An Evaluation of Machine Learning Techniques and Social Media Analysis to Predict an Optimal Fantasy Football Team

CS344 Final Report

Jack Benyon

2208738

Supervisor

Dr. Ian Saunders

University of Warwick

April 2025

Abstract

The use of AI and machine learning has become increasingly popular as a tool to aid with fantasy sport games. In fantasy sport games, players earn points based on their real life performances in matches. This project focuses on Fantasy Premier League, a fantasy sport game based on the Premier League, which is the highest tier football league in England. In this project we aim to use machine learning to try and predict, as accurately as possible, how many points a player will score in a Premier League football match. Using data collected from the 2021/22, 2022/23 and 2024/25 Premier League seasons, we explore the use of different types of models and datasets to predict player points, with the least error possible. The player points predictions were then used to calculate the predicted highest scoring team of players for a given set of upcoming matches, and to recommend player transfers that should be made to an existing team of players in order to maximise the predicted score of the team. A web app was created which included features to allow a user to generate the highest predicted scoring team for a set of upcoming matches, and to allow a user to input their current team and generate personalised transfer recommendations.

To try and strengthen the performance of the machine learning models, posts related to Premier League players and Fantasy Premier League, on the social media platform X, were analysed. The content of the posts was analysed to try and detect information on players that was not available to the models, such as a player's recent injury status. Several analysis methods were explored, and it was found that including the analysis as features in the machine learning models' training and test datasets led to a reduction in error in player points predictions.

Keywords

Machine Learning, Football, Natural Language Processing, X/Twitter, Web Application, Social Media

Acknowledgments

Firstly, I would like to express my sincerest thanks to my project supervisor, Dr Ian Saunders. The feedback and suggestions that were provided during our supervisor meetings have been crucial to the overall success of the project. Furthermore, the feedback provided for the drafts of my progress report and final report was invaluable and ensured that I did not miss any important details in the final documents.

Secondly, I would like to thank my friends for their continuous support and motivation during the entire course of the project.

Finally, I would like to thank my family for their perpetual support and advice during the entire course of the project.

Contents

1 Introduction

1.1 Fantasy Sports

Fantasy sport games are enjoyed by sports fans all over the world. The main objective in a fantasy sport game is to choose a team of virtual players, usually on a weekly basis, which earn points based on how the players in the chosen team perform in real life. The early history of fantasy sports can be traced to the mid-19th century with the tabletop game Parlor Base-Ball [1]. Before the introduction of the World Wide Web, fantasy games were mainly played through newspapers and on the telephone. In 2022, in the USA and Canada alone, it was estimated that 62.5 million people were playing fantasy sports [2]. Football, Basketball, and Baseball were the most popular fantasy sports which these people played. Fantasy sport games allow friends and family to compete against each other, and there are often prizes awarded to the highest scoring players at the end of game.

Fantasy Premier League (FPL) [3] is the fantasy game that we focus on for this project. FPL is based on the Premier League; the highest level of professional football in England.

1.2 Fantasy Premier League (FPL)

The Premier League (PL) consists of 20 teams which play each other twice during the season (once at each team's home stadium). A season consists of 38 "gameweeks" in which there will usually be 10 fixtures where each team plays one match against another team (sometimes there may be more or less than 10 fixtures due to fixture rearrangements).
From this point onwards we will refer to people that play FPL as "managers" and the footballers that that can be selected in an FPL team as "players". Before the first game of each gameweek, FPL managers must submit their team of players. This team will then earn points based on how each player performs in their respective fixture. Note that a manager's FPL team cannot be changed while there are still games to be played in the current gameweek.

FPL begins prior to gameweek one of the PL season. Managers must choose an initial team of 15 players from a pool of over 600 available players. The team must not exceed an initial budget of £100 million and there can be at most three players from each PL team within the chosen team. Also, every team must consist of two goalkeepers, five defenders, five midfielders and three forwards. For the 2024/25 season each player had a starting price of between £4.0 and £15.0 million. Prices are determined by how well a player has performed in previous seasons, with the most desirable players to own typically being the highest priced.

Figure 1 shows part of the team selection page on the FPL website. Each square on the pitch represents a player and stores the following information :

- An image of the player's team kit
- The player's name
- The team that the player is playing against in the current gameweek and whether they are playing at home (H) or away (A)
- If the player has been captained or vice-captained
- The current injury status of the player

For each gameweek you must select 11 starting players from your team of 15 players, and the remaining four form the substitute bench. Only players in the starting 11 gain points for your team, unless a player does not play at all in their match, in which case they are swapped with a player on the bench, and the swapped player's points are counted instead. The order of players on the bench determines the priority of which they are substituted, if necessary. In addition, a captain and vice-captain must be selected every gameweek. The captain scores double points for the gameweek and the vice-captain's points are doubled only if the captain does not play at all in their match.

The injury status of the player is represented by the coloured triangle (lack of triangle means the player is likely not injured) in the top right corner of their square. These are known as "flags". There are different severity of injuries, represented by the colour of the flag, which are detailed as follows :

- Yellow - The player's status is uncertain
- Amber - The player is unlikely to play
- Red - The player will almost certainly not play

Figure 1: An example of part of the FPL Team Selection Page

There are a number of ways in which a player can score points in a match [4]. These are outlined below in Table 1. The overall score that a manager gets for their team at the end of a gameweek is the sum of the points of all the players in starting 11 of their team and any the points of any players which have been subbed on from the bench.

Action	Points
For playing up to 60 minutes	1
For playing 60 minutes or more	2
For each goal scored by a goalkeeper	10
For each goal scored by a 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 kept by a goalkeeper or defender	4
For a clean sheet kept by a midfielder	1
For every 3 shots saved by a goalkeeper	1
For each penalty save by a goalkeeper	5
For each penalty miss	-2
Bonus points for the best players in a match	1-3
For every 2 goals conceded by a goalkeeper or defender	-1
For each yellow card received	-1
For each red card received	-3
For each own goal scored	-2

Table 1: The possible ways that a player can score FPL points in a match

In FPL, an assist can be earnt by directly passing the ball to the player who scores a goal, or by being fouled and earning a penalty which is then scored by a different player. Only one player can be credited with an assist for a goal, and it is not possible for a player to score and assist the same goal. A clean sheet is the term for when a player's team concedes zero goals in a match. For a player to achieve a clean sheet, they must play at least 60 minutes in the game and their team cannot concede a goal while they are on the pitch. Note that only goalkeepers, defenders and midfielders can receive points for keeping a clean sheet, and not forwards. Bonus points are earnt at the end of a match, and the top three best performing three players earn an extra three, two and one point(s) respectively. More details on how bonus points are awarded can be found online [5].

Before the start of each gameweek, managers can make transfers to their team, which involves swapping an existing player in their team to a player which they do not currently own. Every manager gains a "free transfer" at the start of each new gameweek, which is a transfer that can be used and does not incur any points deductions to their gameweek score. If the free transfer is not used then it is carried over to the next gameweek, and a manager can have a maximum of five free transfers at any point. For every transfer that is made that exceeds the number of free transfer a manager currently has, four points are deducted from the manager's score at the end of the gameweek.

Player prices are not constant throughout the season and are affected by the number of managers that transfer the player in or out of their team each gameweek. If a player is transferred in by a lot of managers, then their price is likely to increase by £0.1 million and if they are transferred out by a lot of managers then their price is likely to decrease by £0.1 million. The exact algorithm which determines if a player will rise or fall in price is not publicly available. A manager's overall budget changes throughout the season, based on the price changes of players that are in their team. For every £0.2 million that a player rises in price while they are in a manager's team, £0.1 million is added to the player's selling price. For every £0.1 million that a player falls in price while they are in a manager's team, £0.1 million is deducted from the manager's total team budget.

Finally, there are four special "chips" that can each be played once during a season, apart from the wildcard chip which can be played once before and after the halfway point of the season.

1. **Wildcard**
   Allows a manager to make unlimited free transfers to their team for one gameweek only.
2. **Free Hit**
   Allows a manager to make unlimited free transfers to their team for one gameweek only but then the manager's team from the previous week is restored for the next gameweek.
3. **Triple Captain**
   The selected captain of the team earns triple points instead of double points.
4. **Bench Boost**
   All 15 players in the team earn points, instead of the usual starting 11.

1.3 Motivation

The tricky part of FPL is choosing the best team of 15 players from the pool of over 600 available. With the abundance of statistics and data available online for all players and teams, it can often be overwhelming trying to decide which players are the best choice to include in your team.

The aim of this project is to try and reduce the error of machine learning models which aim to predict how many fantasy points each player will score in a given gameweek of the PL. By achieving this, we can then extend to choosing the most optimal team of 15 players.

Although it is useful to predict the best team for a given gameweek, as outlined above, manager's have a fixed number of free transfers each gameweek. Therefore they are not able to simply pick a new team of 15 players every week that they think will score highest, without incurring a large points penalty to their total score. If there is sufficient time, I will aim to implement a transfer recommendation algorithm which will suggest transfers that a manager should make for a given gameweek, in order to maximise the expected points gain for their team.

Aside from raw statistics and data, another useful way to measure player performance is by analysing social media posts about players. This can help to gather information such as a recent injury for a player, or a player's recent performance in a different competition from the Premier League. We will

investigate how the addition of social media data affects the performance of the machine learning models.

Finally, by creating a web application to display team predictions or transfer recommendations, it will benefit FPL managers who need help with deciding which players to bring into their team.

1.4 Objectives

The objectives of the project were outlined in the progress report and project specification documents, but as the result of research and development that occurred during the earlier stages of the project they have been refined into the following list :

1. **Dataset Creation**
   Create the training and test datasets required to train and test the machine learning models, by using existing sources of data available online.
2. **Machine Learning Model Development**
   Develop and test different machine learning models and identify the best performing model for predicting player and team points.
3. **Analysis of Social Media Data**
   Obtain data from a suitable social media source and test different methods of analysing the data. Investigate how the addition of this data to the training and test datasets affects the points predictions of the machine learning models.
4. **Development of Web Application**
   Create a web application to be able to display the optimal team for a given gameweek.

Towards the second half of term two, an additional extension objective was created. This was to be able to create a transfer recommendation algorithm which would take a user's team as input and recommend players to transfer in and out of the team in order to increase the expected number of points that the team would score.

2 Literature Review

A number of academic papers and a web application were studied which were related to using machine learning to predict fantasy football points or analysing football related data from social media. These were useful in shaping the overall structure of the project and providing some comparable results for later on in the project. The most notable work has been explored in detail throughout this section.

2.1 Existing Literature

Bangdiwala et al [6] compared the performance of using a linear regression, decision tree and random forest model to predict FPL player points. The data for training and testing their models was collected from Vaastav's GitHub repository [7]. This repository contains data for every player in each gameweek from the 2016-17 to the 2024-25 Premier League seasons.

It is unclear which exact features were chosen to be used in the training and test data, since some post-processing was performed on Vastaav's dataset. There is a mention of creating features that use data enumerated over a fixed time window, such as "total minutes played in the last three games".

Root mean squared error (RMSE) and mean absolute error (MAE) were used to evaluate the performance of each model, by calculating the error between the number of fantasy points each player was predicted to score and the number of points that each player actually scored. Table 2 shows the results that were obtained.

Model	Train RMSE	Test RMSE	Train MAE	Test MAE
Linear Regression	2.172	2.159	1.290	1.264
Random Forest	0.261	2.146	0.025	1.215
Decision Tree	0.261	2.987	0.025	1.527

Table 2: Error metrics for each model produced by Bangdiwala et al [6]

It was concluded that the linear regression model had the best performance out of the models since it did not overfit on the training data as much as the random forest model, although the random forest model has the lowest RMSE and MAE on the test data out of the three models.

Valouxis [8] explored the use of random forest, XG boost and kernel ridge models to predict FPL player points. Similar to Bangdiwala et al [6], Vaastav's GitHub repository [7] was a key component used in creating the required training and test data used by Valouxis. He also also collected data from a couple of other sources, Understat [9] and the FiveThirtyEight Soccer Power Index [10] (please not that this website no longer exists and the citation includes an archive from Wayback Machine [11]). Understat is a website containing detailed statistics for players and teams in top European leagues, including the Premier League. These statistics include expected goals (xG), a metric which "measures the quality of a chance by calculating the likelihood that it will be scored by using information on similar shots in the past" [12]. Understat uses a neural network trained on over 100,000 shots to generate xG values for each shot that a player takes [9]. Unfortunately, Understat does not provide any information regarding the implementation of the neural network or the data used to train the model. Expected goals is a useful metric as it allows us to measure the quality of chances that players are receiving during games. For example, consider two arbitrary players : player A and player B. Suppose that both players have taken 10 shots in their last five games and player A has scored five goals but player B only scored one goal. On the surface, it appears that player A is just better at shooting than player B. However, suppose player A has an xG of 4.1 and player B has an xG of 1.3, over their last five games. Then we can see that chances that player A are receiving are of much higher likelihood to score, compared to player B. This could indicate that player A is shooting from better positions than player B, therefore scoring more goals. The FiveThirtyEight Soccer Power Index contained ratings for each Premier League team which indicated how strong a team was based on their recent performances. These ratings were then used as features in Valouxis' datasets. Unfortunately, the FiveThirtyEight Soccer Power Index stopped updating in 2023, and the FiveThirtyEight website has recently been shut down [13].

Valouxis trained an ensemble of six models which aimed to predict the number of non penalty goals a player will score, the number of assists a player will make, the number of penalties a player will score, the number of goals a player's team will concede (this was then converted into the probability

of the player's team keeping a clean sheet), the number of saves a player will make (if they are a goalkeeper) and the number of bonus fantasy points a player will earn in a match. A random forest, XG boost and kernel ridge model were tested to predict each of the six above values and then the best performing model for each value was used when calculating each player's final predicted fantasy points. The number of minutes a player is expected to play was also calculated, although it is not clear how this is done. The output from each model and the number of minutes a player is expected to play are then used to calculate a final expected fantasy points value for each player. This is done by multiplying the number of points gained or lost by each event in listed in Table 1 by the predicted value for each event.

