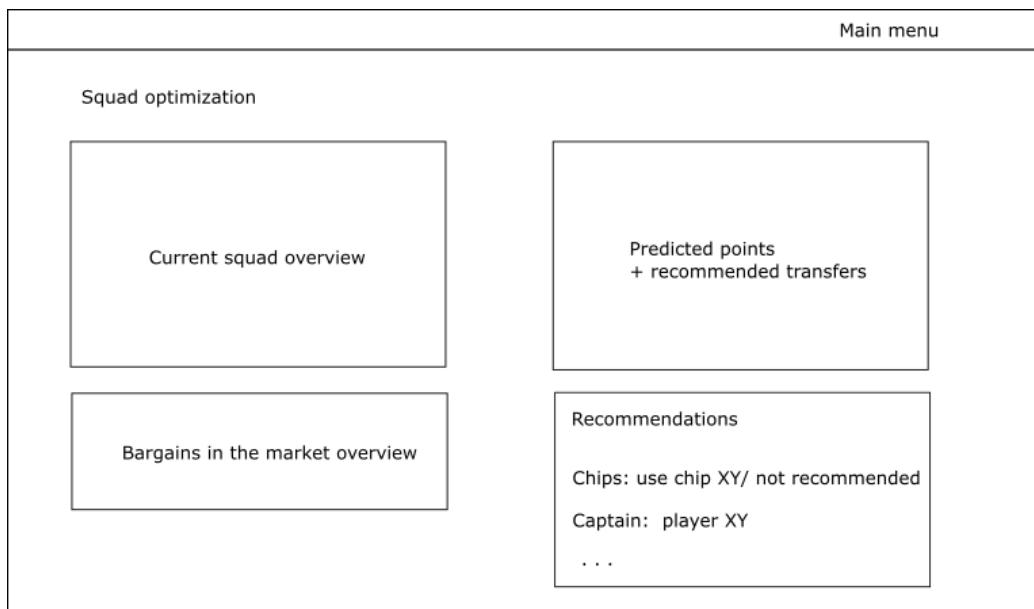


Figure 5.6: “Squad optimization” screen design. There is a view of the current squad and the recommended squad with predicted points. There are also recommendations for some bargains on the transfer market worth seeing. This screen also has sections for recommended chips usage and more options.

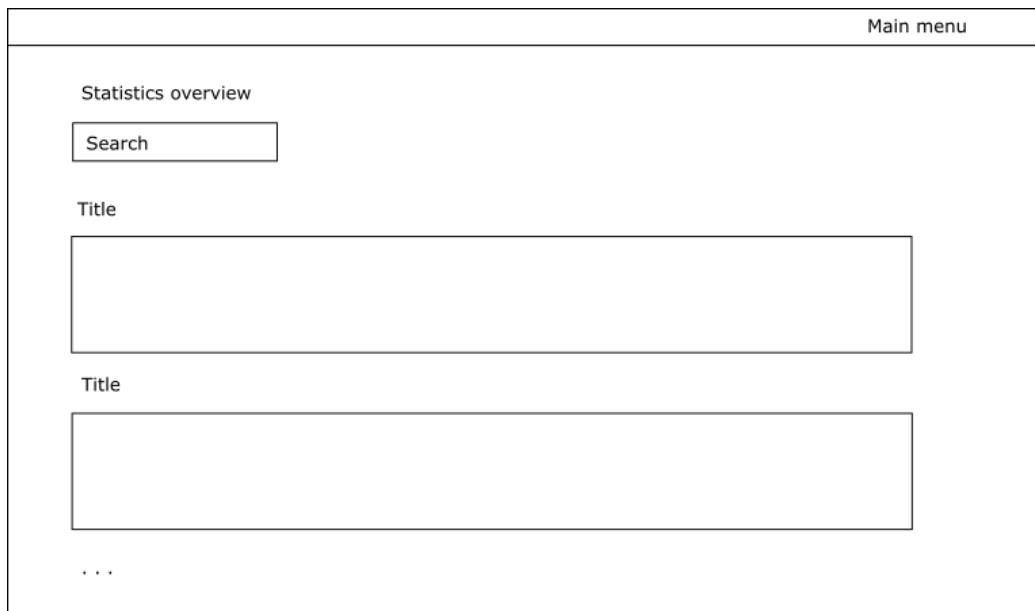


Figure 5.7: “Statistics overview” screen design. It offers options to browse some statistics about the game, past game weeks, most picked players, player statistics and more.

# Chapter 6

# Implementation

This chapter describes all the necessary parts of the implemented application. Firstly there is a focus on the server part – its architecture, description of models, and API. The next part describes the implemented neural network, with the implementation processes and problems. The last part is about the web application, how it changed from the original design, and what it can do.

## 6.1 Application server

The application server was created using Spring Boot<sup>1</sup> configuration. This was selected because it can accelerate early development and it generates a lot of things automatically. The application is built using Gradle Wrapper<sup>2</sup> which has many benefits over other building tools, mainly it sets up a lot of project files on its own. The Gradle build file contains all the dependencies needed to run the project: all Spring dependencies, WEKA Java API library<sup>3</sup>, annotation processors, JSoup<sup>4</sup> library for HTML parsing and more. This server contains REST API that is used by the web application to fetch the data. It also contains services to manipulate dataset, create, and use the prediction model and output data.

### Server architecture

The server is using hybrid architecture, where there is no presentation and no database layer. The presentation is handled by the web application, which is described in Section 6.6. It is configured to run only locally, on the same IP address as the web application. There is also no database, as the authentication runs against the FPL API and there is no need to store any user's information. Also, the player data do not need to be stored in the database, the files in the server filesystem are enough. The server consists mainly of 3 parts:

- Controller part: it contains all available API endpoints, that can be used. There are endpoints for loading player and team data, fetching predictions and optimizing user's squad, for logging and authenticating users, and some administrative endpoints for the server administrator to manipulate dataset.

---

<sup>1</sup><https://spring.io/projects/spring-boot>

<sup>2</sup>[https://docs.gradle.org/current/userguide/gradle\\_wrapper.html](https://docs.gradle.org/current/userguide/gradle_wrapper.html)

<sup>3</sup><https://weka.sourceforge.io/doc.stable-3-8/>

<sup>4</sup><https://jsoup.org/>

- Service part: this is where most of the logic is happening. There are services for manipulating the dataset, for reading and writing to files and for communicating with the FPL API. It also contains the WEKA prediction models.
- Data part: the application server contains the dataset and the scripts for downloading newer data. Also, predictions and player statistics are stored on the filesystem.

The detailed tables of all endpoints used in the application server can be found in Tables 6.1, 6.2, 6.3 and 6.4. The server was designed to react to the changes of the dataset, that will happen after every game week. When every round is finished and new data are downloaded to the server, there are API endpoints available for the administrator, to regenerate the datasets used in the prediction models, and to create new predictions based on updated data. These changes can happen without the need to reload the server. The other option would be to create cron jobs to regenerate dataset, but they are unreliable in the changing world of English Premier League, where many fixtures are postponed and canceled often.

Table 6.1: Login controller. It handles operations regarding authentication to the web application.

Endpoint URI	HTTP Method	Description
/api/login	POST	Handles login logic using email and password.

Table 6.2: Dataset controller. Handles all operations with the dataset.

Endpoint URI	HTTP Method	Description
/api/dataset/init	GET	Handles dataset filtering and cleaning.
/api/dataset/divide	GET	Handles dividing dataset to training and testing sets.
/api/dataset/predict-all	GET	Handles creating prediction model for every player.
/api/dataset/stats	GET	Handles creating predictions based on past statistics.

Table 6.3: Team controller. Contains endpoint for getting information about teams in the Premier League.

Endpoint URI	HTTP Method	Description
/api/team	GET	Returns data about every team.

Table 6.4: Player controller. Handles all operations regarding players in the game, and the predictions.

Endpoint URI	HTTP Method	Description
/api/player	GET	Returns data about every player.
/api/player/ids	GET	Returns ids of all players.
/api/player/projected-points	GET	Returns projected points for every player.
/api/player/projected-points/{id}	GET	Returns projected points for specific gameweek number.
/api/player/detail/{id}	GET	Returns detailed historical data of a player.
/api/player/injuries	GET	Returns information about all unavailable players.
/api/player/optimize	POST	Using current squad and optimize parameters, returns optimized squad.