The training data used was data from the 2019/20 to the 2021/22 Premier League season, and the test data used was data from the 2022/23 season until gameweek 25. This was an 80% - 20% train-test split. The final test error metrics that Valouxis obtained are shown in Table 3

MAE	RMSE	R^2
1.2898	2.3004	0.3334

Table 3: Valouxis' [8] final error metrics on his test data

Valouxis' approach of using multiple models highlights a different method of calculating predicted fantasy points, rather than using a single model like in the work produced by the Bangdiwala et al [6].

Bonello et al [14] investigated how the addition of data obtained from betting markets, Twitter, web articles and news blogs affected the accuracy of predicting an optimal FPL team. Initially, the train and test datasets consisted of statistical data collected using the FPL API. The label of the datasets was a binary variable indicating whether a player would score six or more points. Multiple machine learning models were considered, including support vector machines, random forests and gradient boosting machines. It was found that gradient boosting machines gave the best performance.

Tweets were collected for each player using the Tweepy Python library [15] and the sentiment score of each tweet was used as an additional feature to the datasets. However, it was found that this was not a viable strategy due to low performance accuracy of sentiment analysis models on tweets. This was mainly due to poor grammar and the use of emojis within tweets.

Betting odds were retrieved through api-football [16]. 100 Blog posts were retrieved through the Aylien API (depreciated since 2022) and named entity extraction was used to extract players that were mentioned in the posts. Sentiment analysis was then performed on the components of the posts related to each player that was mentioned in the post. It was found that a combination of betting odds data and sentiment analysis of blog posts gave the greatest performance increase of the model. The model trained on statistical data achieved an average team score of 52 points per gameweek whereas the addition of betting and sentiment analysis from blog posts yielded an average team score of 63 points per gameweek.

This source tells us that more robust methods, other than sentiment analysis, will be required to successfully extract useful data from sources such as Twitter (now known as X).

Finally, Godin et al [17] explored four different methods to try and accurately predict the outcome of football matches.

1. **Method One** - use purely statistical data about each team, such as each team's league position and recent form.
2. **Method Two** - investigate how the volume of tweets about each team before games affects predictions.
3. **Method Three** - use a support vector machine to classify tweets about each team before the game as positive, negative or neutral.
4. **Method Four** - aggregate the predictions of fans made on Twitter before the game.

It was found that a combination of each method gave the best overall accuracy of predictions. Methods one and four had the greatest individual accuracy whereas method two and three had the worst individual accuracy.

Although this paper aims to solve a different problem, it provides some useful insights into analysing twitter data. Similar to the paper by Bonello et al [14], sentiment analysis struggles to increase performance which illustrates that new methods are required for our project.

2.2 Existing Applications

Fantasy Football Hub (FFHub) [18] is one of the most popular online tools for assisting with FPL team selection. It provides features such as AI team reviews, transfer recommendations and player points predictions. Figure 3 shows part of the team review page.

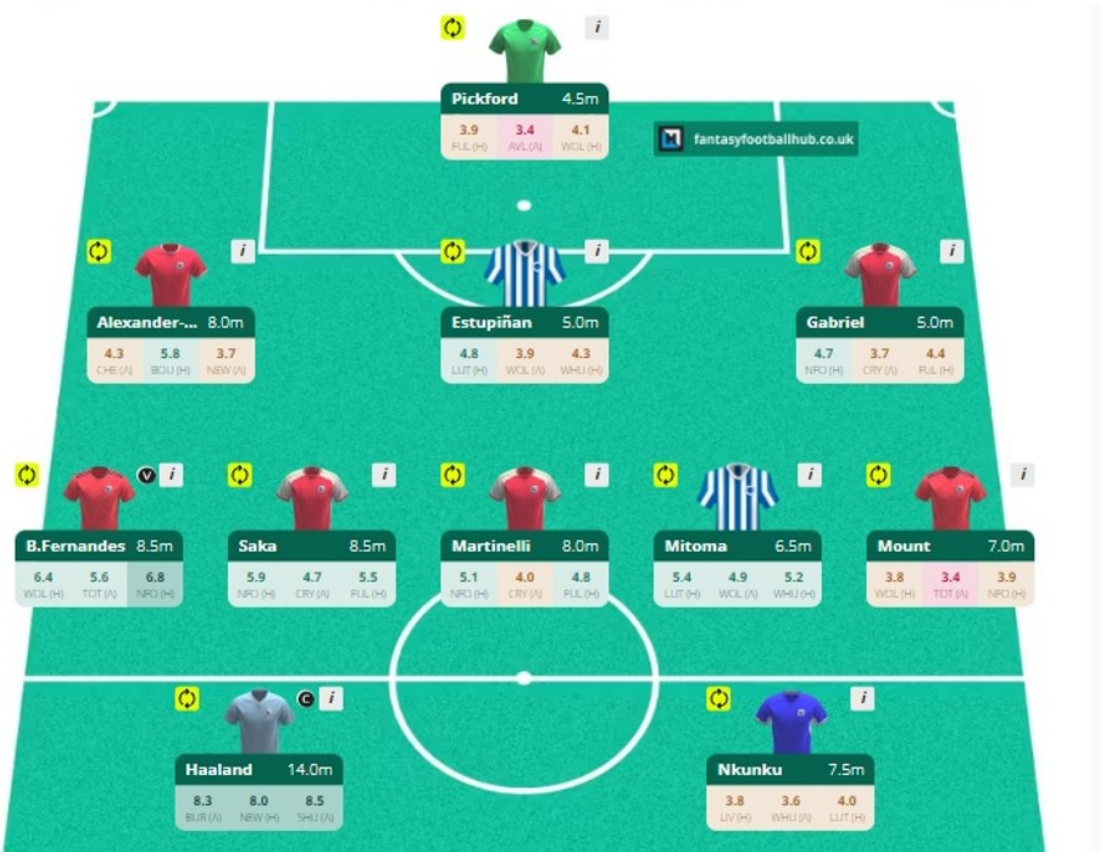

Figure 2: A screenshot from Fantasy Football Hub showing of a team of players with their predicted points for the next three games

Since FFHub is a premium, state-of-the-art product, it does unfortunately require a paid subscription in order to access the majority of features that are offered. Also, this means that there is very little information available regarding the details behind how the points predictions and transfer recommendations are generated, for example.

Although this makes it difficult to use as a comparison to our project, it does provide an example of some features that could be implemented into our web app, such as transfer recommendations, and definitely some scope for possible future work.

3 Methodology

3.1 Tools Used

This section outlines the main tools that were used throughout the the project and the reasons for choosing them.

3.1.1 Python & Google Colab

The development of all the code throughout the project was done using the Python [19] programming language. Python provides several libraries which were crucial to the development of the machine learning models that were produced. These include `sckit-learn`, which provides the implementations of many machine learning models and algorithms [20], `pandas` [21], which uses dataframes to allow the easy manipulation of `.csv` files, and `matplotlib`, which allows the simple creation of scientific plots [22]. Google Colab [23] was used as an environment to develop the machine learning models within. Google Colab allows free and paid access to GPUs and TPUs which was very useful to be able to run particularly computationally expensive models or algorithms in a reasonable time. The DCS Batch Compute System [24] was also used for when the allocated runtime of Google Colab was insufficient to run certain algorithms.

3.1.2 Flask

Flask [25] is a web framework written in Python which is used to develop web applications. Flask was used alongside HTML and CSS code to create the web application for the project.

3.1.3 GitHub & Google Drive

GitHub was used for version control for all code developed during the project and to ensure that all code could be accessed from any machine. All of the Google Colab notebooks were automatically backed up to Google Drive, as well as all of the datasets that were accessed or created during the project. This was essential to make sure that code would not be lost in case of failure of my laptop.

3.2 Project Management

3.2.1 Timeline

At the beginning of the project, a Gantt chart was created to outline the development plan for the first part of the project. Figure 3 shows the original Gantt chart for term one and Figure 4 shows the original Gantt chart for term two.

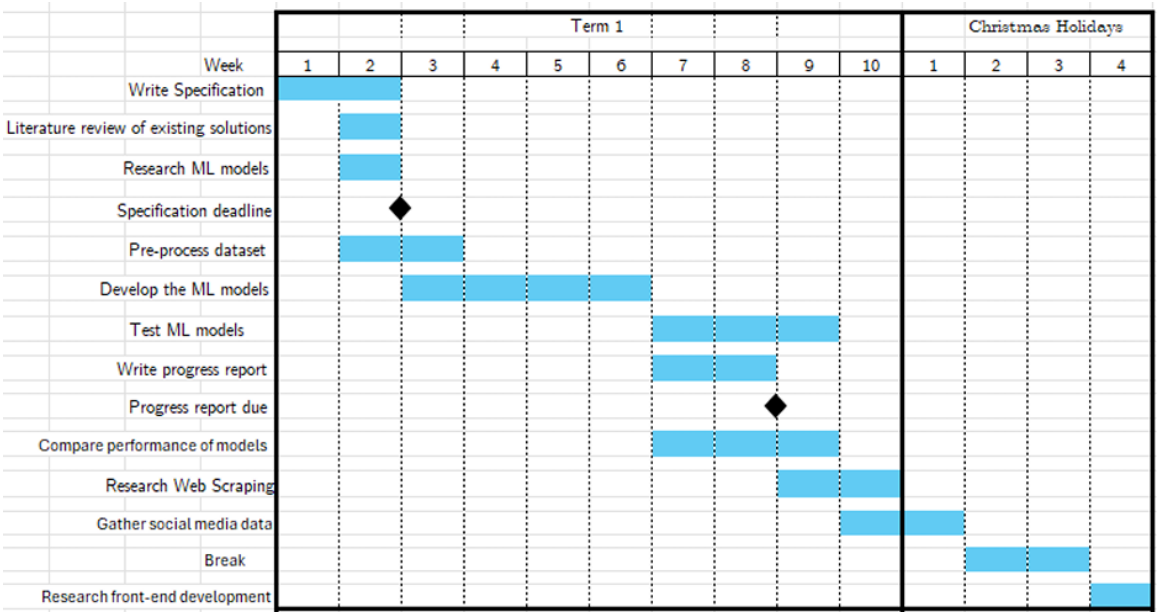

Figure 3: Term 1 Gantt Chart

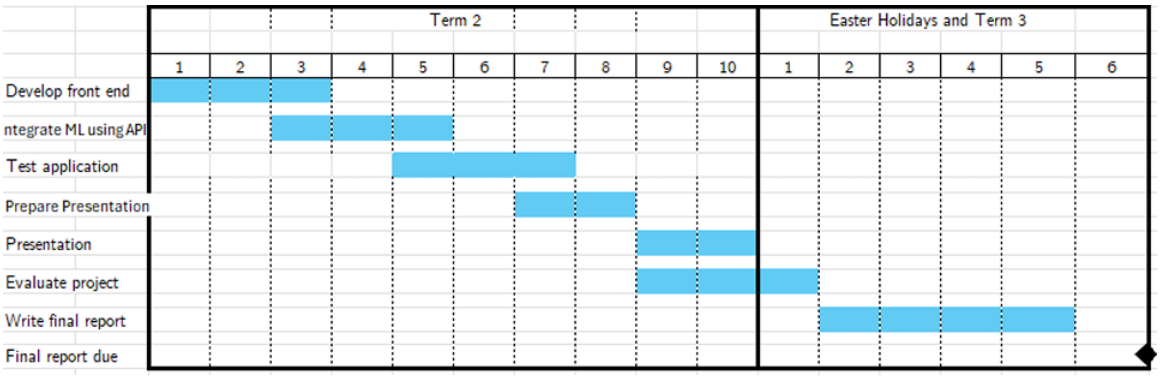

Figure 4: Term 2 Gantt Chart

Progress was fairly consistent with the objectives defined within the Gantt charts although there were a few changes made to the original schedule. The "Pre-process dataset" objective had to be redefined since it was realised that the data from Vastaav's GitHub repository could not solely act as the training and test datasets, and that they would have to be created as part of the early stages of the project. This meant more time had to be spent creating the dataset than was originally planned. However, the "Develop ML models" and "Test ML Models" objectives took significantly less time than expected so the additional time that was spent on creating the dataset did not delay the overall progress of the project. In addition, there were some unexpected challenges faced with the "Gather social media data" objective which meant its completion was delayed until term two. This is explained in more detail in Section 10. However, the "develop front end" and "integrate ML using API" objectives were completed faster than expected so this made up for any time that was lost with collecting the social media data.

3.2.2 Development Methodology

The primary development methodology that was followed during the project was an iterative waterfall methodology. This is an adaptation of the traditional waterfall methodology, in which development follows a linear approach with little flexibility to go back and amend previous completed stages. However, the iterative waterfall methodology incorporates feedback loops which allows previous stages of the project to be revisited if necessary. This was particularly useful when discussing the progress of the project with my supervisor, since improvements to previous stages were often made based on feedback or suggestions that were received. A key component of the iterative waterfall methodology is having clearly defined structure at the start of a project. By constantly staying up to date with timetable detailed in Section 3.22 and the objectives defined in Section 1.4, it ensured that project development rarely went off track.

3.3 Risks

There were multiple sources of risk associated with the project which are detailed below. Each risk includes a mitigation strategy which was devised to reduce the impact of the risk, in the event that it did end up occurring.

3.3.1 Availability of Data

The dataset that was created depended on data from multiple online sources, including Vastaav's GitHub repository and Understat. Therefore it was crucial that there was reliable access to these sources. To ensure this a clone of Vastaav's GitHub repository was made at the start of the project, so that if the access to the repository was removed then there would still be a local copy available. Any data collected from Understat was saved to GitHub and Google Drive so that it could still be accessed if the Understat website unexpectedly went down.

3.3.2 Loss of Data or Code

Any data that was collected was constantly saved to Google Drive and pushed to the project's GitHub repository so that it could be accessed from any machine in the event that my laptop was

not accessible. Similarly, any code developed on Google Colab or using VS Code was regularly pushed to the project's GitHub repository.

3.4 Legal, Ethical, Social and Professional Issues

For my project there were no legal, ethical, social or professional issues that needed to be considered. All of the sources of data that were used were publicly available online at the time of writing. In the Progress Report, there was mention of collecting user feedback after the development of the web application. However, it was decided that this was not necessary since the web application only formed a small component of the project and would not significantly benefit from user feedback in its current state.

4 Dataset Development

4.1 Creating the Dataset

As explored in the literature review, there are many available sources containing raw data and statistics related to Premier League players and teams, including Vastaav's GitHub repository [7] and Understat [9]. However, in order to train a machine learning model to predict player's fantasy points we require a dataset that contains statistics aggregated over previous games played by each player so that the model can learn how a player and their team's recent performances affect the number of points that they are likely to score.

There were no suitable publicly available and up-to-date datasets that could be found online, so it was decided that the training and test datasets would have to be created during the early stages of the project. The training dataset contained data collected from the 2021/22, 2022/23, 2023/24 seasons and the test dataset contained data collected from the 2024/25 season. Note that the 2024/25 PL season is ongoing at the time of writing so the test dataset contains data up until gameweek 30 of the season(the most recent completed gameweek). This gave an approximate 80%-20% train-test split.

4.1.1 Vastaav's GitHub Repository

As mentioned in the Literature Review, Vastaav's GitHub Repository includes data for every player and the match they played in each gameweek from the 2016-17 PL season until the 2024-25 PL season (the 2024-25 data is until gameweek 30 of the season).

For each season, there is a `gw.csv` file for every player who was a registered player on the FPL game for that season. This file contains data related to that player and their team for each gameweek of the season. Table 4 shows an example of the `gw.csv` file for Mohamed Salah for the 2023/24 PL season. Each row corresponds to data collected for a match that the player played in a certain gameweek and each feature (column) is an individual statistic, such as the number of goals the player scored in the match.

assists	bonus	bps	clean_sheets	creativity	element	expected_assists	expected_goals	fixture	goals_conceded	goals_scored	ict_index	influence	minutes
0	0	3	0	20.2	308	0.11	0.03	8	9	0	3.6	1.0	90
0	0	7	0	9.3	308	0.01	0.09	2	5	0	1.9	1.2	90
0	0	6	0	26.8	308	0.09	0.00	3	3	0	2.4	4.6	90
0	0	2	0	33.9	308	0.07	0.05	10	3	0	4.4	3.9	90
0	0	5	0	5.8	308	0.10	0.03	5	2	0	1.2	1.6	90
0	0	8	0	31.0	308	0.17	0.05	1	2	0	5.9	8.8	90
0	0	4	0	32.7	308	0.05	0.09	9	1	0	3.5	3.3	90
0	0	0	0	32.7	308	0.00	0.03	6	0	0	2.8	3.3	69
0	0	5	0	33.3	308	0.08	0.17	7	0	1	9.3	14.9	90

Table 4: A section of Mohamed Salah's data for the 2023/24 season

For each `gw.csv` file, some non-numerical features were removed, such as the date each game was played on, and any features that were highly correlated to other features were also removed. For example, the "expected_goal_involvements" feature is the sum of the "expected_goals" and "expected_assists" features so there is no additional information gained by including it. The complete list of features that were used from each player's `gw.csv` file are detailed in Table 5. A copy of each `gw.csv` file, with the features from Table 5, was created and stored as `gw_v2.csv`. This was done for every player's `gw.csv` file for the 2021/22, 2022/23, 2023/24 and 2024/25 seasons.

Feature Name	Description
name	The name of the player
element	The player's FPL ID (a unique identifier for the player)
gameweek	The gameweek number in which the game is being played in
position	The player's registered position in FPL
goals	The number of goals that the player scored in the match
assists	The number of assists the player made in the match
bonus	The number of bonus FPL points the player was awarded at the end of the match [5]
clean_sheets	A binary variable denoting if the player kept a clean sheet in the match (one if they did, zero otherwise).
expected_goals	The number of goals the player was expected to score in the match
expected_assists	The number of assists the player was expected to make in the match (the likelihood that a pass becomes a goal assist).
shots	The number of shots the player took during the match
key_passes	The number of key passes (a pass to a teammate which results in a shot on goal) a player made during the match
ict_index	A metric that summarises a player's influence, creativity and threat in the match [26]
minutes	The number of minutes that the player played in the match
yellow_cards	The number of yellow cards the player received in the match
red_cards	The number of red cards the player received in the match
saves	The number of saves the player made in the match (only relevant for goalkeepers)
total_points	The number of fantasy points the player scored for the match
transfers_in	The number of times the player was transfered in to a manager's FPL team before the match
transfers_out	The number of times the player was transfered out of a manager's FPL team before the match
was_home	A binary variable denoting if the match was played at the home (at the player's team's stadium) or away (at the opposition team's stadium)

Table 5: The data collected for each player from Vastaav's GitHub repository

Note that the feature "position" is normalised to an integer between one and four. One represents the goalkeeper position, two is the defender position, three is the midfielder position and four is the forward position.

4.1.2 Understat

An important factor in predicting a player's match performance is the ability and current form of the team that the player plays for, and the opposition team that the player is playing against. The data collected in Section 4.1.1 is almost entirely player specific so team based data was collected using Understat.

In order to collect data from Understat, the `understat` Python library was used [27]. This library contains multiple functions that allow the data which is shown on the Understat website to be accessed in a JSON format.

I created a `getSeasonGames()` function which required three parameters : the name of a PL team, the season to collect data for and whether to collect data for their home games, away games or all games for that season. This function then calls the `get_team_results()` function from the understat library and its output is converted from the JSON format into a pandas dataframe. Table 6 shows the data that was collected using these functions. This data was stored in a `.csv` file for every team that was in the PL for each of the 2021/22, 2022/23, 2023/24 and 2024/25 seasons. e.g. The data collected for all of Chelsea's matches in the 2024/25 season was stored as `Chelsea_2024_all.csv`.

Feature	Description
Opposition Team	Name of the opposition team that the team is playing against
Team Goals Scored	Number of goals scored by the team in the match
Team Goals Conceded	Number of goals scored by the opposition team in the match
Team xG Scored	Expected number of goals scored by the team in the match
Team xG Conceded	Expected number of goals scored by the opposition team in the match

Table 6: The data collected for each PL team from Understat

In order to join each player's data collected in Section 4.1.1, with their team's data collected above, it was necessary to have a feature for the team(s) that the player played for each season. This was obtained using the `get_league_players()` understat function, which returns a list of players and some corresponding data, that played in a specified league and season. The teams that each player played for were extracted from this data. Then for every player's `gw_v2.csv` file created in Section 4.1.1, an additional `Team` feature was added which denotes the team that the player played for in each gameweek of the season. For the majority of players this value will be the same for each gameweek since players typically play for one team during a season.

However, one problem that was faced was that it is possible for players to play for multiple teams in the same league during a season. This is because in the PL there are two transfer windows which run from June-August and January-February that allow players to transfer between different teams. To solve this I had to find all players, in the seasons that I collected data for, that had played for multiple teams. Then using each player's Wikipedia page I found the exact date that they transfered between teams and matched this to the most recent completed gameweek. The `Team` feature was then added accordingly.

4.1.3 Match Odds

Another useful source of information in predicting a player's match performance are the betting odds for the outcome of a match. Football-Data [28] is a free online betting portal which provides historical football results and odds. For each of the 2021/22, 2022/23, 2023/24 and 2024/25 PL seasons there is a `.csv` file containing extensive match statistics and betting odds for each match in the season. Table 7 details the data that was collected from each of these files.

Feature	Description
Home Team	Name of the team playing at home
Away Team	Name of the team playing away
AvgH	The average odds for the home team to win
AvgD	The average odds for the match to end in a draw
AvgA	The average odds for the away team to win

Table 7: The data collected for each PL match from Football-Data [28]

Note that the odds are averaged over 15 different betting markets and are stored in decimal form. The collected odds were then converted from decimal form to probabilities, and normalised, so that for each match the probability for winning, drawing and losing sum to one. Decimal odds are converted to probabilities by calculating the reciprocal of the decimal odds.

Then for each player's `gw_v2.csv` file from Section 4.1.1, their corresponding team data from Table 6 and Table 7 was concatenated with the file.

4.1.4 Previous Season Performance

The final data that was collected were statistics about each player's performance in the previous season that they played. During the early parts of seasons, it can be difficult to predict player performance when there have not been many games played so far. How a player performed in the previous season is often a good indicator of how well they are likely to perform in the current season. The `understat` Python library was once again used to collect previous season data for each player for each of the 2021/22, 2022/23, 2023/24 and 2024/25 PL seasons. Table 8 details the data that was collected.

Feature	Description
Previous season goals per 90 mins	The average number of goals scored per 90 minutes played
Previous season xG per 90 mins	The average number of expected goals scored per 90 minutes played
Previous season assists per 90 mins	The average number of assists made per 90 minutes played
Previous season xA per 90 mins	The average number of expected assists made per 90 mins
Previous season shots per 90 mins	The average number of shots taken per 90 mins played
Previous season key passes made per 90 mins	The average number of key passes made per 90 minutes played
Previous season yellow cards	The average number of yellow cards received per 90 minutes played
Previous season red cards	The average number of red cards received per 90 minutes played
Previous season minutes	The average number of minutes played in each match
Previous season clean sheets	The total number of clean sheets kept (only counting games which the player started in)

Table 8: The data collected for each player's previous season performance

4.1.5 Aggregating the Data

Currently the data that we have collected in the previous sections contains statistics for every player and their team's performance in each game that they play in a given season. In order to predict how many fantasy points that a player will score in an upcoming game, we need historical data about how a player and their team has been performing, as well as data about the upcoming game.

The final step to create the training and test datasets was to combine and aggregate all the data collected in each player's `gw_v2.csv` file for each season that we collected data for. A function named `generateSeasonData()` which has a single parameter, the season number e.g. 2021 for the 2021/22 season, was created to do this. To create the dataset for one season, every player who played during that season `gw_v2.csv` file was iterated over and up to 38 feature vectors were created (Each PL season consists of 38 gameweeks so a player can play in maximum 38 games for their team). Each feature vector corresponds to a match that the player played in during the season and the features consist of historical data averaged over previous games in the season for the player and their team, specific data about the match, and the label which is the number of FPL points that the player scored in that match. The data generated for the 2021/22, 2022/23 and 2023/24 seasons was concatenated together to form the training dataset and the data generated for the 2024/25 season was used as the test dataset. The training dataset contains 59365 feature vectors and and the test dataset contains 15164 feature vectors. This gives an approximate 80% - 20% train-test split. All feature vectors are 125 dimensions.

Table 9 details all the features in the training and test datasets and the type of each feature. Each feature is classed as one of five types :

1. Features that provide additional information about a player or were used as part the process to create the datasets. These are not used during the training and testing of any models in this report.
2. Features which are a statistics about how a player performed in the match. For example, the number of goals a player scored in the game is only known after the game has finished and not before the game. Therefore, these are also not used during the training and testing of models but are useful for future analysis of the datasets.
3. Features which give an indication of a player or their team's recent performances. For the majority of features of this type, five separate features are included in the datasets, which are are average the feature over the player or team's previous game played, previous three games played, previous five games played, previous ten games played and all games played so

far in the current season. This was chosen to give an overview of the player and their team's short term and long-term form during the season. For example, for the "Goals scored per 90 minutes" feature there are five separate features for "Goals scored in the previous game", "Goals scored per 90 minutes averaged over the previous three games", "Goals scored per 90 minutes averaged over the previous five games", "Goals scored per 90 minutes averaged over the previous ten games" and "Goals scored per 90 minutes averaged over all games played so far in the current season". Note that the number of games each feature has been averaged over are indicated in brackets below the feature name in Table 9.

4. Features which are specific to the match the player is playing in. For example, the win probability feature is dependent on the opposition team which the player's team are playing in the match.

5. Features which are statistics about a player's performance in the previous season which they played in. All features are set to zero if the current season is the first season that the player is playing professional football.