## Authentication

There is no authentication required to run the web application. It offers an option to manually input the team there. But if the user wants to import his team automatically, he has to log in with his FPL credentials. Using `requests`<sup>5</sup> package, the authentication runs on the server and it calls endpoint `https://users.premierleague.com/accounts/login/` with user's credentials and two static parameters:

- app: plfpl-web
- redirect\_uri: `https://fantasy.premierleague.com/a/login`

Then, the script receives required cookies, and uses them for the next request, to the URL `https://fantasy.premierleague.com/api/me/`, from where it gets user's team ID. The team ID is then used to call endpoint `/api/my-team/{team ID}`, from where the script gets required information about the user. The response contains following data:

- picks: this is an array containing all the players in user's squad. Every element of the array is an object, with details about player position, his current selling and purchase price, his captaincy status and whether he does have the bonus multiplier for the current round.
- chips: this is an object, that stores data about user's usage of his chips
- transfers: this object contains user's current squad value, his transfer limit and how many transfers he made for the current round

When the first endpoint returns an error, the authentication failed and user is presented with an error message.

---

<sup>5</sup><https://github.com/hsiafan/requests>

## Optimizing the squad

In Section 6.6, that describes the webserver, I mention the option to optimize the squad using various parameters. This option is implemented as a service on the application server. It receives all parameters provided by the user and then proceeds to optimize his squad. This is the process of optimization:

- First the service fetches predicted points for every player, based on selected parameter **number of gameweeks** since there are different predictions for the number of weeks ahead.
- Another parameter is **transfers**. If the user selected 0 transfers to perform, service just sorts the players in his team based on predicted points and then advises him on captain and vice-captain roles. This option is useful when the user is happy with his squad and wants to save his transfer token for the next round.
- If the user selected 1 transfer to perform, the process is more complicated. For every position in his team, the worst performing player is selected. Then, from the pool of free players, that can fit inside the player's budget, all eligible players are picked and sorted based on predicted performance. The sorted squad then needs to be adjusted for the FPL rules, so the match squad contains 1 goalkeeper, at least 3 defenders, and 1 forward. The rest of the team is then built based on the predicted points and the worst players of the 15 selected are placed on the bench. Then, the best options are picked based on user-provided parameters. The biggest factor is the proposed increase in predicted points. Another rule that the algorithm needs to have in mind is that a maximum of 3 players from any team can be in the user's squad.
- If the user selected two transfers, the process is similar to the previous point, but two players are selected from the whole squad. The transfers must still fit into the budget and the number of players in each position must remain unchanged, so the calculations in these options are more complicated. Here the options of players to remove/add to the team are increased because now the algorithm needs to select from these possibilities: 2 goalkeepers, 2 defenders, 2 midfielders, 2 attackers, and combinations of 1+1 from every couple of positions.

The optimized team must adhere to multiple constraint rules set by the FPL application:

$$\begin{aligned}
 \sum_{i=1}^n c_i * y_i &\leq 100.0 \\
 \sum_{i \in G} y_i = 2; \quad \sum_{i \in D} y_i = 5; \quad \sum_{i \in M} y_i = 5; \quad \sum_{i \in F} y_i = 3 & \\
 y_i \in \{0; 1\}; \quad 1 \leq i \leq n &
 \end{aligned} \tag{6.1}$$

where  $n$  is the total number of players,  $c_i$  is the cost of the player, with index  $i$ ,  $y_i$  is a binary value, that indicated whether player with index  $i$  is included in the squad or not,

$G$  is a set of all goalkeepers,  $D$  is a set of all defenders,  $M$  is a set of all midfielders and  $F$  is a set of all forwards.

Next, I implemented the algorithm, that creates a valid team from all the players in the squad. It gets an unsorted group of 15 players and proceeds to pick 11 players for the starting lineup. These 11 should be the best possible combination from his whole squad, while following the FPL rules. The algorithm works like this:

- The array of 15 players is sorted by the predicted points.
- The flags for minimum and maximum possible number of players in the squad are set: minimum(1 GK, 3 DF, 2 MD, 1 FW), maximum(1 GK, 5 DF, 5 MD, 3 FW).
- The flags for current number of players in each position are set to 0.
- First, the better goalkeeper out of the 2 is selected and put into the squad.
- Then, the minimum number of players is selected by filtering the array by each of the positions.
- After every pick, the respective flag is raised.
- From the rest of the players, the best ones are selected based on predicted points. The number of selected players in each position must not be higher than the flags set.
- At last, after selecting the 11 players, the other 4 are put on the bench.

## 6.2 Data preparation

Creating the dataset for this thesis consists of two parts. The first part is obtaining player data from the FPL API. This was mainly covered in section 5.3 in the previous chapter. In section 6.3 there will be more information about the scripts downloading the data are functioning and what are their outputs. The second part of dataset preparation is dividing and filtering the downloaded data to form that can be used by a prediction model. This is covered in section 6.4. When these two steps are completed, the dataset is prepared to be used by both the Web application, to read data about the players, and by prediction model to read historical statistics and player attributes.

## 6.3 Downloading and extraction

The downloading of the FPL player data works in these steps:

1. First step is to load the main FPL data endpoint<sup>6</sup>, that contains most of the available information, and store the raw response in a file. The `elements` property of the payload is extracted to a separate file.
2. From the `events` property of the first payload, the current game week is extracted. This will be needed to distinguish between past and future game week data.

---

<sup>6</sup><https://fantasy.premierleague.com/api/bootstrap-static/>

3. In the next step, player data that can be used for prediction are extracted to a separate file. This step filters out the static player data. Also, the fixtures need to be downloaded from a different endpoint, and stored as well. These are needed for mapping past results to data and to create new records for future results prediction.
4. Player names and identification numbers are mapped together and stored in a mapping file. These will be useful to map historical player records to players because, in past data, the players and teams are referenced just by the id.
5. Last step is to download historical player data for every player.

At this time, no data filtering or feature selection is happening. This process creates the same data structure as in the [vaastav<sup>7</sup>](#)'s project, that collects older FPL data, so the old and new data can be easily merged. The dataset contains information about all of the 628 players playing in the Premier League in the current season. The old dataset also contains some information from the [Understat<sup>8</sup>](#) web page, which provides some more information about players, but this will not be used in this thesis. In the next section, I will describe the process of filtering and selecting only the data needed for my prediction algorithm.

## 6.4 Filtering and feature selection

This is the description of the process of reading the prepared dataset and selecting and filtering the data needed for this thesis. After this, the dataset is prepared to be used by the prediction model to learn and predict the next scores of players.

1. The first step is to determine, which players left the Premier League before the current season, and therefore we do not need to predict their scores. Their data is still in the dataset, so they need to be filtered out. This is achieved by getting all the player IDs in the current season data and filter out data from past seasons, that does not belong to the selected player ids.
2. The next step is to determine the remaining fixtures in the current season and create empty records in the dataset for them. The information about the opposition team and fixture difficulty are stored in the records, so the prediction model can consider this information.
3. Another important thing is to determine the predicted minutes played by a player for the next matches. Since the total points obtained by a player depends on minutes played, this feature is very useful. Also, not every player can play 90 minutes a match. If a player is injured and he definitely will not play, the algorithm should predict 0 points for him. So I created the **predicted minutes** feature for future fixtures as an average of minutes played in the last 6 fixtures. On top of that, if a player has a positive injury status in the dataset for the current game week, he will not be recommended at all in the web application.
4. Then, there is a loop over all the dataset records in all the past seasons. For every player that is contracted to a Premier League in the current season, the script loads his historical data. Not all of the historical data are loaded, just those that can help

---

<sup>7</sup><https://github.com/vaastav/Fantasy-Premier-League>

<sup>8</sup><https://understat.com/league/EPL/2019>

the prediction. For example goals, assists, and minutes played are very helpful for the prediction model but the kickoff time and how many FPL users had this player selected could be filtered out. In total, 16 features were selected for every player and every past game week.

5. All of this filtered out player data is stored in a file with the player name. Information about the upcoming fixtures, that was created before, is added to the end of the file, to be ready for the prediction model.
  6. The next step is to divide the data for every player into 3 groups: training set, testing set, and future data. The main issue here is that not every player has the same number of records. Some players are in the league for a long time and some came just before the current season. So the datasets for some players can be very small and predictions for these players are not going to be that precise.

This process creates training and testing sets for every player in the dataset. These sets will then be used by a prediction model to predict future player data. It is worth noting, that after each round is finished, the dataset needs to be recreated, because one more data record should be available for every player, and the prediction model needs to be readjusted for the next round.

## 6.5 Forecaster and statistical models

The player performance forecaster also runs on the application server mentioned in Section 6.1 above. It is part of the WEKA<sup>9</sup> library. For making predictions, I used class `WekaForecaster` from its time series package. The forecasting works like this:

1. The script iterates over the `/dataset/players` folder, which contains historical player data divided into training and testing sets. For every player, it loads `.arff` file with his training set. The sizes of the sets vary over every round, but here is the estimate: all past player data until 8 game weeks before current round = training set, 8 past game weeks = testing set. The size of the training set gets bigger, as the player plays more games in the league. For new players in the league, the training set is non-existent, and predicted points are very random.
  2. The script then removes the first two lines in the `.arff` data part. Those lines are same for every player, and they are there to generalize `.arff` file generation in the dataset creation part. They need to be removed before making predictions. Those lines provide value 0 for every number parameter, and boolean values True, False so the `.arff` file gets properly autogenerated. They are shown in Listing 6.1.