Feature	Description	Feature Type
Name	Full player name	1
Web Name	The name which a player is most commonly known by online	1
FPL ID	The player's unique numerical identifier in FPL	1
Team	The team that the player plays for	1
Gameweek	The gameweek number that the match was played in	1
Date	The date the match was played and the time it began	1
Goals	Number of goals scored by the player in the match	2
Assists	Number of assists the player made in the match	2
Clean Sheet Kept	A binary variable denoting if the player kept a clean sheet in the match (one if they did, zero otherwise)	2
Saves Made	The number of saves made by the player (if they a goalkeeper)	2
Bonus Scored	The number of bonus points scored by the player at the end of the match	2
Minutes Played	The number of minutes played by the player in the match	2
FPL Points Scored	The number of FPL points scored by the player in the match (The label we want to predict)	2
Average FPL Points Scored (averaged over the current season)	The average number of FPL points the player has scored per match	3
Goals scored per 90 minutes (averaged over previous 1,3,5 and 10 games and current season)	The average number of goals scored by the player every 90 minutes they play	3
Expected goals scored per 90 minutes (averaged over previous 1,3,5 and 10 games and current season)	The average number of expected goals scored by the player every 90 minutes they play	3
Shots taken per 90 minutes (averaged over previous 1,3,5 and 10 games and current season)	The average number of shots taken by the player every 90 minutes they play	3
Assists made per 90 minutes (averaged over previous 1,3,5 and 10 games and current season)	The average number of assists made by the player every 90 minutes they play	3

Feature	Description	Feature Type
Expected number of assists made per 90 minutes (averaged over previous 1,3,5 and 10 games and current season)	The average number of expected assists made by the player every 90 minutes they play	3
Key passes made per 90 minutes (averaged over previous 1,3,5,10 games and current season)	The average number of key passes made by the player every 90 minutes they play	3
Minutes played (averaged over previous 1,3,5,10 games and current season)	The average number of minutes the player plays each match	3
Yellow Cards per 90 minutes (averaged over current season)	The average number of yellow cards received by the player every 90 minutes they play	3
Red Cards per minutes (averaged over current season)	The average number of red cards received by the player every 90 minutes they play	3
ICT score per 90 minutes (averaged over previous 1,3,5 and 10 games and current season)	The average ICT score [26] achieved by the player every 90 minutes they play	3
Saves (averaged over previous 1,3,5 and 10 games and current season)	The average number of saves made by the player each game (only applicable to goalkeepers)	3
Clean Sheets (totalled over previous 1,3,5 and 10 games and current season)	The total number of clean sheets kept	3
Team goals scored (averaged over previous 1,3,5 and 10 games and current season)	The average number of goals scored each game by the player's team	3
Team goals conceded (averaged over previous 1,3,5 and 10 games and current season)	The average number of goals conceded each game by the player's team	3
Expected team goals scored (averaged over previous 1,3,5 and 10 games and current season)	The average number of expected goals scored each game by the player's team	3
Expected team goals conceded (averaged over previous 1,3,5 and 10 games and current season)	The average number of expected goals conceded each game by the player's team	3
Opposition goals scored (averaged over previous 1,3,5 and 10 games and current season)	The average number of goals scored each game by the opposition team	3

Feature	Description	Feature Type
Opposition goals conceded (averaged over previous 1,3,5 and 10 games and current season)	The average number of goals conceded each game by the opposition team	3
Opposition expected goals scored (averaged over previous 1,3,5 and 10 games and current season)	The average number of expected goals scored each game by the opposition team	3
Opposition expected goals conceded (averaged over previous 1,3,5 and 10 games and current season)	The average number of expected goals conceded each game by the opposition team	3
Home	A binary value indicating if the game is being played at the player's home stadium (1) or away (0)	4
Transfers In	The total number of times the player has been transfered in to a manager's FPL team between the end of the previous gameweek and the start of the current gameweek	4
Transfers Out	The total number of times the player has been transfered out of a manager's FPL team between the end of the previous gameweek and the start of the current gameweek	4
Win Probability	The probability of the player's team winning the match	4
Draw Probability	The probability of the player's team drawing the match	4
Position	An integer between 1 and 4 representing the player's position in FPL	4
Previous season goals	The number of goals the player scored per 90 minutes in the previous season that they played in	5
Previous season assists	The number of assists the player made per 90 minutes in the previous season that they played in	5
Previous season expected goals	The number of goals the player was expected to score per 90 minutes in the previous season that they played in	5
Previous season expected assists	The number of assists the player was expected to make per 90 minutes in the previous season that they played in	5
Previous season key passes	The number of key passes the player made per 90 minutes in the previous season they played in	5
Previous season yellow cards	The number of yellow cards the player received per 90 minutes in the previous season that the player played in	5
Previous season red cards	The number of red cards the player received per 90 minutes in the previous season that the player played in	5
Previous season minutes	The average number of minutes the player played in each game they played in the previous season	5
Previous season clean sheets	The total number of clean sheets the player kept when they started the game for each game played in the previous season	5

Table 9: The features in the training and test datasets. The feature types correspond to the types of features in the list above the table

5 Machine Learning Models

Machine learning was chosen as the general technique to predict player FPL points due to the ability of machine learning models to identify patterns and trends within datasets which can be used to inform predictions. Although the number of FPL points that players score are always integer values, only regression models were investigated, which output a real predicted value, instead of classification models which would aim to classify a player's predicted points as an exact integer value. This is because all previous work explored in the Literature Review used regression models so it provides a natural comparison for our future results. However, this is not to say that using classification models would be a bad approach as this could definitely be explored as part of future work regarding FPL points predictions.

The machine learning models that were chosen to use were based on two factors: initial testing of regression models and the models used in the explored literature. The `lazypredict` [29] Python library allows 40 different types of machine learning regression models to be trained and tested on any training and test datasets provided by the user. This is a useful tool in evaluating the baseline performance of multiple types of models on any datasets. All of the models are trained without any hyperparameter tuning, so the output from `lazypredict` is purely used to get an idea of the most suitable models, and not return the definitive best performing models. The `LazyRegressor` function was used to train and test each regression model on the training and test datasets that were created in Section 4.1.5. Each model is evaluated on its adjusted R-squared score (adjusted R^2), R-squared score (R^2) and RMSE between each player's predicted points and the actual points they scored. R-squared score, Adjusted R-squared score, and RMSE are defined in Equation 1, Equation 2 and Equation 3 respectively. The R^2 score of a model describes the proportion of variance in the labels that are explained by the model [30]. The maximum R^2 score is one, and the higher the R^2 score the better fit the trained model is to the data. Adjusted R^2 score is the same as R^2 score but it penalises the presence of features which do not contribute significantly to the value of the predicted labels. RMSE is the square root of the averaged squared differences between the predicted labels of the data and the true labels. This helps to describe how close the predicted labels are to the true labels, on average. The output of the `LazyRegressor` function is shown below in Table 10, with the best performing models at the top of the list.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{1}$$

y_i = Label of feature vector i

$\hat{y}_i$ = Predicted label of feature vector i

$\bar{y}$ = The mean of all feature vector labels

$$\text{Adjusted } R^2 = 1 - \left(\frac{(1 - R^2)(n - 1)}{n - p - 1}\right) \tag{2}$$

$R^2 = R^2$ score of the model

n = Number of feature vectors in the dataset

p = Number of features in the dataset

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3}$$

The top three performing models are all variations of Gradient Boosting Regressors. The Random Forest model and Linear regression also score highly and these were both explored in the earlier Literature Review Section. Therefore, the Histogram Gradient Boosting, Random Forest and Linear Regression models were chosen to explore in more detail during the project. Each model was implemented using its corresponding class from the `scikit-learn` [20] library.

Model	Adjusted R-Squared	R-Squared	RMSE
HistGradientBoostingRegressor	0.29	0.30	2.21
GradientBoostingRegressor	0.29	0.30	2.21
LGBMRegressor	0.28	0.29	2.22
RandomForestRegressor	0.27	0.28	2.24
ExtraTreesRegressor	0.27	0.28	2.27
Lars	0.25	0.26	2.28
TransformedTargetRegressor	0.25	0.25	2.28
LinearRegression	0.25	0.25	2.28
Ridge	0.25	0.25	2.28
RidgeCV	0.25	0.25	2.28
BayesianRidge	0.25	0.25	2.28
LassoCV	0.25	0.25	2.28
LarsCV	0.25	0.25	2.28
LassoLarsCV	0.25	0.25	2.28
LassoLarsIC	0.25	0.25	2.28
ElasticNetCV	0.25	0.25	2.28
XGBRegressor	0.24	0.25	2.29
OrthogonalMatchingPursuitCV	0.24	0.25	2.29
OrthogonalMatchingPursuit	0.24	0.25	2.29
TweedieRegressor	0.23	0.24	2.31
MLPRegressor	0.01	0.02	2.61
NuSVR	0.21	0.21	2.62
SVR	0.20	0.20	2.62
BaggingRegressor	0.19	0.20	2.58
SGDRegressor	0.10	0.12	2.67
HuberRegressor	0.16	0.14	2.64
ElasticNet	0.14	0.14	2.71
KNeighborsRegressor	0.14	0.14	2.74
LinearSVR	0.04	0.05	3.50
LassoLars	0.00	0.00	3.59
Lasso	0.04	0.04	3.56
DummyRegressor	-0.01	-0.01	3.77
QuantileRegressor	-0.06	-0.06	4.14
KernelRidge	-0.11	-0.12	4.45
GaussianProcessRegressor	-0.11	-0.11	4.52
PassiveAggressiveRegressor	-0.25	-0.24	5.39
ExtraTreeRegressor	-0.35	-0.34	5.17
DecisionTreeRegressor	-0.47	-0.46	5.41
AdaBoostRegressor	-0.35	-0.34	4.97
RANSACRegressor	-3.63	-3.60	5.67

Table 10: Results output from the `LazyRegressor` function

Hyperparameter tuning was performed on the HGBM model and the RF model using the scikit-learn `GridSearchCV` class. The LR model did not require any hyperparameter tuning. For each model, a `GridSearchCV` object was initialised, which requires three parameters. The first parameter was the instance of the model which was being tuned. The second parameter was a dictionary of hyperparameter values. Each dictionary key was the name of the hyperparameter, and its corresponding value was an array of possible values that the hyperparameter could take. The third parameter was the number of cross-validation folds to use. This was set to the default value of five to ensure enough folds were averaged over and that there was a wide range of player FPL points scores (the labels) within each fold. The number of available CPU cores was also passed as a parameter as this allowed parallel processing to be utilised, which decreased the overall runtime of the hyperparameter tuning. The `fit()` method of each `GridSearchCV` object was then called, with the training data and labels passed as parameters. For each possible combination of hyperparameters, the training data is split into five folds where each fold is used exactly once as a test dataset, and the remaining four folds are used to train the model. The R^2 score of the model is averaged over the R^2 score of each test fold to give an overall R^2 score for that combination of hyperparameters. After all hyperparameter

combinations have been evaluated, the set of hyperparameters which gave the greatest mean R^2 score over all five test data folds was returned as the optimum hyperparameters for that model.

To evaluate the performance of each of the models, three different metrics were used. R^2 score and RMSE were used, which were defined previously in equations 1 and 3 respectively. Mean absolute error (MAE) was also used as the third metric which is defined below in equation 4. MAE describes the average difference between the predicted labels of each model and the actual labels. MAE is similar to RMSE, but it is less sensitive to when there is a large difference between a predicted label and an actual label, since the difference is not squared like when using RMSE. R^2 score is the default scoring metric used by each model's `scikit-learn` implementation during model training. Each of the metrics were used to evaluate the models explored in the Literature Review so it provides a natural comparison for our future results.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i) \tag{4}$$

$$n = \text{Number of feature vectors in the dataset}$$
$$y_i = \text{Label of feature vector } i$$
$$\hat{y}_i = \text{Predicted label of feature vector } i$$

5.1 Linear Regression (LR)

Linear regression models work by trying to find a linear relationship between the features in a dataset and the value of the labels for each feature vector. A weight is assigned to each feature during the model training, which defines how the value of the feature contributes to the value of the label. The model can then be used to predict labels for unseen test data by computing the weighted sum of the feature vectors in the test dataset. As mentioned previously, there were no hyperparameters required to tune for the linear regression model. The linear regression model was implemented using the `sklearn.linear_model.LinearRegression` class.

5.2 Random Forest (RF)

Random forest models are a type of ensemble method, based on a popular machine learning concept named bagging. Bagging refers to training the same type of model on independent subsets of the same dataset. This is used to reduce the amount of overfitting on the training data. Random forests are a collection of regression trees whose predictions are averaged to give an overall prediction. Each regression tree is trained on a random sample of the features from the training data. The trees are formed by recursively splitting the training data, by using the feature value which gives the lowest error over both parts of the split in each iteration. This is repeated until some pre-defined rule is violated, such as the maximum depth of the tree is reached. The random forest model was implemented using the `sklearn.ensemble.RandomForestRegressor` class.

5.2.1 Hyperparameters

The hyperparameters that were tuned for the RF model were `min_samples_split` and `n_estimators`. `min_samples_split` defines the minimum number of feature vectors that must be present in a split to continue splitting it further. This helps to ensure that the depth of the tree is not too large, hence reducing overfitting on the training data. `n_estimators` controls the number of regression trees that are fitted in the ensemble, which also reduces overfitting. Table 11 shows the hyperparameter values which were tested and the optimal hyperparameters which were found.

	Hyperparameter Tuned	
	`min_samples_split`	`n_estimators`
Values Tested	[20, 40, 60, 80, 100, 200, 300, 400, 500]	[70,80,90,100,110,120,130]
Optimal Value	100	120

Table 11: Hyperparameter values tested for the random forest model

5.3 Histogram Gradient Boosting Machine (HGBM)

Gradient Boosting is another type of ensemble method, based on another popular machine learning technique named boosting. Boosting refers to sequentially training the same type of model, where each model aims to correct the errors made by the previous models in the ensemble. The models used in gradient boosting are regression trees. Each tree is trained to be maximally correlated with the negative gradient of the of the loss function for the current ensemble [31]. The trees are recursively split on the entire training data by finding the optimal feature split during each iteration. Finding the optimal feature split is a very time consuming process since all possible splits must be iterated over.
The HGBM model is a variant of the traditional gradient boosting model suitable for larger datasets with over 10000 feature vectors [32]. It provides much faster training and test speeds, without a reduction in model performance. For each feature, its set of unique values are grouped into intervals which are then used as bins for a histogram. This significantly reduces the number of unique values that have to be checked when finding the best feature split during the regression tree training. The HGBM model was implemented using the `sklearn.ensemble.HistGradientBoostingRegressor` class.