Listing 6.1: These 2 lines are put in every .arff file to allow correct autogeneration.

3. Then the `WekaForecaster` is initialized with all needed parameters. The following values were set:

<sup>9</sup><https://www.cs.waikato.ac.nz/ml/weka/>

- fields to forecast: only the total points predicted are forecasted,
- forecaster: the neural network, which is called **Multilayer Perceptron** in WEKA,
- time stamp: the game week number is used as a time variable, it increases for every round,
- lag variables: values between 1 and 8, with flag to adjust for trends set to true,
- overlay fields: opponent team and expected minutes played are known beforehand, so these were selected.

The final parameters and model attributes were selected after testing multiple variants, and the variant with the best results was picked in the end. Then the forecaster is built with the training data, and primed with the validation data. After these steps, the predictions are created.

4. After the forecaster creates the predictions for next game weeks, those are stored on the filesystem in the form of (player name, predicted points) tuples. Every player has some predicted points for each of the following rounds. The current setting is to predict for 3 rounds to the future. These predictions are stored in `/predictions/{gameweek number}.csv` file, and are regenerated every game week.

Here is a list of features that are used for the prediction in the final implemented model:

- game week index: this is the timestamp variable, used to organize data records in time,
- total points: the attribute that the model is predicting,
- creativity, threat, influence, ict index: these values represent player's strengths and they vary for every match,
- bonus points, player prize, goals, assists, yellow cards, red cards, goals conceded, saves, opponent team: these are statistical data about every past match,
- minutes: how many minutes players played in past matches.

Most of the model development and optimizing was done in WEKA GUI, only then after all errors and deviations were eliminated and the model accuracy was high enough, I implemented it in Java code. The WEKA GUI allows to tinker with all the parameters of the model and to see the results immediately. After every iteration, it shows how well the model is performing and I could see, if the changes in parameters I made improved the model.

Besides the **WekaForecaster** model for prediction, that was tinkered with a lot and tested with various players in WEKA GUI, and with this Java API, there is one more model implemented, that does not use machine learning to predict future points. It uses pure arithmetics and counts predicted points directly from points in past rounds. This was created to further compare gained results in Experiments Chapter 7. The algorithm was inspired by a paper [10] about time series modeling, where it was used to evaluate players while building a dream team. This model works like this:

1. The model works with just a few pieces of data: player prize, the points that the player got across all past game weeks, and his position on the pitch.

2. First, the **total points** gained by a player is calculated, using all of his past data. Simultaneously, the model is calculating the second result, which only takes into consideration his past 6 matches. The second model is used for determining, if the form makes a big impact on predicted and achieved points.
3. Then the Cost-per-point (CPP) index is calculated using simple function:

$$\text{costPerPoint} = \text{totalPoints} > 0 ? \text{Math.round}(\text{prize}/\text{totalPoints}) : 0.0 \quad (6.2)$$

This index is useful to find cheaper players that acquire a lot of points, so a smaller value is better here.

4. Next part is to calculate Points-per-match (PPM) index using following function:

$$\text{pointsPerMatch} = \text{Math.round}(\text{totalPoints}/\text{gameweeks}) \quad (6.3)$$

This value represents average points that a player gets during a match, so a higher value is better.

5. Now the CPP and PPM need to be normalized to range [0-1]. This is because they need to be then added together, by giving equal weight to each and then all of the players are evaluated equally. The weights can be adjusted if the 50/50 approach is not optimal. This sum is called Cost-point index (CPI) and it can identify high performers and cheaper players worth a lot of points. It is calculated using following functions:

$$\text{normalizedCPP} = (\text{CPP} - \text{minCPP}) / (\text{maxCPP} - \text{minCPP}) \quad (6.4)$$

$$\text{normalizedPPM} = (\text{PPM} - \text{minPPM}) / (\text{maxPPM} - \text{minPPM}) \quad (6.5)$$

$$\text{cpi} = \text{Math.round}(\text{normalizedCPP} + \text{normalizedPPM}) \quad (6.6)$$

where minCPP and maxCPP, are minimal and maximal values of CPP across all players, similarly for minPPM and maxPPM. With every player now evaluated using CPI, the optimization algorithm can choose the best options for the squad even using this method. All these processes are repeated for the second version of the algorithm, using just the last 6 records.

## 6.6 Webserver

The second part of the application is the webserver. It is the part visible to the user. The technologies used for building the web application are React<sup>10</sup> Javascript library, which is currently one of the most popular tools available and Bootstrap<sup>11</sup> for building components. For CSS stylesheets, I used Sass<sup>12</sup>. For code quality, I used Prettier<sup>13</sup> and other libraries were used as well, for example, loading, pagination, slider, and more. Yarn<sup>14</sup> was used as

---

<sup>10</sup><https://reactjs.org/>

<sup>11</sup><https://react-bootstrap.github.io/>

<sup>12</sup><https://sass-lang.com/>

<sup>13</sup><https://prettier.io/>

<sup>14</sup><https://yarnpkg.com/>

a package and dependency manager. The project was bootstrapped using Create React App<sup>15</sup>, which is a build tool, that creates all necessary files and dependencies at the start of the project, so the development can start right away. It was tested in Firefox and Chrome web browsers, in the latest<sup>16</sup> versions. The web server allows users to access all developed tools for optimizing his current FPL squad. I will explain all the developed features in the next sections. The application offers Czech and English translations.

## State management

All the general player data that are displayed on the dashboard, are loaded at the mounting of the application. The data are stored in Redux<sup>17</sup> state container. The application uses one single container for every page. These data can be accessed from any component in the application. The player detail data are loaded only after clicking on the player icon or row in any of the tables. Any data that are not needed for the dashboard, are only fetched after visiting the page, they are used in. There is no web browser's local storage used, which means that the data is lost when the window is closed. The data is only reloaded after logging out of the application.

## Authentication

User does not need to authenticate to use this application, it works without logging in as well. But he has this option, and it has its benefits. If he chooses so, he must enter the same credentials that he uses for the FPL App. The credentials are then used to load his squad from the FPL servers. This process was further described in Section 6.1, in the Authentication part. If the user does not want to enter his credentials, he can still open the app, but he has to input his team manually by choosing players from the transfer market. This way is a little inaccurate, because the costs of the players may be different than at the time he bought them originally. But some users don't want to use their credentials on third-party websites, so this is why this option is provided. In Figure, 6.1 there are two possible screens that the user can get, first one is with his loaded team from the FPL, the second one is empty because he wants to input the team manually.

## Changing the team manually

The first option available to the user is the option to change his team manually, by checking the transfer market. The user has options to replace any member of his squad for players in the transfer market, as long as the rules of the FPL are still unbroken. More on the rules can be found in Section 2.3. The transfer market has options to filter players by position and club and to sort them by various parameters. The transfer market is shown in Figure 6.2. There is also a screen, where the user can see how the algorithm predicts his players to play in the next round. He can use this information to navigate the transfer market, and it can help him choose, which player to replace. When the user makes a change, the predictions are reloaded based on his current selected squad. These screens can be found in Figure 6.3. The user also has the option to browse prediction for all available players. All of these options are shown in one single dashboard, which makes them easily accessible and the user can use them all at once.

---

<sup>15</sup><https://create-react-app.dev/>

<sup>16</sup>Firefox: version 76.0.1, Chrome: version 83.0.4103.61

<sup>17</sup><https://redux.js.org/>

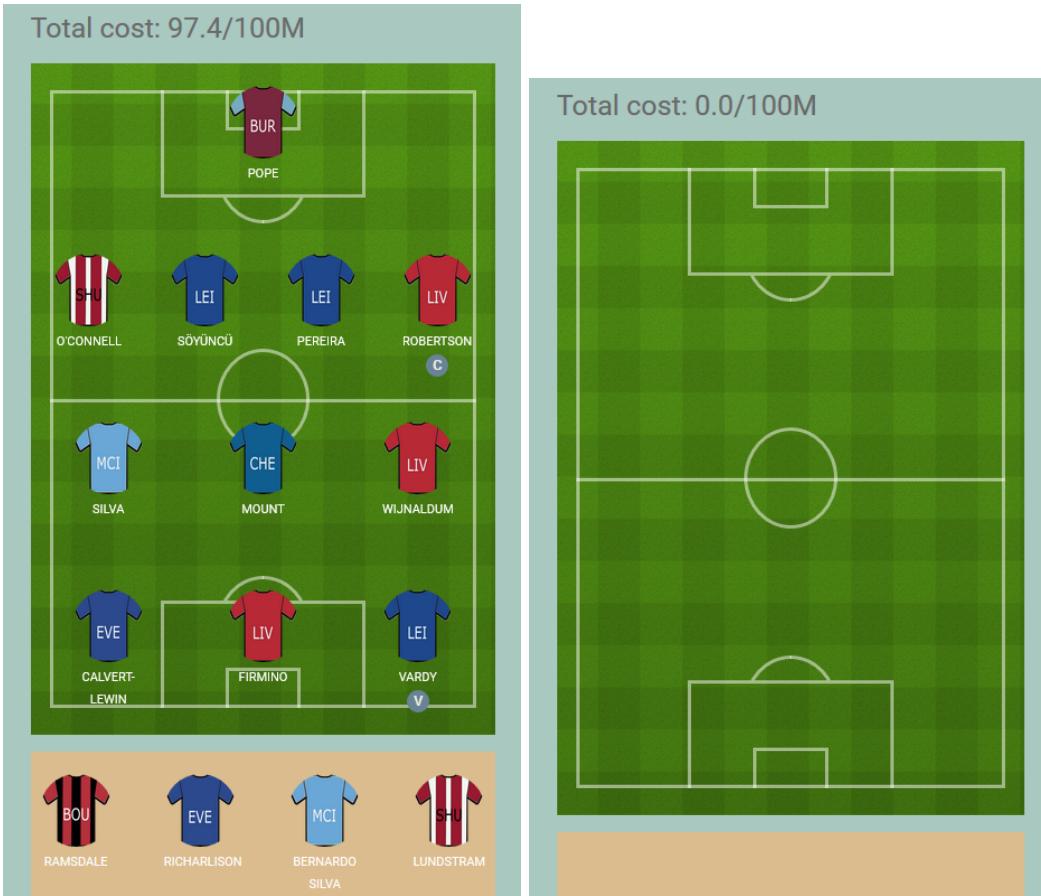


Figure 6.1: Team Overview screen. On the left there is already loaded team after inputting FPL credentials, on the right there is empty squad screen ready to manually input the team by user. This is my current team for the Game Week 29.

Every player has his player detail card, where there are options to remove him from the squad, to substitute him for a player on the bench, to select a player as a captain or a vice-captain. If the user is interested in past statistics of a player, there is also the option to see those on the player detail card. The card can also show his current injury or suspension status. These options are shown in Figure 6.4. There is also an option to reset squad to its original form in the “Team Overview” screen.

### Automatic optimization

This is the main feature of this web application, it is an option to call the created prediction model in the application server and make it automatically optimize the user’s squad. There are multiple optimization options, that are also shown in Figure 6.5. The options provided are explained here:

- Number of transfers: every week, just one transfer is allowed in general. But in some cases, when there is no transfer in the previous week, the transfer chip is moved to the next round. That’s why there are three options: 0, 1, and 2 transfers.

The screenshot shows a user interface for a transfer market. At the top, there are three dropdown menus: 'Position' set to 'Defender', 'Team' set to 'Liverpool', and 'Sort by' set to 'Points'. Below these is a table with six rows, each representing a player: Alexander-Arnold (LIV, 166 points, 7.8 price), van Dijk (LIV, 141 points, 6.6 price), Gomez (LIV, 76 points, 5.3 price), Matip (LIV, 33 points, 5.2 price), Lovren (LIV, 17 points, 5.3 price), and Williams (LIV, 0 points, 4.0 price). The table has columns for Name, Team, Points, and Price.

Name	Team	Points	Price
Alexander-Arnold	LIV	166	7.8
van Dijk	LIV	141	6.6
Gomez	LIV	76	5.3
Matip	LIV	33	5.2
Lovren	LIV	17	5.3
Williams	LIV	0	4.0

Figure 6.2: “Transfer Market” screen. User can browse and select players that he wants to add to his team. The page contains pagination, so user can browse multiple pages, if the filtering criteria are too broad. Every row is clickable, and user can display more information about the particular player.

The screenshot displays two side-by-side tables. The left table, titled 'PROJECTED PERFORMANCE', shows 'Total points: 87' and lists player names with their projected next-round points: Ramsdale (1), Robertson (6), Rüdiger (C) (15), Bernardo Silva (4), Silva (1), Calvert-Lewin (7), and Wijnaldum (4). The right table, titled 'PROJECTED POINT DIFFERENCES', compares 'Original' and 'Current' total points (73 and 87 respectively) and shows changes for specific players: Söyüncü (from 1 to 15), Robertson (C) (from 6 to 8), and Vardy (V) (from 4 to 8).

#	Name	Next round
0	Ramsdale	1
1	Robertson	6
2	Rüdiger (C)	15
3	Bernardo Silva	4
4	Silva	1
5	Calvert-Lewin	7
6	Wijnaldum	4

Original		Current	
Total points:	73	Total points:	87
Name	Next round	Name	Next round
Söyüncü	1	Rüdiger (C)	15
Robertson (C)	6	Pereira (V)	8
Vardy (V)	4		

Figure 6.3: “Projected Performance” screens. On the left there is information about predicted points for every player in the team. On the right there is comparison between original and current team, and how the predicted points have changed. The screens also contain information about captaincy changes.

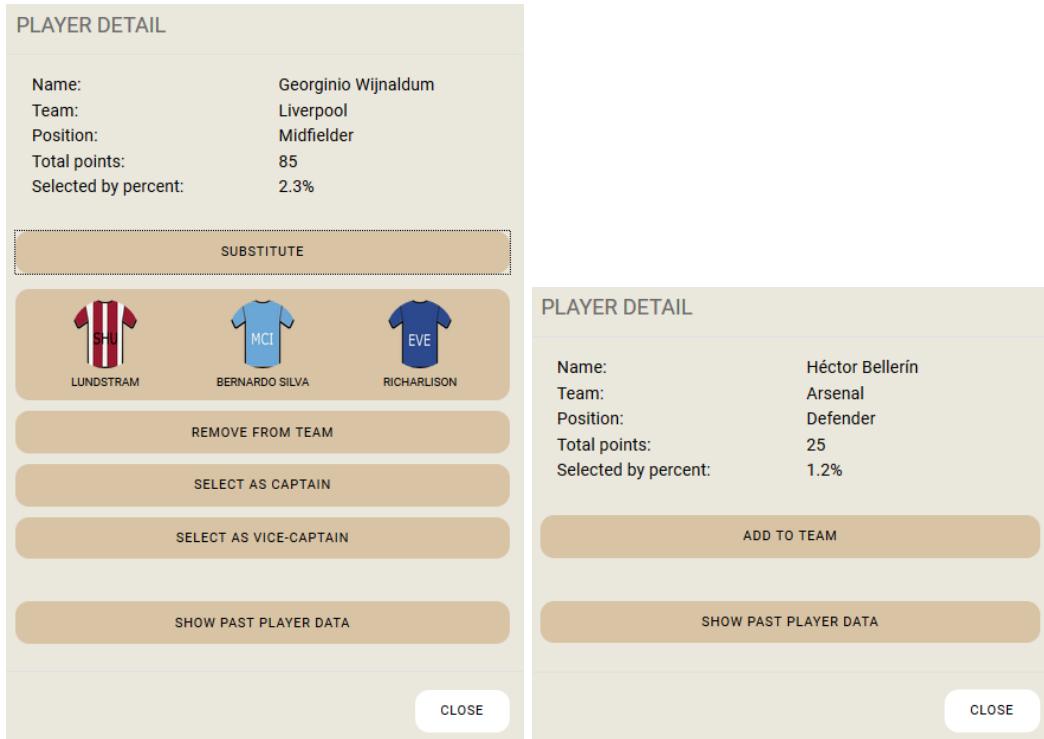


Figure 6.4: “Player Detail” screen. The options provided vary based on if the player is in the team or on the bench, or if he is in the transfer market, is he is captain or not.

- Selection technique: there are 3 options to choose from, and they all call different models on the application server. The user has the option to choose which one he likes the most.
- Gameweeks: the user has the option to predict results up to 3 weeks ahead currently. More game weeks can be added in the future.
- Tips count: this is just an option for users to select, how many different tips he wants to get, so he can choose in the end, what changes he likes the most.

When the prediction model finishes the computations and optimizations, the final results are fetched from the server and displayed to the user. Then he has the option to apply the changes or select another option. Then the changes are applied and he can see his new team on the “Team Overview” screen and adjust it more if he wants. The results are displayed in Figure 6.6.

### Player statistics

The last part of the website are the player statistics. Users can browse through statistics from past seasons with sorting and filtering options. He can also see, which players are currently injured, suspended, or unavailable for selection. This information can help users decide on team changes. They are shown in Figures 6.7 and 6.8.