5.3.1 Hyperparameters

The hyperparameters that were tuned for the HGBM model were `learning_rate`, `max_iter`, `max_bins` and `min_samples_leaf`. `learning_rate` is a multiplicative value for the values of the leaves in each regression tree, which controls how fast the model learns. A lower learning rate reduces overfitting but leads to longer training times. `max_iter` controls the number of regression trees which are fitted. `max_bins` controls the number of bins used within each histogram and has a maximum value of 255. `min_samples_leaf` defines the minimum number of feature vectors that must be present in a given split of a regression tree in order to split it further. All of the three hyperparameters mentioned help to give a balance between increasing training performance and reducing overfitting. Table 12 shows the hyperparameters which were tested and the optimal hyperparameter values which were found.

	Hyperparameter Tuned			
	`learning_rate`	`max_iter`	`max_bins`	`min_samples_leaf`
Values Tested	[0.01, 0.02, 0.025, 0.05, 0.1, 0.2, 0.4, 0.8]	[50, 100, 150, 200, 250, 300]	[190, 200, 210, 220, 230, 240, 255]	[20, 40, 60, 80, 100, 200, 300, 400, 500]
Optimal Value	0.025	250	255	300

Table 12: Hyperparameter values tuned for the HGBM model

5.4 Model Results

Each of the LR, RF and HGBM models were trained and tested on the training and test datasets, created in Section 4.1.5, with each model's set of optimal hyperparameters. Table 13 shows the performance of each model on the training data and Table 14 shows the performance of each model on the test data. The best models for each metric are highlighted in bold. For the RF and HGBM models there is a small amount of randomness involved during training, so all metrics are averaged over five instances of each model.

Model Type	**Train RMSE**	**Train R2 Score**	**Train MAE**
HGBM	2.139	0.346	1.278
LR	2.308	0.238	1.425
RF	**1.869**	**0.501**	**1.123**

Table 13: Train error and R^2 Score For each model

Model Type	Test RMSE	Test R2 Score	Test MAE
HGBM	**2.254**	**0.255**	**1.369**
LR	2.292	0.230	1.404
RF	2.281	0.237	1.427

Table 14: Test error and R^2 Score for each model

In Table 13, we can see that the RF model clearly outperformed the HGBM and LR models on the training data, since it has the lowest RMSE and MSE values as well as the highest R^2 score. However, in Table 14, the best performing model was the HGBM model. This suggests that the RF model was slightly overfitting to the training data whereas the HGBM model was more robust to the unseen test data. The LR model had the worst performance across both the training and test data.

6 Dataset Analysis

In order to try and improve the results obtained in Section 5.4, some key properties of the training and test datasets were identified.

During a football match there are 22 players playing on the pitch at any one time (11 from both teams). In the Premier League, teams are allowed to make up to five substitutions during the match, in which they can swap a player that is currently on the pitch with a player that is on the bench. This means that over the entire course of a match, there can be up to 32 different players that have played. In FPL, the most common ways that a player will earn at least three points is by scoring a goal, providing an assist or keeping a clean sheet (if they are a defender or goalkeeper). Figure 5 shows the number of goals scored in each match during the 2021/22, 2022/23 and 2023/24 PL seasons.

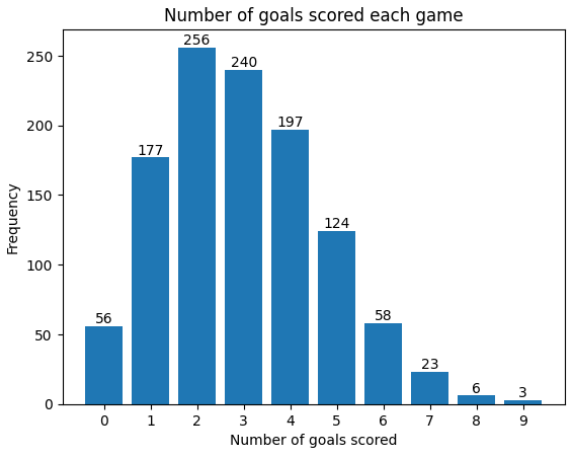

Figure 5: The number of goals scored in each game of the 2021/22, 2022/23 and 2023/24 seasons

We can see that the majority of matches only have two or three goals in. The average number of goals scored per game is 2.98. Figure 6 shows the number of goals scored by each individual player during each match.

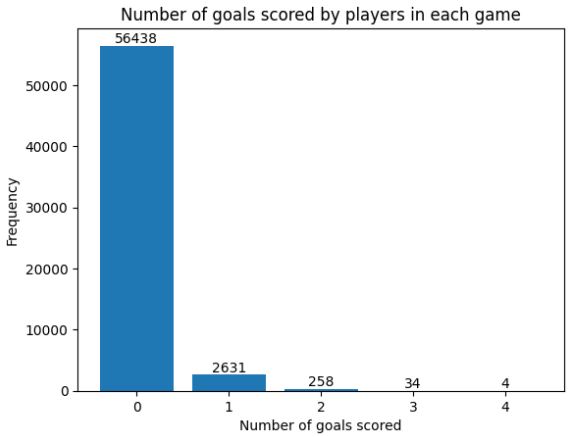

Figure 6: Total number of goals scored by each player in each game of the 2021/22, 2022/23 and 2023/24 PL seasons

Figure 10 highlights that during a match most players will not score any goals, and if they do then they typically only score one. The number of assists made by players each game follows an almost identical distribution, as shown by Figure 7. This is because each goal that is scored is usually assisted by another player on the same team, unless the ball is diverted to the goalscorer by an opposition player, in which case no assist is awarded. An assist for a goal can only be awarded to one player. Figure 8 shows the number of clean sheets kept by teams across all games in the

2021/22, 2022/23 and 2023/24 PL seasons. Although a clean sheet was kept in almost half the matches played, in FPL only defenders, midfielders and goalkeepers which play at least 60 minutes during the game earn points for keeping a clean sheet (four points for defenders and goalkeepers, and one for midfielders). A team usually starts with between three and five defenders, and one goalkeeper on the pitch so only these players will earn four points for a clean sheet if one is kept.

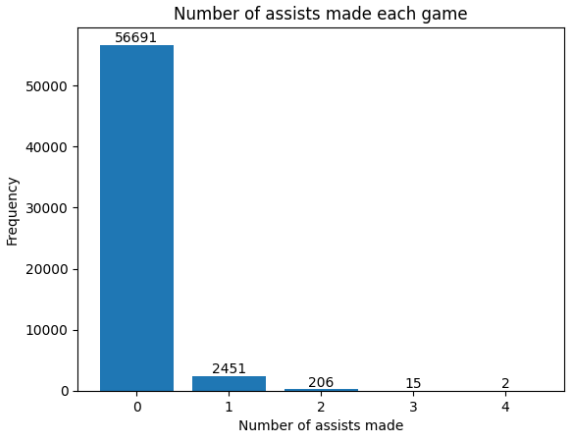

Figure 7: Total number of assists made by each player in each game of the 2021/22, 2022/23 and 2023/24 PL seasons

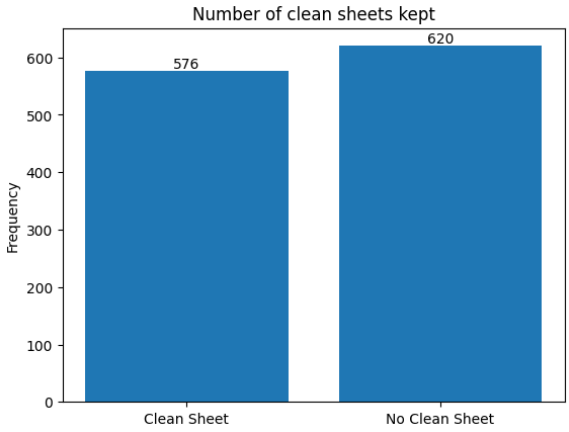

Figure 8: Total number of clean sheets kept by teams in the 2021/22, 2022/23 and 2023/24 PL seasons

To summarise, we can see that the most common actions which players earn FPL points by are only completed by a small fraction of players each game. The number of minutes which a player plays also has an impact on how many points they are likely to earn. Figure 9 shows the number of minutes played by players across all games in the 2021/22, 2022/23 and 2023/24 seasons, and the average points scored by players in each histogram bin. We can see that the data is dominated by players which played under 30 minutes and that the average number of points scored increases significantly for players who play over 60 minutes. Therefore, in each game a significant proportion of players will end up scoring below two FPL points. This is highlighted by Figure 10, which shows the distribution of FPL points scored by each player in our training dataset. There is a significant class imbalance; the number of player's that score zero, one or two points is much greater than the players that score in the other points classes. This suggests that our models are struggling to predict players that end up scoring highly in the test data, due to the lack of representation of high scoring players in the training dataset.

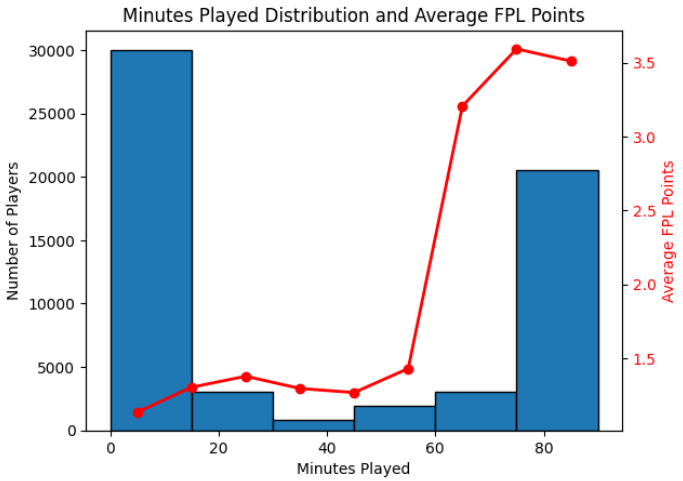

Figure 9: Histogram showing the number of minutes played by players in each game of the 2021/22, 2022/23 and 2023/24 PL seasons, overlaid with the average points earnt by players in each histogram bin

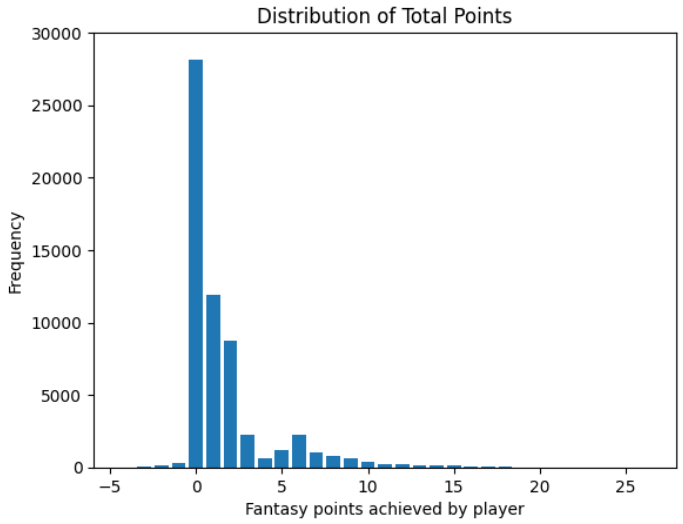

Figure 10: Distribution of total FPL points scored by players in each game of the 2021/22, 2022/23 and 2023/24 PL seasons (the seasons in the training dataset)

A useful way of viewing the effect of the class imbalance is by using a confusion matrix. A confusion matrix is a table that can be used to visualise the performance of a classification model by comparing the frequency that each class in the dataset is predicted versus the frequency that each class actually occurs in the labels of the dataset. Since our problem is a regression task, we can round each player's predicted FPL points to the nearest integer and compare it to the number of FPL points that they actually scored. All confusion matrices were plotted using the `sklearn.metrics.ConfusionMatrixDisplay` class. Each entry of the confusion matrix C was normalised so that $C(i, j)$ represents the fraction of players that were predicted to score j points but who actually scored i points. Figure 11 shows the normalised confusion matrix for the test points predictions made by the HGBM model.