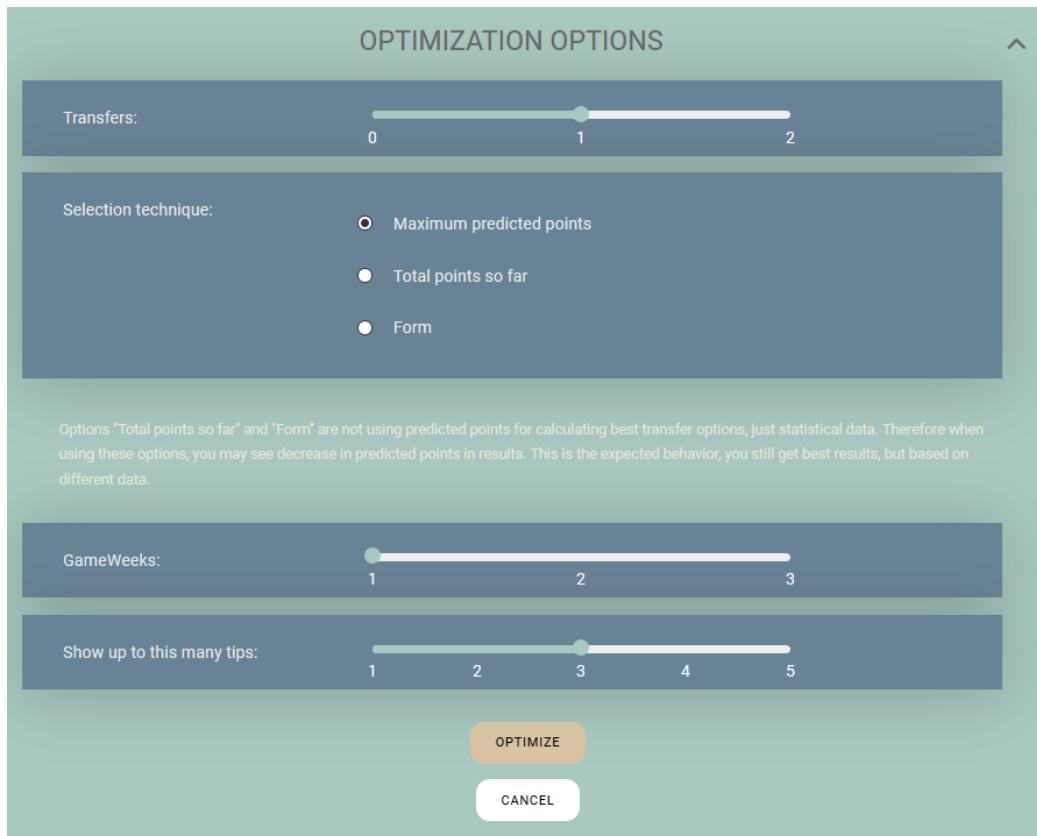


Figure 6.5: “Optimization Options” screen. All the options that can be set on the optimization model can be set here. They are described further above.

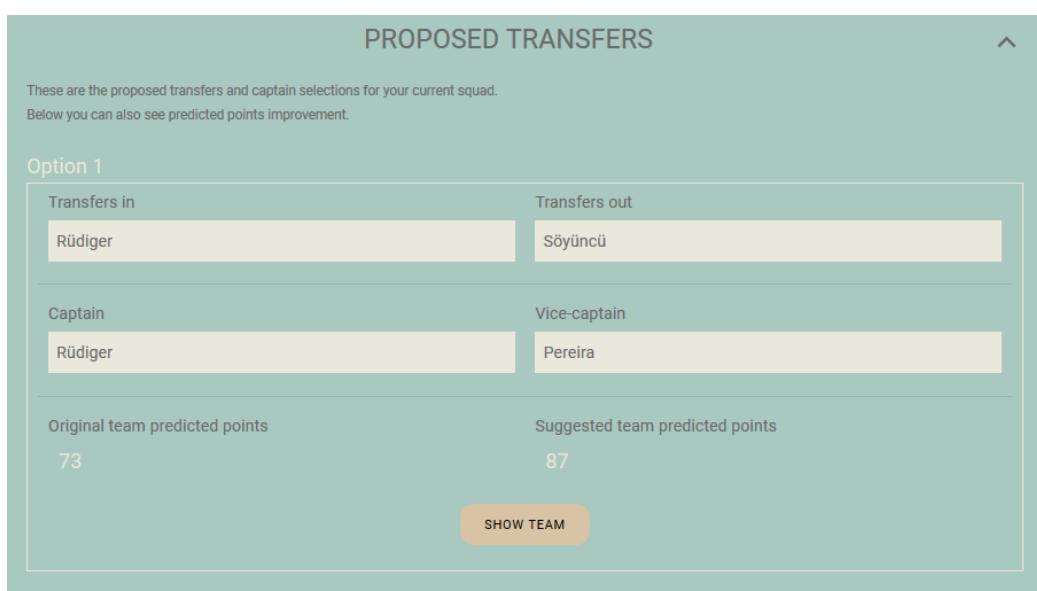


Figure 6.6: “Optimization Results” screen. Here the user is provided with advice how to change players in his team based on selected optimization options.

Position	Team	Sort by		
Forward	Arsenal	Points		
Name	Team	Selected by	Points	Bonus points
Aubameyang	ARS	27.7 %	152	601
Lacazette	ARS	2.1 %	78	277
Martinelli	ARS	2.8 %	39	129
Nketiah	ARS	1.3 %	12	27
John-Jules	ARS	0 %	0	0

Figure 6.7: “Player Statistics” screen. It shows comprehensive information about every player. Every row can be clicked on, and there are more detailed information about each past gameweek.

Team	Sort by		
Chelsea	Name		
Name	Team	Reason	Yellow/Red Cards
Abraham	CHE	Ankle injury - 75% chance of playing	2/0
Jorginho	CHE	Suspended until 12 Apr	10/0
Hudson-Odoi	CHE	Hamstring injury - Unknown return date	1/0
Kanté	CHE	Thigh injury - 75% chance of playing	3/0
Kovacic	CHE	Achilles injury - 75% chance of playing	7/0
Mount	CHE	Knock - 75% chance of playing	3/0
Pulisic	CHE	Lack of match fitness - 75% chance of playing	0/0

Figure 6.8: “Unavailable Players” screen. Here user can see which players are unavailable for selection for next gameweek and are sure to get zero points, so he can remove them from his squad.

# Chapter 7

## Experiments and evaluation

This chapter will discuss all the experiments, that were made during the implementation. There were experiments with the feature selection and tuning the prediction model. These experiments were needed to choose the best combination of features and model parameters for player performance prediction. After the model was implemented, a small group of random players was selected and their predicted points were compared to the real ones, to test the prediction model accuracy. Then, there were experiments with the web application, to determine if all the tools provided to the user are working as expected. Last but not least, there were tests conducted to compare the model prediction over several game rounds with the points obtained in the FPL App by a regular and a top-ranked user. These experiments should provide information about whether using the provided model can help with the overall performance in the game.

### 7.1 Model attributes and feature selection testing

Experiments with the model attributes and selected features were conducted in the early parts of the model implementation. Using the WEKA GUI with a small number of randomly picked players, the best combination of attributes was selected. The metrics used to compare different combinations of attributes were Mean Absolute Error(MAE) and Root Mean Square Error(RMSE). WEKA GUI offers more options for evaluation, but these two were selected as the most accurate. The goal was to find the combination of parameters with the lowest RMSE and MAE for the testing set of players.

#### Mean Absolute Error

Mean Absolute Error is a model evaluation metric, used mostly with regression models. It represents the mean of the absolute values of the individual prediction errors on all the instances in the testing set. The individual prediction errors are calculated as the difference between the true value and the predicted value for each instance. It is represented as the following formula:

$$MAE = \frac{\sum_{i=1}^n abs(y_i - \lambda(x_i))}{n} \quad (7.1)$$

where  $n$  is the number of instances,  $y_i$  is the true value and  $\lambda(x_i)$  is the predicted value.

## Root Mean Square Error

Similarly to the MAE, it is used to measure the difference between true and predicted values. It represents the square root of the average of squared individual errors. It is always non-negative and a value of 0 would represent the perfect fit. Root Mean Square Error is represented as:

$$\sqrt{\frac{\sum_{i=1}^n (y_i - \lambda(x_i))^2}{n}} \quad (7.2)$$

where  $n$  is the number of instances in the testing set,  $y_i$  is the true value, and  $\lambda(x_i)$  is the predicted value.

The selected attributes and features are further explained in Section 6.5. In Figure 7.1, there is an example of the output of the WEKA GUI, that was used to compare different combinations of attributes. For every calculation, the graphical representation of the errors was displayed, as shown in Figure 7.2. Players had a different number of training and testing instances available, due to some of them have been playing in the league just for a couple of matches, which was a problem. This meant that the MAE and RMSE errors were very different for various groups of players.

```
==== Predictions for test data: total_points (1-step ahead) ====
==== Evaluation on training data ====
Target          1-step-ahead    inst#   actual   predicted   error
=====
total_points
N               95           104      2        2.3612    0.3612
Mean absolute error 0.6532     105      3        3.291     0.291
Root mean squared error 1.0213    106      6        2.8097   -3.1903
                                         107      0        0.4352    0.4352
                                         108      0        -0.5224   -0.5224
                                         109      0        -0.1128   -0.1128
                                         110      2        2.9339    0.9339
                                         111      2        3.936     1.936
                                         112      5        4.4865   -0.5135
                                         113      2        4.0243    2.0243
                                         114      2        2.8313    0.8313
Total number of instances: 103
=====

==== Evaluation on test data ====
Target          1-step-ahead    inst#   actual   predicted   error
=====
total_points
N               37           115      2        2.5504    0.5504
Mean absolute error 1.8134     116      3        2.7341   -0.2659
Root mean squared error 2.3304    117      2        2.1863    0.1863
                                         118      2        3.143     1.143
                                         119      2        3.1887   1.1887
                                         120      1        3.2863   2.2863
                                         121      1        3.371     2.371
                                         122      2        3.8892   1.8892
                                         123      2        4.2311   2.2311
                                         124      1        1.7862   0.7862
Total number of instances: 44
                                         125      7        4.0962   -2.9038
                                         126      2        4.0651   2.0651
```

Figure 7.1: Example output of the WEKA GUI with the MAE and RMSE values.

## 7.2 Graphical user interface testing

This part of the testing was done during the whole implementation of the web application. The new pages and tools were tested during the development and then once in a while, the whole application was tested again. These were the main parts that were tested the most:

- Proper loading of the player data and predictions: if the data structure changed, the changes needed to be made in the web application as well.

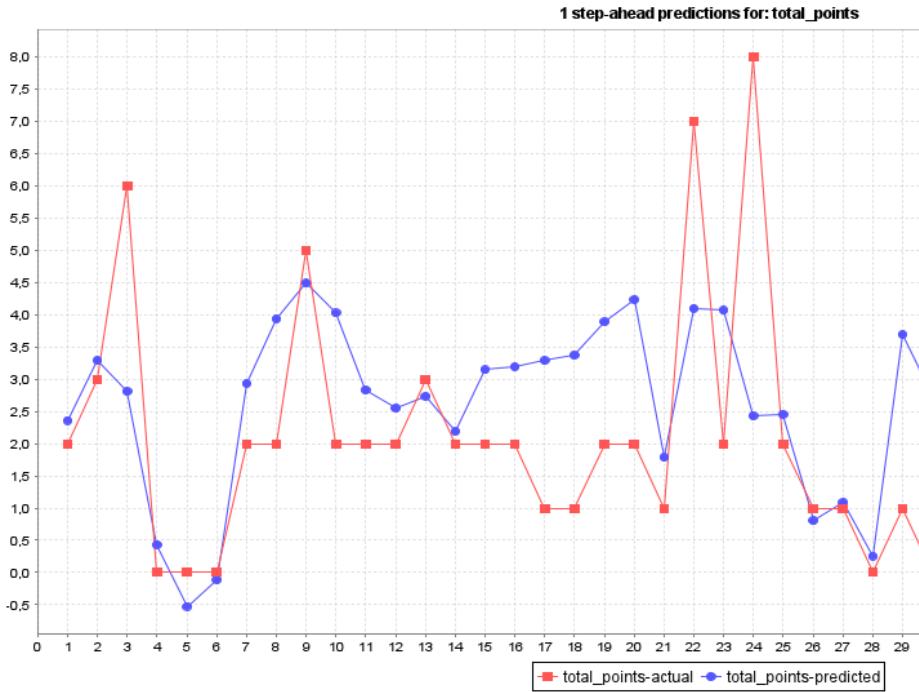


Figure 7.2: Example graphical output of the WEKA GUI for the difference between true and predicted values for a selected testing set.

- Sorting and filtering based on different parameters: as the data is not uniform and has a different structure in every table, the sorting, and filtering needed to be adjusted in each of them.
- Operation with the team: adding, removing, substituting players.
- Different combinations of selected players.

The biggest issues encountered during the web application testing were the edge cases within the data. For example, if the user had no data available to show in the tables if he had already the best possible team and there was nothing to optimize. All of the use cases tested are further explained in Section 6.6.

### 7.3 Prediction accuracy testing

This section focuses on showing the results provided by the prediction model. Since the dataset generates a lot of time-series data, the process of predicting results and comparing them to the true values needed to be automated. I chose a few random players and simulated a few Game weeks with them, and compared their predicted points with the true ones, to showcase prediction accuracy. Moreover, I created a service that gets the current user's team in the Game week 10 and simulates the season for 10 weeks and then provides the predicted and real values for points obtained for every player. This process is very time and resource-demanding. Generating the predictions and simulating the squad optimizing just for the 10 weeks takes up to 7 hours of processing time. That means, that not a lot of experiments of this size could be done, and that's why I abandoned the idea to simulate

the whole season. In the next two sections, there will be more information about these experiments.

### Predicted values comparison experiment

In this section, the process of comparing the predicted values with real data for 24 selected players is described. This comparison is needed to see, if the predicted values get close to the real ones, and how close the model is for different groups of players. I selected 3 groups of players, in every group, there are 2 players for each of the 4 positions on the pitch.

- Group 1: Players with generally higher scores that are playing in the league for more than 1 season. Here the selected players were: Goalkeepers: K.Schmeichel, Alisson; Defenders: V.van Dijk, C.Soyuncu; Midfielders: K.de Bruyne, Richarlison; Forwards: J.Vardy, P.E.Aubameyang.
- Group 2: Players with generally lower scores that are playing in the league for more than 1 season. The selected players were: Goalkeepers: H.Lloris, Kepa; Defenders: K.Zouma, S.Mustafi; Midfielders: J.Lingard, Pedro; Forwards: S.Long, T.Deeney.
- Group 3: Players that are new to the league. Goalkeepers: D.Henderson, A.Ramsdale; Defenders: M.Aarons, J.Cancelo; Midfielders: M.Mount, N.Pepe; Forwards: Wesley, Joelinton.

I ran the simulations from 10th until the 28th round, as this was the last round with all the completed fixtures before the Premier League was suspended. I started deliberately in the 10th round, because I wanted the players in Group 3 to have at least some past data, for the training set.

In Table 7.1, there is a comparison between average MAE and RMSE over all 19 rounds in this experiment for each of the 3 groups. Unsurprisingly, the model shows better results with players that get lower points consistently (Group 2). These players don't very often get points bigger than 8/9 in a round, more often they get points closer to zero. That's why the model has a lower error rate with these players. For Group 1, the error rate is higher. The reason for that is, that these players often have game rounds, where they get much higher points than the round before, and the round after that they get average points again. The model, predicts those higher valued weeks well, but the difference between obtained points is higher too, so the error rate increases. In Figure 7.3 there is an example graph of 3 players, one from each group, with differences between the true and predicted values for multiple rounds.

Table 7.1: MAE and RMSE values for all 3 groups of players in selection. The values are averaged over the 19 rounds in experiment.

Group	average MAE	average RMSE
1	3.17	3.45
2	2.58	2.80
3	2.72	2.77

There was no evidence, that the models perform much worse with players in Group 3, which has fewer available training data. This can mean, that the player form from the past does not have such a great impact on his score. There was also no evidence, that the

model performs predictions with different accuracy for the 4 player positions, the error rate is very similar to all 4 of them. The data shows, that if the player has a great game, the model does not automatically start to predict much higher scores for him, but adjusts the predictions more sensibly. This can be seen in Table 7.2.

Table 7.2: The average values of MAE and RMSE for all 4 different positions. It shows only small differences between them.