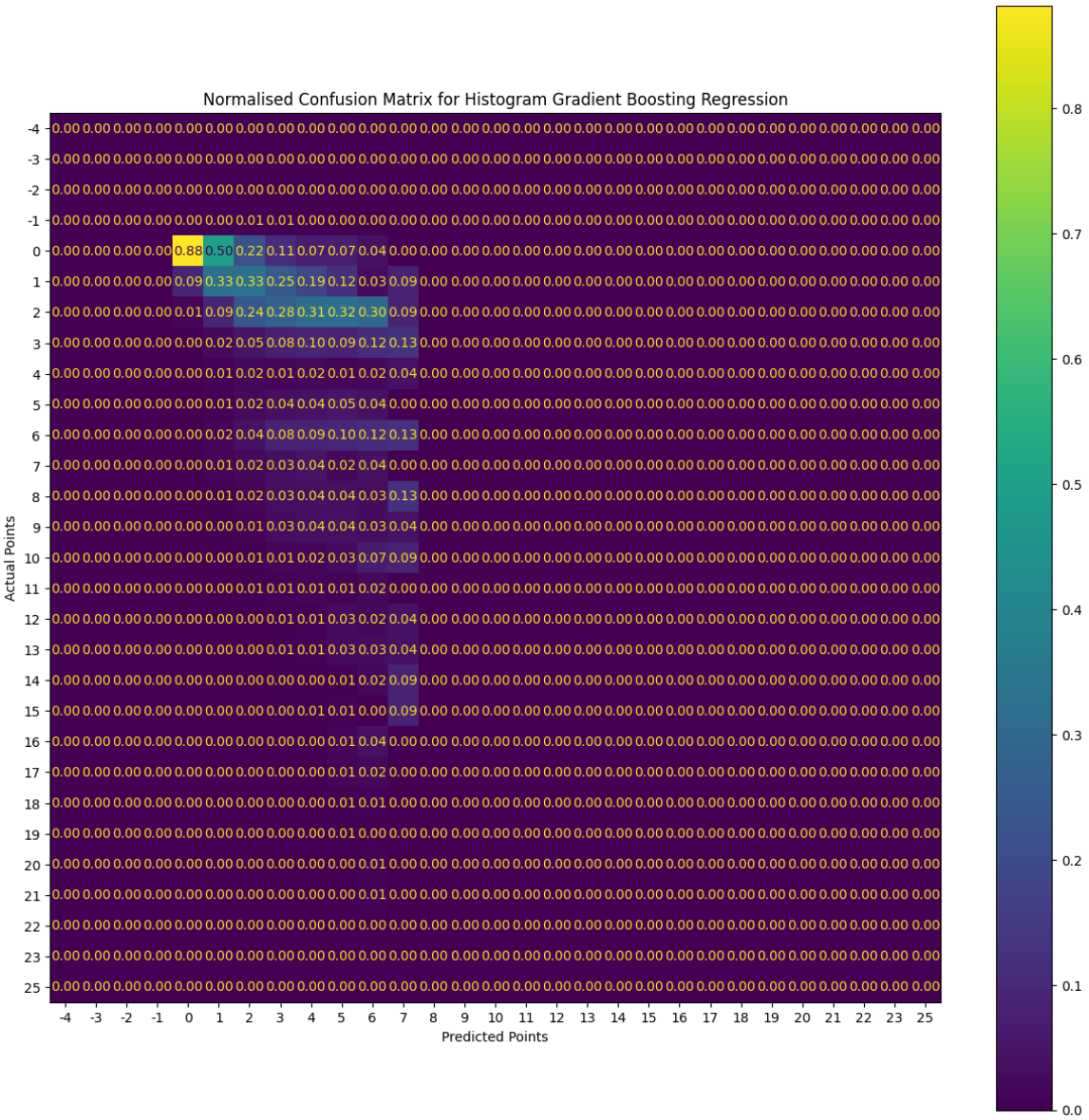

Figure 11: Normalised confusion matrix for test predictions made by the HGBM model

The confusion matrix shows that model does well at predicting player's who score zero points; 88% of players who were predicted to score zero points actually scored zero. A player usually scores zero points if they do not play at all during their match. However, as we move down the lead diagonal of the matrix, the proportion of correct predictions decreases significantly. We can also see that the model never predicts a player to score greater than seven points but 4.82% of players in the test dataset actually scored greater than seven points in a game that they played in.

6.1 Addressing the class imbalance

As a result of the class imbalance in the training (and test) datasets, two strategies were explored to try and reduce the class imbalance in the training dataset with the aim of improving the predictions made by the models on the test dataset.

6.1.1 Random Undersampling (RUS)

The first strategy explored was random undersampling. This works by removing random feature vectors from the over-represented classes in our dataset. Figure 10 showed that the number of players that score zero, one or two points is significantly higher than all other points classes. Therefore we remove a random fraction of players from each of these points classes in our training data. To find the best under-sampling rate, each class was undersampled by 10, 20, 30, 40 and 50 percent and then a HGBM model was trained on the undersampled training dataset and tested on our test dataset. Undersampling rates of over 50 percent were not tested since it is still important that the over-represented points classes have a higher frequency than all other points classes, else the distribution of points classes in the training dataset would be significantly different to the test dataset and the trained model would not be a good fit to the test data. The best combination of sampling rates was the model instance which had the lowest MSE between predicted and actual FPL points on the test dataset. Random under-sampling was done using the `sample()` method from the `random` Python library.

The best combination of random under-sampling rates found were 10, 20 and 10 percent for the players that scored zero, one and two FPL points respectively. The new distribution of FPL points scored in the training dataset is shown in Figure 12.

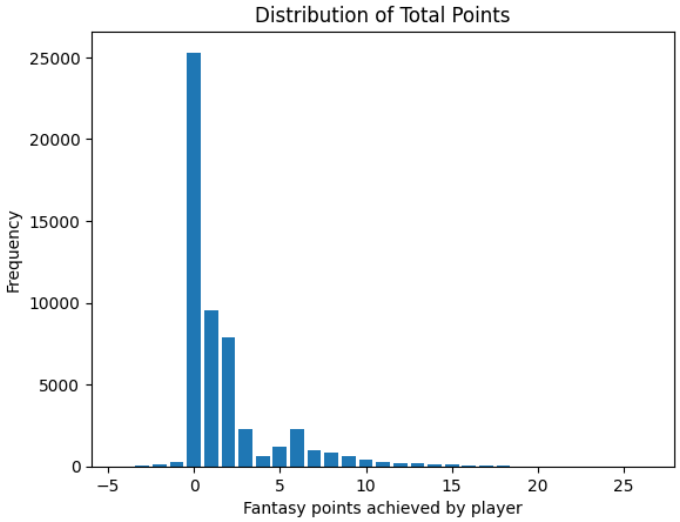

Figure 12: Distribution of FPL points scored in the training dataset after applying RUS

Each of the HGBM, RF and LR models were trained on the undersampled training dataset and tested on the test dataset. Table 15 shows the test performance of each of the models. The RMSE and MAE increased for each model and the R^2 score decreased, compared to the models trained on the original training dataset (Table 14). However, this is expected as we have changed the distribution of the training dataset so it is now less representative of the test dataset.

Model Type	Test RMSE	Test R2 Score	Test MAE
HGBM	**2.257**	**0.253**	**1.403**
LR	2.295	0.228	1.440
RF	2.287	0.233	1.462

Table 15: Test Error and R2 Score for each model trained on the RUS Dataset

To better compare how the models trained on the undersampled training dataset compare to the

models trained on the original training dataset, we define a new error metric named "Threshold RMSE". Threshold RMSE (TRMSE) is defined in Equation 5.

$$\text{TRMSE}(y, \hat{y}, \alpha) = \sqrt{\frac{1}{\sum_{i=1}^{n} w(y_i, \hat{y}_i, \alpha)} \cdot \sum_{i=1}^{n} w(y_i, \hat{y}_i, \alpha) \cdot (y_i - \hat{y}_i)^2} \tag{5}$$

where :

$$w(y_i, \hat{y}_i, \alpha) = \begin{cases} 1 & \text{if } y_i > \alpha \text{ or } \hat{y}_i > \alpha \\ 0 & \text{otherwise} \end{cases}$$

y_i = Label for feature vector i

$\hat{y}_i$ = Predicted feature vector label for feature vector i

n = Number of feature vectors

α = Threshold parameter

The TRMSE metric helps to understand how the accuracy of the model's predictions are for the players which score, or are predicted to score, in the higher, under-represented points classes (greater than three points). In general, we prefer the models to have improved accuracy on these players since these are the players which will be considered when we build the optimal team for a certain gameweek, not players who are predicted to score a low number of points. Table 16 compares the TRMSE for the test data predictions made by each model type when trained on the original training data, and when trained on the undersampled training data. We can see that for all models, the TRMSE decreases when the models are trained on the RUS training data. The RF model has the lowest TRMSE, despite the HBGM model having the lowest RMSE, which shows that the HGBM model is more accurate for overall points predictions but the RF model is more accurate for higher-scoring players.

Training Data	**HGBM**	**LR**	**RF**
Original Training Data	4.267	4.425	4.105
Random undersampled Training Data	4.121	4.213	3.967

Table 16: The TRMSE ($\alpha = 3$) of the test data predictions for each of the models trained on the original training data and the RUS training data

6.1.2 Synthetic Minority Over-Sampling with Gaussian Noise (SMOGN)

The second strategy explored was using an algorithm named Synthetic Minority Over-Sampling with Gaussian Noise, known as SMOGN to rebalance the training dataset. This algorithm was created by Branco et al [33] to address the problem of imbalanced datasets within regression tasks. In particular, it was found that the SMOGN algorithm had the greatest improvement to model performance when used with random forest or MARS (Multivariate Adaptive Regression Splines) models. The SMOGN algorithm uses a combination of random undersampling of over-represented classes and the creation of synthetic data for the under-represented classes. The synthetic data is created using the of interpolation of the data belonging to the under-represented classes and the addition of Gaussian noise.

In order to apply the SMOGN algorithm to our training dataset, the `smogn` Python library [34] was utilised. This library provides an implementation of the SMOGN algorithm which allows the dataset to be rebalanced be input as a pandas dataframe. The `smogn.smoter` function was used to run the SMOGN algorithm on our training dataset. Before the training dataset was passed to the function, all feature vectors representing players that scored a negative number of points were removed from the dataset. This is because we were not interested in over or undersampling these players, since in general, players will very rarely score a negative number of points and we don't want our models to predict negative numbers of points very often. To determine which classes of points to over and under sample, the function requires a parameter `rel_method`. This parameter specifies whether these values are to be automatically determined or manually specified by the user. The parameter that

was passed in specified that the feature vectors representing players who scored between zero and three points (inclusive) should be undersampled and all other feature vectors should be oversampled.

Figure 13 shows the distribution of FPL points in the training dataset after applying the SMOGN algorithm. Compared to the distribution in Figure 10, the shape of the distribution for players that scored above three points is very similar to the shape of a normal distribution, centred around players which scored six points. The number of players that scored between zero and three points has decreased as expected, but they still make up a large proportion of the dataset since we don't want to harm the models' ability to accurately predict players that score in this range of points.

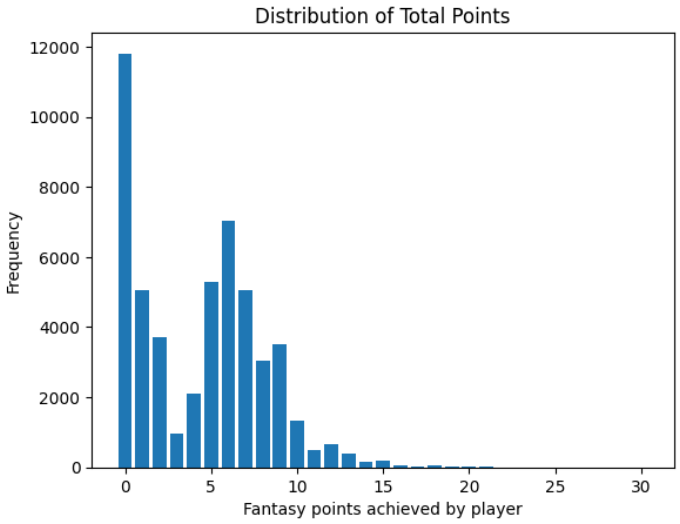

Figure 13: Distribution of FPL points scored after applying SMOGN

Each of the HGBM, LR and RF models were trained on the SMOGN training dataset and tested on our test dataset. Table 17 shows the test performance of each model. Again, the error of each model trained on the SMOGN training data has increased compared to the error of each of the models trained on the original training data, shown in Table 14. The R^2 score of each model has decreased too but this is expected since the distribution of player points scored in the SMOGN training dataset is very different to the distribution of player points scored in the test dataset. Figure 14 shows the normalised confusion matrix for the predictions made by the HGBM model trained on the SMOGN dataset. We can see that compared to the confusion matrix in Figure 11, the points predictions made by the model are higher on average and that the model is now slightly more accurate at correctly predicting players which score over 6 points.

Model Type	**Test RMSE**	**Test R2 Score**	**Test MAE**
HGBM	**2.547**	**0.049**	**1.866**
LR	3.045	-0.359	2.473
RF	2.699	-0.068	2.039

Table 17: Test Error and R2 Score for each model trained on the SMOGN Dataset

Furthermore, we can use the TRMSE metric (Equation 5) to calculate the accuracy of the points predictions for players who scored, or were predicted to score over three points. Table 18 compares the TRMSE for the test data predictions made by the three models trained on the original training data, random undersampled training data, and the training data after applying the SMOGN algorithm. We can see that the each model trained on the SMOGN data has the greatest reduction in TRMSE, and again the RF model has the lowest overall error.

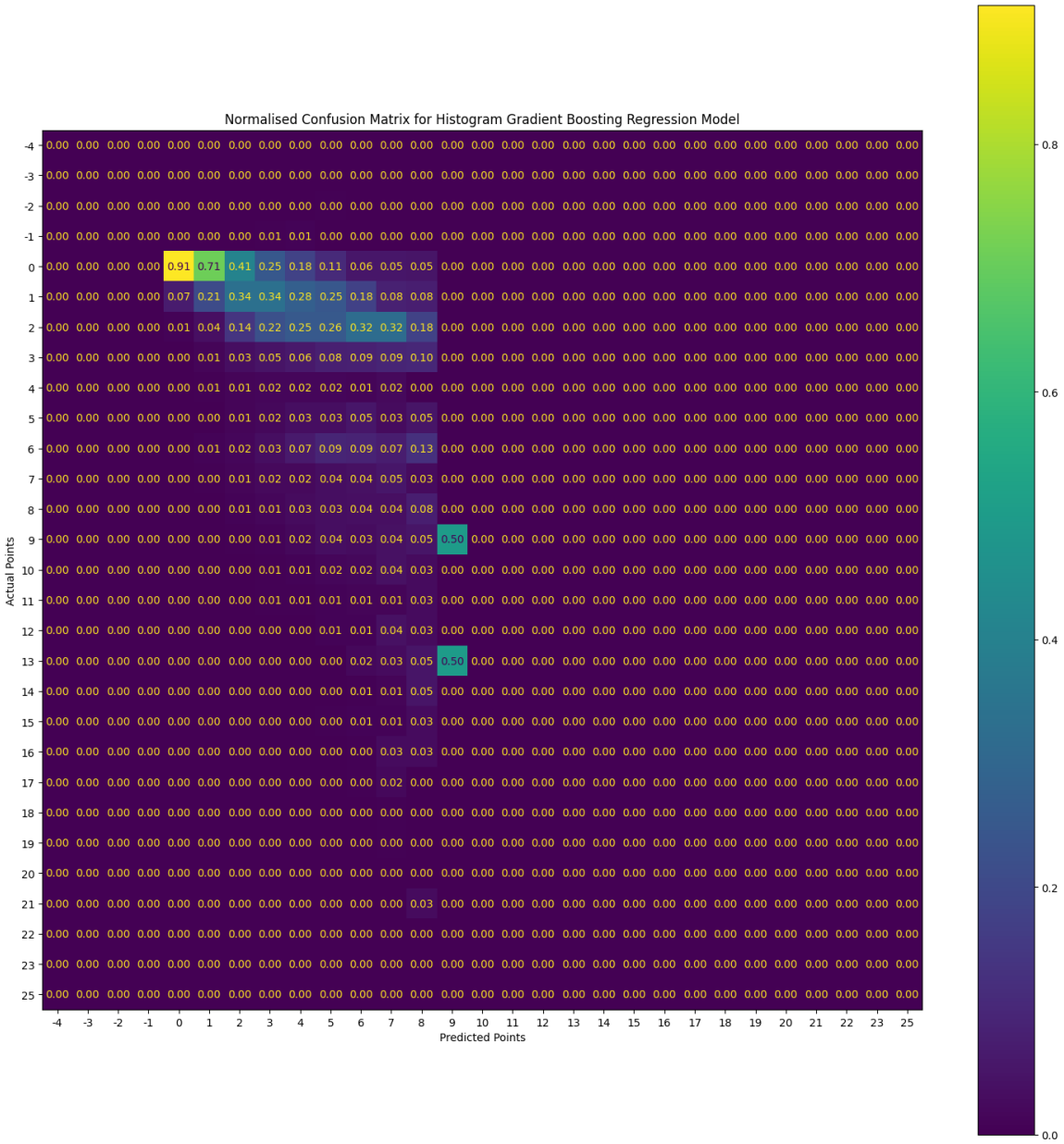

Figure 14: Normalised confusion matrix for test predictions made by the HGBM model trained on SMOGN data

Training Data	**HGBM**	**LR**	**RF**
Original Training Data	4.267	4.425	4.105
undersampled Training Data	4.121	4.213	3.967
SMOGN Training Data	3.718	3.853	3.650

Table 18: The TRMSE ($\alpha = 3$) of the test data predictions for each of the models trained on the original training data and the RUS training data

6.1.3 Summary

In this section, we explored the use of random under-sampling of the training data and applying SMOGN algorithm to the training data in order to reduce the class imbalance present in the training data. As expected, the RMSE, R^2 and MAE metrics for the predictions made on the test data were worse for the models trained on RUS and SMOGN training datasets, since the distribution of the training data was changed, so the models were a weaker overall fit to the test data. However, both strategies produced a reduction in TRMSE for the predictions made on the test data, with the

models trained on the SMOGN training data producing the greatest decrease. This illustrates that models trained on the RUS and SMOGN datasets are better at predicting points for higher scoring players than the models trained on the original training datasets.

7 New Models

In this section we explore a couple of new approaches to predicting players FPL points and compare them to the models we have discussed so far, with the aim of improving the accuracy of the model's predicted points further.

7.1 Position Models

In all training and test datasets explored so far we have the same set of features for all players. However, due to how players in different positions earn points in FPL, there are subsets of features which are naturally more relevant for each player position. For example, players which are forwards in FPL do not earn points when their team keeps a clean sheet in their match. Midfielders earn one point for keeping a clean sheet and defenders and goalkeepers both earn four points. Therefore a team's recent defensive performance is more important to consider when predicting player points for defenders and goalkeepers, rather than midfielders and forwards. It is possible that there is noise is being generated by less important features for each player position, and that this is affecting the accuracy of the current models' predictions.

To test this hypothesis, four separate models were created to predict FPL points for goalkeepers, defenders, midfielders and forwards separately. The original training and test datasets were partitioned into four training and test datasets, where each dataset contains players from each of the four player positions. These new training and test datasets are referenced as as the training and test "position datasets" for the remainder of the report. Recursive feature elimination was the feature selection technique used to select the best subset of features for each of the training and test position datasets. The `sklearn.feature_selection.RFECV` function was used to implement this. The `RFECV()` function uses recursive feature elimination with repeated cross-validation to select the subset of features with the best cross validation score. Repeated cross validation was used with a fold size of five and five as the number of repeats. Recursive feature elimination works by first training a model on the entire set of features and then assigning a weight to each feature, which corresponds to how important the feature is. An instance of the `sklearn.ensemble.GradientBoostingRegressor` class was used as the evaluation model during RFECV since the `sklearn.ensemble.HistGradientBoostingRegressor` class does not provide a class attribute which stores the weighting of the features. After the initial feature weightings have been calculated, the least important features are recursively removed and repeated cross validation is used to evaluate the models performance on each new set of features. The best set of features is returned as the set of features with the best cross validation score averaged over all folds and repeats. The RMSE between predicted and actual player points was used as the metric for cross validation score.

Table 19 shows the new dimensions of the feature vectors in each of the position datasets after applying recursive feature elimination.

Dataset	Feature Vector Dimension
Goalkeeper Dataset	52
Defender Dataset	31
Midfielder Dataset	57
Forward Dataset	44

Table 19: Dimensions of the feature vectors in each position dataset

After the feature selection process is complete, we then perform hyperparameter tuning on an instance of a HGBM and RF model for each of the training position datasets. No hyperparameter tuning is required for the LR model. Grid search cross validation was used again as the method for hyperparameter tuning, with the `sklearn.model_selection.GridSearchCV` class used to implement this.

7.1.1 Results

An instance of each of the tuned HGBM, RF, and LR models were then trained and tested on each of the four position training and test datasets. Table 20 shows the test performance of each model for the points predictions made for each player position, and the combined predictions for all player

positions. The best performing models for each position and metric are highlighted in bold. On average, points predictions for defenders had the lowest error and forwards have the highest error. We can see that the HGBM model had the lowest RMSE and MAE, and highest R^2 score, for all the positions' predicted points apart from the defenders' points predictions, where the LR model had lower RMSE and a higher R^2 score. The RF model performs best for the TRMSE metric for all positions.

		Position				
Metric	**Model Type**	GK	DEF	MID	FWD	ALL
RMSE	HGBM	**2.188**	2.135	**2.265**	**2.725**	**2.269**
	LR	2.197	**2.120**	2.304	2.730	2.284
	RF	2.189	2.147	2.283	2.761	2.286
R^2	HGBM	**0.258**	0.177	**0.263**	**0.273**	**0.245**
	LR	0.252	**0.189**	0.237	0.271	0.235
	RF	0.257	0.168	0.251	0.254	0.234
MAE	HGBM	**1.403**	**1.337**	**1.328**	**1.705**	**1.377**
	LR	1.426	1.373	1.355	1.731	1.407
	RF	1.406	1.400	1.362	1.780	1.423
TRMSE	HGBM	3.511	4.197	4.389	4.627	4.280
	LR	3.449	4.424	4.544	4.512	4.390
	RF	**3.417**	**3.984**	**4.301**	**4.383**	**4.132**

Table 20: Test set performance metrics for the predicted player points for each position and model type

Table 21 compares the HGBM model's test metrics for the combined position predictions (final column of Table 20) with the test metrics for the original HGBM model from Section 5.4, which was trained on the original training dataset from Section 4.1.5, and the RF model trained on the SMOGN training dataset. There is a very small increase in MSE and MAE and decrease in R^2 score for the HGBM Combined Position Model compared to the original HGBM model. The TRMSE has also increased, compared to the both of the other models so we can see that predicting each position individually does not help to deal with the class imbalance of the datasets. This tells us that our original models were in fact robust at predicting points for the different player positions, and that having a single feature representing the position of each player in the original datasets is sufficient.

	Model		
Metric	HGBM Combined Position Model	HGBM Original Model	RF SMOGN Model
RMSE	2.269	2.254	2.699
R^2	0.245	0.255	-0.068
MAE	1.377	1.369	2.039
TRMSE	4.280	4.267	3.650

Table 21: Comparison of the test performance of the HGBM model trained on the original training data, the combined predictions of the HGBM position models and the RF model trained on the SMOGN training data

7.2 Points Action Models

In the Literature Review Section we explored the thesis written by Valouxis [8]. The problem of predicting player FPL points was broken down into several sub problems, where each sub-problem aimed to predict the value of a player action e.g. number of goals scored, or team event e.g. number of goals conceded by the player's team, which could happen in a match that would cause a player's points total to increase or decrease. The sub-problems were defined in the Literature Review section. The number of predicted points gained or lost for each player action or team event was calculated by multiplying the predicted value of the action or event by the number of points that would be gained or lost if it happened during a match, as defined in Table 1. Then the total predicted points for the player can be calculated as the sum of all predicted points gains or losses, for each player action or team event. In this section we will adopt a similar approach and compare its performance to the results obtained by the models in the previous sections.

In Table 1, the actions which most frequently earn players points in matches are scoring a goal, assisting a goal, keeping a clean sheet (for all positions except forwards), making saves (only for goalkeepers), playing at least one minute in the match and earning bonus points based on the player's match performance. Note that if a goalkeeper saves a penalty they also gain five points, but this is a very rare action in a football match so this was not considered as an action which was required to be predicted. The most common action which causes a player to lose points is losing a point for every two goals conceded by the player's team while they are playing in the match (recall that this only applies to goalkeepers and defenders). The other actions which cause a player to lose points are receiving a yellow card, receiving a red card, missing a penalty, and scoring an own goal. Yellow cards and red cards are highly random events during football matches so we will use a player's average number of yellow and red cards during the current season as a indicator for the expected number of yellow and red cards they will receive in a match. A player missing a penalty or scoring an own goal are also very rare events in football matches so these events were not considered as necessary to predict either.

Seven seperate models were identified which were trained in order to predict the following player actions or match events :

1. **Goals model** : predict how many goals a player will score in the match.
2. **Assists model** : predict how many assists a player will make in the match.
3. **Clean sheet model** : predict whether the player will keep a clean sheet in the match.
4. **Minutes played model** : predict how many minutes the player will play in the match.
5. **Bonus points earned model** : predict how many bonus points a player will earned at the end of the match
6. **Goals conceded model** : predict how many goals the player's team will concede.
7. **Saves made model** : predict how many saves the player will make in the match (for goalkeepers only).

These seven models are refereed to collectively as the "Points Action Models" for the remainder of the report. For each model, a new training and test dataset was required. Recursive Feature Elimination was used to find the best subset of features from the full set of features which were in the original training and test datasets, that were created in Section 4.1.5. For each dataset, the label is the player action or event that we want to predict e.g. The label for the goals dataset is the number of goals scored by the player in the match. Table 22 shows the dimension of the feature vectors in each dataset.

Dataset	Feature Vector Dimension
Goals Dataset	29
Assists Dataset	47
Clean Sheet Dataset	45
Minutes Played Dataset	39
Bonus Points Dataset	61
Saves Made Dataset	28
Goals Conceded Dataset	21

Table 22: Dimension of the feature vectors in the training and test datasets for each points action model

After the feature selection process, hyperparameter tuning was then performed on an instance of a HGBM and RF model for predicting each of the seven events. Again, grid search cross validation was used as the method for hyperparameter tuning.

Section 7.2.1 shows how the predicted values from each of the seven points action models are combined to give the predicted FPL points for a given player. For ease of notation, each of the models listed on the previous page is labelled as their corresponding number in the list.

7.2.1 Predicted Points Derivation

Suppose we want to calculate the number of points player i is predicted to score for a match in any gameweek.

Let x_j denote the vector corresponding to the output from model $j \quad \forall j \in [1, 7]$.

Let $g^{(s)} = x_{1i}$ denote the number of goals player i is predicted to score in the match.

Let $a = x_{2i}$ denote the number of assists player i is predicted to make in the match.

Let $c = x_{3i}$ denote the probability of player i keeping a clean sheet in the match.

Let $m = x_{4i}$ denote the number of minutes player i is predicted to play in the match.

Let $b = x_{5i}$ denote the number of bonus points player i is predicted to score at the end of the match.

Let $g^{(c)} = x_{6i}$ denote the number of goals that player i is predicted to concede in the match.

Let $s = x_{7i}$ denote the number of saves that player i is predicted to make.

Then construct a vector $y_i \in \mathbb{R}^7$ such $y_i = (gs, a, c, m, b, gc, s)^T$.

Let $p_i \in \{1, 2, 3, 4\}$ denote player i's position, where $\{1, 2, 3, 4\} = \{\text{Goalkeeper, Defender, Midfielder, Forward}\}$.

Let c_i be the average number of yellow cards player i has received for every 90 minutes played in the current season.

Let r_i be the average number of red cards player i has received for every 90 minutes played in the current season.