Position	average MAE	average RMSE
Goalkeepers	2.80	2.99
Defenders	2.74	2.93
Midfielders	2.82	3.03
Forwards	3.32	3.07

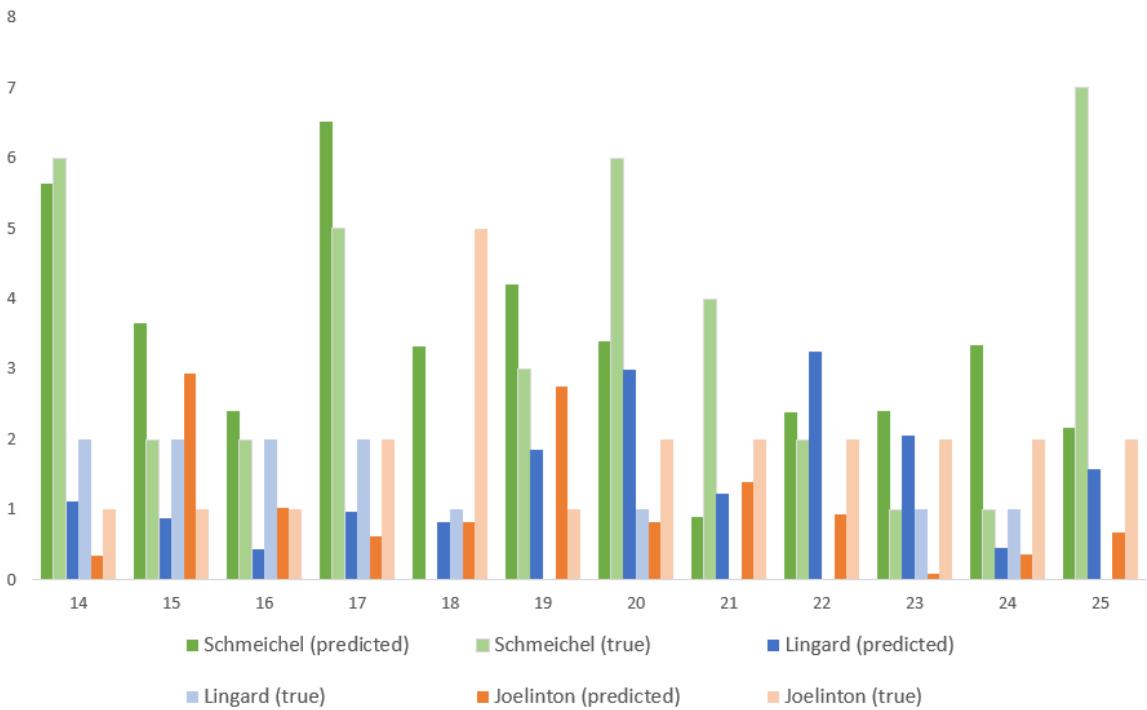


Figure 7.3: Comparison in predicted and true point values for 3 selected players from each group. Generally, the predicted points are lower than the true ones. The model shows a higher error rate with players that have a great match once in a while, with much higher scored points than the matches before.

## Optimized team improvement experiment

This experiment was designed to see, if using the implemented predictor for 10 weeks and strictly adhere to its predictions and tips would improve the user's total obtained points in the FPL game. The experiment starts in week 9 because I wanted the players that are new in the league, to have at least some training data. I used my team, and then the team of the current highest-ranked player in the FPL, to compare the impact of the predictor on mediocre and superb teams. For Game week 9, I used the squad, that both of

our teams had in Game week 9 in reality. Then I simulated creating predictions, retrained model, and optimizing the squads for 10 weeks. I used 3 selection techniques for each round to see, which one would be the most effective. Also, I used 3 different configurations of using transfers. In the first one, I used 1 transfer token every week. The second and third combinations used 2 tokens one week and 0 the next one, with alternating weeks in each combination. That meant that for every squad, I got 9 different new teams after every round. The results of the experiment can be seen in Figure 7.4 for my team, which is mediocre and gets average points every week, and in Figure 7.5, for the current number 1 team in the leaderboard, which gets the highest scores almost every week.

For my squad, it can be seen, that the optimized squad does not get higher points than the real one every week for any of the combinations. But for every week, at least one combination gets higher or equal points. That leads to a conclusion, that no single method improves the results every week. This means, that for a mediocre squad like mine, the best solution for optimizing the squad is to get predictions for multiple parameter combinations, and then select the best ones for me. In the next section, I will discuss possible improvements to the prediction models, that can improve one of the selection techniques, and allow better results.

However, for the highest-ranked squad, almost none of the combinations provided better results. This means that the squad is already highly optimized and forcing transfers on it only decreased the total obtained points.

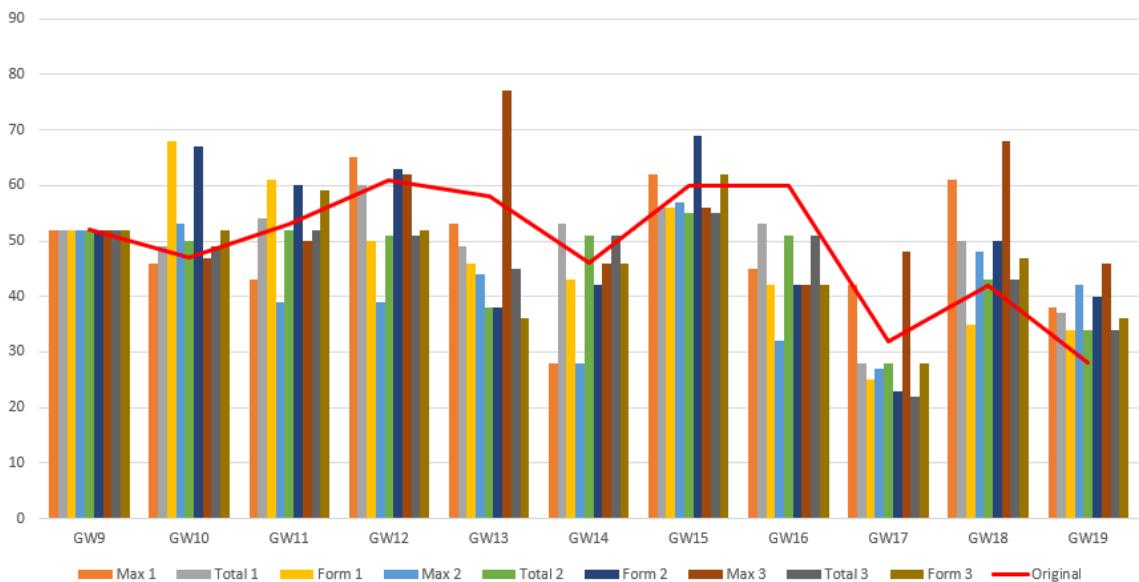


Figure 7.4: Difference in real obtained points for my own squad and the points, that the squads optimized with my algorithms got. Labels “Max”, “Total” and “Form” mean all different selection techniques, numbers mean the combination of transfers.

As a part of this experiment, there was one more calculation created. If for every round in the experiment, I selected the best team, that the application predicted for me, the total improvement would be significant. The predicted ranking and percentil of players, that are better than me is shown in Table 7.3. It is worth noting, that I started playing in round 7 this season, so my ranking is very low due to scoring 0 points in first 6 rounds. My average

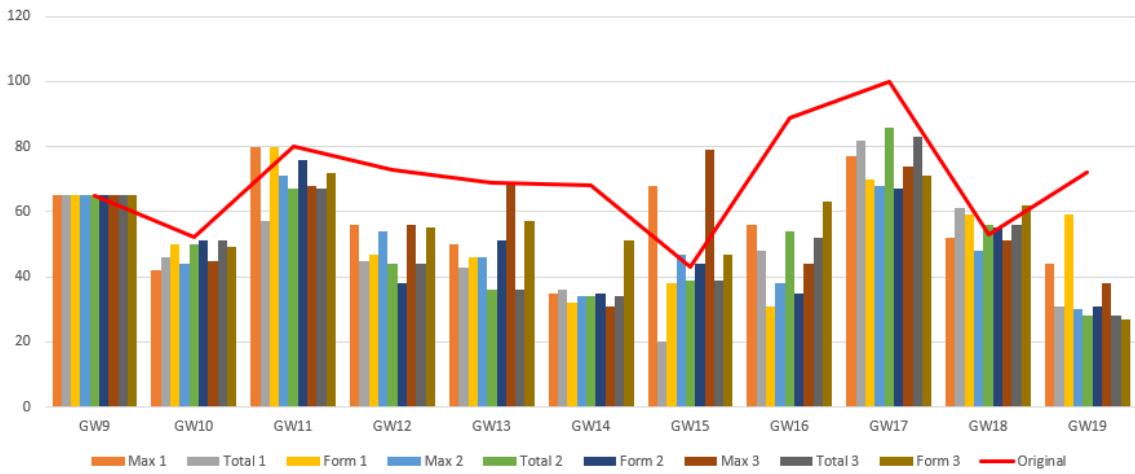


Figure 7.5: Difference in real obtained points for the highest ranked squad and the points, that the squads optimized with my algorithms got. Labels “Max”, “Total” and “Form” mean all different selection techniques, numbers mean the combination of transfers.

position for rounds 7-29 is 3 556 474, so the overall position would be improved more, if I started playing at the start of the season.

Table 7.3: The difference in the ranking between reality and best provided prediction by the algorithm. The total number of players is 7 476 043.

	Ranking	Percentil
Reality	6 539 127	87.47
Best provided prediction	5 958 199	79.70

## Lessons learned

These are the lessons learned from the experiments performed in the previous sections:

- Model predicts the points for players fairly accurately, there are no big deviations for any position or group of players.
- Regarding optimizing the squad, the model works for squads, that are not getting a lot of points, and they can be improved to average. With an improved prediction model, these squads can be improved even more.
- For users in the higher ranks in the FPL leaderboard, this optimizing algorithm does not provide enough improvement.

## Possible improvements

There are some things that I was unable to implement in the prediction model, but they can have a bigger impact on the predicted scores.

First is the **fixture difficulty**. Currently, the model knows, who is the opposition team for every fixture, but if it knew about how powerful the opponent was as well, it could improve the performance. However, this data is not available in the dataset and needs to

be added from the different data sources. Also, the strength of the teams can change every season.

Another useful option to add is to perform **as many transfers as needed**. When I looked at the top players in the leaderboard, some of them are using 6-10 transfers every week. Those additional transfers are taking points from them, but the additional point value from the newly picked players seems to be higher. This option could improve the results of the optimization more.

More experiments like the one in Section 7.3 must be performed to further analyze the impact of the model on different groups of players. The experiment could be expanded to cover players from different clubs, injury-prone, and healthy players, and more.

## Chapter 8

# Conclusion and future work

In this master's thesis, an analysis and a design were made for a web application that predicts the performance of players in the online game Fantasy Premier League and recommends to users how to choose the best possible team. The web application uses prediction model, that was also created, and it can be parametrized with multiple user-selected values. In the first part of the thesis, the Fantasy Premier League game itself was described, how it works, what are the rules and restrictions, and why it is so popular.

In the next chapter existing applications that offer predictions of player performance were discussed. Their advantages and disadvantages were examined, and the most popular features were selected to provide to users. Also, an analysis was made of which services are missing from the applications and would be useful to the user.

There is also introduction to the neural networks and models necessary to create the best and most accurate prediction model. Without this knowledge, it would not be possible to create an accurate prediction model.

In the next part, the architecture of the system and all its parts were designed. The following chapter discusses the dataset that was used to implement the model and analyzes and determines which values and attributes from the dataset are suitable for the prediction model. Various existing libraries have been tested during the analysis and design process to help with machine learning. The WEKA library was selected as the most useful of them.

Next the implementation of the project was described in detail. All the necessary parts were implemented in an effective way. There are pieces of important code and screenshots of the implemented application to allow reader to imagine, how it looks in reality.

The last chapter discusses all the experiments that were conducted to examine the accuracy and usability of the implemented application. The experiments suggest that the application provides moderately good optimization algorithm that works well with average squads in the game. For more superb squads, the results were worse and could be further improved.

Altogether, the application was implemented and tested with all proposed features and provides usable tools for Fantasy Premier League players to get advice on how to improve their squad. For future work, the prediction model needs to be improved by trying to add new features and test their usability and accuracy for the model. The application also needs to be properly tested over the course of the whole Premier League season, from first to last round, which was not possible yet.

# Bibliography

- [1] Association, F. S. . G.: Industry Demographics. 2017. [Online; visited 03.12.2019].  
Retrieved from: <https://thefsga.org/industry-demographics/>
- [2] Box, G. E. P.; Jenkins, G. M.: Time Series Analysis Forecasting and Control. 1970.  
Retrieved from:  
<http://garfield.library.upenn.edu/classics1989/A1989AV48500001.pdf>
- [3] Cho, K.; van Merriënboer, B.; Gulcehre, C.; et al.: Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. 2014. [1406.1078](#).
- [4] Cilimkovic, M.: Neural Networks and Back Propagation Algorithm.  
Retrieved from: <http://www.dataminingmasters.com/uploads/studentProjects/NeuralNetworks.pdf>
- [5] Economics, D.: What is the best programming language for Machine Learning? Dec 2019. [Online; visited 06.01.2020].  
Retrieved from: <https://towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7>
- [6] Gastmans, J.: Fantasy sports is a game of skill, according to MIT study. Jan 2019.  
[Online; visited 04.12.2019].  
Retrieved from: <https://fanarena.com/fantasy-sports-game-of-skill/>
- [7] Goff, B.: The \$70 Billion Fantasy Football Market. Aug 2013. [Online; visited 03.12.2019].  
Retrieved from: <https://www.forbes.com/sites/briangoff/2013/08/20/the-70-billion-fantasy-football-market/#30a3709d755c>
- [8] Green, C.: 'Wink': Wilfred 'Bill' Winkenbach invented Fantasy Football way back in 1962 with GOPPPL in Oakland. [Online; visited 05.01.2020].  
Retrieved from: <https://web.archive.org/web/20150929163914/http://www.newsnet5.com/sports/wink-wilfred-bill-winkenbach-invented-fantasy-football-way-back-in-1962-with-gopppl-in-oakland>
- [9] Greg Greenhalgh, B. B., Brendan Dwyer: There's an App for That. 2014. [Online; visited 05.01.2020].  
Retrieved from: <https://js.sagamorepub.com/jasm/article/download/6093/4851>
- [10] Gupta, A.: Time Series Modeling for Dream Team in Fantasy Premier League. 2019. [1909.12938](#).

- [11] Gurney, K.: *An Introduction to Neural Networks*. USA: Taylor & Francis, Inc.. 1997. ISBN 1857286731.
- [12] Jha, S. K.; Tiwari, A.: Necessity of Goodness of Fit Tests in Research and Development. 2011.  
Retrieved from: <http://www.ijcst.com/vol23/1/sanjeevjha.pdf>
- [13] Liu, Q.; Wu, Y.: Supervised Learning. 2012. doi:10.1007/978-1-4419-1428-6\_451.
- [14] Mitchell, T. M.: *Machine Learning*. McGraw-Hill Education. 1997. ISBN 0070428077.
- [15] Nasteski, V.: An overview of the supervised machine learning methods. *HORIZONS.B*. vol. 4. 12 2017: pp. 51–62. doi:10.20544/HORIZONS.B.04.1.17.P05.  
Retrieved from: [https://www.researchgate.net/publication/328146111\\_An\\_overview\\_of\\_the\\_supervised\\_machine\\_learning\\_methods](https://www.researchgate.net/publication/328146111_An_overview_of_the_supervised_machine_learning_methods)
- [16] Nyquist, R.; Pettersson, D.: Football match prediction using deep learning. 2017.  
Retrieved from: <https://pdfs.semanticscholar.org/e556/af01e86c3414042aa69831ea5fb398e66f94.pdf>
- [17] Porter, J. W.: *Predictive Analytics for Fantasy Football: Predicting Player Performance Across the NFL*. Honors Theses and Capstones. 406. 2018. [Online; visited 04.12.2019].  
Retrieved from: <https://scholars.unh.edu/honors/406>
- [18] Production, L.: Creating an Immersive Experience for Connected Sports Fans. Jan 2016. [Online; visited 04.12.2019].  
Retrieved from: <https://www.live-production.tv/case-studies/sports/creating-immersive-experience-connected-sports-fans.html>
- [19] Release, Y. I. P.: Yahoo! Sports Hits Home Run With Free Fantasy Baseball. 1999. [Online; visited 05.01.2020].  
Retrieved from: <https://web.archive.org/web/20071031081947/http://yaho.client.shareholder.com/press/ReleaseDetail.cfm?ReleaseID=173646>
- [20] Sawe, B. E.: The Most Popular Sports in the World. Sep 2016. [Online; visited 04.12.2019].  
Retrieved from: <https://www.worldatlas.com/articles/what-are-the-most-popular-sports-in-the-world.html>
- [21] Schoenke, P.: It was 20 years ago today. Feb 2017. [Online; visited 05.01.2020].  
Retrieved from: <https://www.rotowire.com/blog/post.php?id=14758>
- [22] Schwarz, A.: *The Numbers Game: Baseball's Lifelong Fascination with Statistics*. Thomas Dunne Books. jul 2004. ISBN 0312322224.  
Retrieved from: <https://www.xarg.org/ref/a/0312322224/>
- [23] Shalabh: Polynomial Regression Models.  
Retrieved from: <http://home.iitk.ac.in/~shalab/regression/Chapter12-Regression-PolynomialRegression.pdf>

- [24] Shalabh: Simple Linear Regression Analysis.  
Retrieved from: <http://home.iitk.ac.in/~shalab/regression/Chapter2-Regression-SimpleLinearRegressionAnalysis.pdf>
- [25] Sridharan, R.: Linear regression. 2015.  
Retrieved from: <http://www.mit.edu/~6.s085/notes/lecture3.pdf>
- [26] T. Matthews, G. C., S. D. Ramchurn: Competing with Humans at Fantasy Football: Team Formation in Large Partially-Observable Domains. 2012. [Online; visited 05.01.2020].  
Retrieved from:  
<http://www.intelligence.tuc.gr/~gehalk/Papers/fantasyFootball2012cr.pdf>
- [27] Zaremba, W.; Sutskever, I.; Vinyals, O.: Recurrent Neural Network Regularization. 2014. [1409.2329](https://arxiv.org/abs/1409.2329).
- [28] Zhang, G.; Eddy Patuwo, B.; Y. Hu, M.: Forecasting with artificial neural networks:: The state of the art. *International Journal of Forecasting.* vol. 14, no. 1. 1998: pp. 35–62.  
Retrieved from: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.138.4828&rep=rep1&type=pdf>