Then we can define a function $p : \mathbb{R}^7 \times \{1, 2, 3, 4\} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ which calculates the predicted points for player i as:

$$p(y_i, p_i, c_i, r_i) = \begin{cases} 10y_{i1} + 3y_{i2} + 4y_{i3} + p^{(m)}(y_{i4}) + y_{i5} - p^{(gc)}(y_{i4}, y_{i6}) + \lfloor (y_{i7}/3) \rfloor - c_i - 3r_i & p_i = 1 \\ 6y_{i1} + 3y_{i2} + 4y_{i3} + p^{(m)}(y_{i4}) + y_{i5} - p^{(gc)}(y_{i4}, y_{i6}) - c_i - 3r_i & p_i = 2 \\ 4y_{i1} + 3y_{i2} + y_{i3} + p^{(m)}(y_{i4}) + y_{i5} - c_i - 3r_i & p_i = 3 \\ 3y_{i1} + 3y_{i2} + p^{(m)}(y_{i4}) + y_{i5} - c_i - 3r_i & p_i = 4 \end{cases}$$

Where $p^{(m)} : \mathbb{R} \to \{0, 1, 2\}$ returns the number of points earnt based on the number of minutes player i played and $p^{(gc)} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ returns the number of points lost based on the number of goals conceded by player i's team. Functions $p^{(m)}$ and $p^{(gc)}$ are defined as :

$$p^{(m)}(x) = \begin{cases} 0 & x = 0 \\ 1 & 0 < x < 60 \\ 2 & x \geq 60 \end{cases}$$

$$p^{(gc)}(n, m) = \begin{cases} \lfloor (n/2) \rfloor & m \geq 80 \\ \frac{3}{4}\lfloor (n/2) \rfloor & 60 \leq m < 80 \\ \frac{1}{2}\lfloor (n/2) \rfloor & 10 \leq m < 60 \\ 0 & 0 \leq m < 10 \end{cases}$$

The output of the $p^{(gc)}$ function is dependent on the number of goals that the player's team is predicted to concede and the number of minutes the player is predicted to play. In FPL, any player classed as a defender and goalkeeper lose a point for every two goals their team concedes while they are on the pitch. This means that if a player is on the bench while their team concedes two goals, they will not lose any points. Therefore the number of points which are lost due to the predicted number of team goals conceded are weighted by the number of minutes the player is predicted to play. If the player is predicted to play at least 80 minutes of the match, then it is likely that they will be on the pitch for the all of the goals conceded by their team so they lose the full amount of points. However, if the player is predicted to play less than 80 minutes then there is a higher chance that they will not be on the pitch when goals are conceded by their team, so the number of points that they lose is reduced accordingly.

7.2.2 Results

An instance of each tuned HGBM, RF and LR models were then trained and tested on each of the seven points action training and test datasets. Table 23 and Table 24 show the performance metrics for each model evaluated on each of the model's test datasets. The best model type for predicting each of the labels, and for each metric is highlighted in bold.

		Predicted Label			
Metric	**Model**	Goals	Assists	Clean Sheet	Minutes Played
RMSE	HGBM	**0.240**	**0.223**	0.305	**25.9**
	LR	0.243	0.235	**0.293**	27.4
	RF	0.242	0.233	0.298	26.3
R^2	HGBM	**0.122**	**0.064**	**0**.015	**0.584**
	LR	0.100	0.044	**0.090**	0.535
	RF	0.111	0.053	0.062	0.573
MAE	HGBM	**0.097**	**0.093**	**0.175**	**18.3**
	LR	0.106	0.097	0.182	19.7
	RF	0.100	0.096	**0.175**	18.8

Table 23: Test dataset performance for using each model type to predict each of the labels

		Predicted Label			
Metric	**Model**	Bonus Points	Saves	Team Goals Conceded	Predicted Points
RMSE	HGBM	**0.503**	**0.480**	1.194	3.004
	LR	0.505	0.516	1.194	3.134
	RF	0.505	**0.480**	**1.184**	**2.999**
R^2	HGBM	**0.080**	**0.551**	0.079	**-0.281**
	LR	0.071	0.483	0.079	-0.354
	RF	0.072	**0.551**	**0.093**	-0.284
MAE	HGBM	**0.216**	0.096	0.944	1.718
	LR	0.226	0.128	0.948	1.816
	RF	0.224	**0.092**	**0.939**	**1.717**

Table 24: Test dataset performance for using each model type to predict each of the labels

We can see that the HGBM model is the best type of model for the predicting goals scored, assists made, minutes played and bonus points scored. The LR model is best for predicting the chance of keeping a clean sheet since it performs best for two out the three metrics. The RF model is best for predicting the number of saves made by goalkeepers and the number of goals conceded by the player's team. Now we can use the predicted values from each of the identified best models and input them into our predicted points function p, from Section 7.2.1, to get the predicted FPL points for each player. The final column of Table 24 shows the test metrics for these predicted points. The RF model was found to give the lowest MSE and MAE for the points predictions. Table 25 shows the TRMSE $\alpha = 3$ values for the points predictions generated by each model. The RF model has the lowest TRMSE.

Table 26 compares the test performance metrics for the predicted points generated by the points action models, position models (Section 7.1) and the original HGBM model (Section 5.3) and the RF model trained on SMOGN data (Section 6.2). For the points action models, the best models that were identified for each predicted factor in Table 23 & 24 were used to calculate the player predicted points. Unfortunately, the predicted points calculated from the predictions of the points action models have the highest RMSE and lowest R^2 score. The R^2 score is negative and significantly lower than the other models which means that predicting the average of the player points labels is a more accurate than using the predicted points. The fact that the TRMSE is the second lowest, despite the RMSE being the highest, suggests that the points action models are quite effective at predicting higher scoring player points but struggle with predicting the lower-scoring players points. This could be a reason why the R^2 score is significantly worse for the player predicted points calculated from the points action models.

Model		
HGBM	LR	RF
4.059	4.551	3.954

Table 25: The TRMSE $\alpha = 3$ for the player points predictions generated by the points action models

	Model			
Metric	Combined Points Action Models	Combined Position Models	HGBM Original Model	RF SMOGN Model
RMSE	2.987	2.269	**2.254**	2.699
R^2	-0.609	0.245	**0.255**	-0.068
MAE	1.714	1.377	**1.369**	2.039
TRMSE	3.905	4.280	4.267	**3.650**

Table 26: Comparison of the test error metrics and R^2 score for the player points predictions generated by the points action models, position models, original HGBM model and RF SMOGN model

8 Predicting the Optimal Team

Now that we have explored several methods of predicting individual player FPL points, we can expand this to choosing the "optimal" team of players for the set of matches in an upcoming gameweek. However, as detailed in Section 1.2, there are a number of restrictions that must be followed when picking an FPL team. This means we can not simply just select the predicted highest scoring fifteen players to make up our team. The rules for choosing a team are summarised below :

- The chosen team must consist of 15 players which includes, two goalkeepers, five defenders, five midfielders and three forwards.
- The starting eleven players must consist of one goalkeeper, between three and five defenders, between two and five midfielders and between one and three forwards. The remaining four players make up the team's bench. Only points earnt by players in the starting eleven count towards the team's points total, unless a player does not play at all in their match, in which case they are swapped with a player from the bench and that player's points are added to the team's total.
- The total cost of all players in the team must be within an allocated budget. At the start of FPL, the budget is £100 million for all managers, but this can increase and decrease throughout the season due to player's prices rising and falling.
- A maximum of three players from each PL team can be chosen in the team.

8.1 Linear Program Construction

To calculate the optimal team for a given gameweek, linear programming was used. First, the variables for problem are defined. Let $X = \{1, 2, \cdots, n\}$ denote the set of players we can include in our team for that gameweek, where each player is represented by a unique positive integer. Then $\forall i \in [1, n]$, let x_i be a binary variable denoting if player i is chosen in the team, y_i be a binary variable denoting if player i is chosen to be in the starting eleven, let c_i be a binary variable denoting if player i is to be chosen as the captain of the team and v_i be a binary variable denoting if player i is to be chosen as vice-captain. Let $B \in \mathbb{R}$ denote the budget that is available to build the team.

The following functions are defined which are used within the constraints of the linear program :

Let $p : X \to \mathbb{R}$ be a function which maps a player to the number of points which they are predicted to score.

Let $c : X \to \mathbb{R}$ be a function which maps a player to the their cost in the given gameweek. Each position is represented by an integer from one to four, as has been done previously in the report.

Let $s : X \to \{1, 2, 3, 4\}$ be a function which maps a player to their FPL position.

Let $l : X \to \{1, 2 \cdots, 20\}$ be a function which maps a player to the club they play for. Each PL club is represented by a unique positive integer from one to twenty.

Finally, a parameter $\alpha \in [0, 1]$ is defined which represents the importance of the players chosen which are not in the starting eleven (players on the bench). As mentioned previously, only points scored by players in the starting eleven count towards the overall points total of the team. However, in the event that a player chosen in the starting eleven does not play, they will be swapped with a player on the bench, whose points scored will be added to the team's total instead. Therefore, although we want to concentrate the majority of the available budget into selecting the highest predicted scoring starting eleven players, it can also be useful to have a bench of players who are predicted to score high. If $\alpha = 0$ then full priority is assigned to selecting the best starting eleven players, the cheapest possible group of players form the bench of the team. However, for $\alpha > 0$, a weight of α is assigned to the predicted points of the players on the bench and these weighted points are included as part of objective function. The larger the value of α, the greater the amount of the total budget that is used to select the bench players of the team.

The full linear program was defined as shown below :

Maximise $\sum_{i \in X} p(i) \cdot \big((1-\alpha) \cdot x_i + c_i + 0.01 \cdot v_i + \alpha \cdot y_i\big)$

subject to:

$$\sum_{i \in X} x_i \cdot c(i) \leq T$$

$$\sum_{i \in X} x_i = 15$$

$$\sum_{i \in X} y_i = 11$$

$$\sum_{i \in X} c_i = 1 \quad \text{(only one captain in the team)}$$

$$\sum_{i \in X} v_i = 1 \quad \text{(only one vice-captain in the team)}$$

$$\sum_{i \in X : s(i) = 1} x_i = 2$$

$$\sum_{i \in X : s(i) = 2} x_i = 5$$

$$\sum_{i \in X : s(i) = 3} x_i = 5$$

$$\sum_{i \in X : s(i) = 4} x_i = 3$$

$$\sum_{i \in X : l(i) = j} x_i \leq 3 \quad \forall j \in [1, 20]$$

$$\sum_{i \in X : s(i) = 1} y_i = 1$$

$$3 \leq \sum_{i \in X : s(i) = 2} y_i \leq 5$$

$$2 \leq \sum_{i \in X : s(i) = 3} y_i \leq 5$$

$$1 \leq \sum_{i \in X : s(i) = 4} y_i \leq 3$$

$$x_i \geq y_i \ \forall i \in [1, n]$$

$$y_i \geq c_i \ \forall i \in [1, n]$$

$$y_i \geq v_i \ \forall i \in [1, n]$$

$$c_i + v_i \leq 1.5 \ \forall i \in [1, n] \quad \text{(vice captain and captain must be different players)}$$

$$x_i, y_i, z_i \in \{0, 1\} \ \forall i \in [1, n]$$

Note that the coefficient of v_i in the objective function can be any positive real number less than one, as it just needs to ensure that the player in the team with the second highest predicted points is chosen as vice-captain.

In order to solve the linear program, the PuLP Python library [35] was used. The `LpProblem` function was used to define a maximisation linear programming problem. The variables were initialised using the `LpVariable` class and the constraints were initialised using the `LpSum` function. Once the problem had been defined, the `solve()` function was called, which automatically chooses an appropriate solving algorithm to solve the linear program. The $\{i : x_i = 1\}$ were then be used to construct the chosen team of players, and the $\{i : y_i = 1\}$ defined the starting eleven, with the rest of the chosen players forming the bench. Finally, the predicted points of the team was calculated as the sum of the predicted points of the players chosen in the starting eleven.

A function named `optimal_team()` was created which takes in three parameters : the gameweek to output the optimal team for, a dictionary containing each player fpl ID mapped to the number of points that they are predicted to score, and the budget available to build the team. This function uses the above linear programming formulation to calculate the optimal team, based on the parameters provided, and returns the players which form the optimal team as well as the number of points each player is predicted to score, and the total team's predicted score.

8.2 Optimal Team Predictions for Gameweeks 1-30 of the 2024/25 Premier League Season

Now that we have multiple models for predicting individual player points, and we can use linear programming to find the optimal team for a given gameweek, we investigate the optimal teams that are selected each in each gameweek of the 2024/25 season (up until gameweek 30). We also compare the number of points that each team was predicted to score against the number of points that each team actually would have scored.

It is important to reiterate that in FPL you cannot simply just choose an entirely new team of players every week which you think will score the highest, without incurring significant points penalties to your overall score for that week (See Section 1.2). The only times where you can build a new team are at the beginning of the game (prior to gameweek one of the season) and when you choose to play your wildcard or free hit chip. These chips were defined in Section 1.2. However, having a tool to predict the optimal team for each gameweek is useful for highlighting which player's are likely the best to transfer into an FPL team, as well as helping to build the best team when playing the wildcard or free hit chip.

Comparing the number of points that a team was predicted to score versus the number of points it actually scored for each gameweek is a more practical way of comparing the performances of the models that we have explored so far, in Sections 5, 6 and 7. So far we have only compared the MSE, MAE, R^2 and TRMSE metrics between the predicted and actual player points for each of the models. Although these metrics are useful to analyse the accuracy of each model's individual player points' predictions, when we are trying to calculate the optimal team we only want to include players that are predicted to score the highest, and we are less interested in the lower predicted scoring players. Therefore, by comparing the overall team points predictions, using the individual player predictions generated by each model, we can get a better idea of how accurate each model is at predicting the higher scoring players.

The four models that we will compare are listed below. These are the best models that we have explored in the most detail so far in the report.

1. The HGBM model trained on the original training dataset, from Section 5.3.
2. The RF model trained on the SMOGN training dataset, from Section 6.1.2.
3. The ensemble of position models, from Section 7.1
4. The ensemble of points action models, from Section 7.2

Then for gameweeks 1 to 30 of the 2024/25 PL season, each of the four models were used to predict player points for each gameweek, and then input these predicted points to the `optimal_team()` function, along with the gameweek number and the budget for that week. As mentioned in Section 1, the budget to build an FPL team starts at £100 million, for gameweek one of the season, and can increase or decrease by multiples of £0.1 million throughout the season. Since we are building a new team each gameweek, we have to estimate the available budget for each gameweek. The budget for week i is defined by Equation 6. This equation was based on the average weekly increase or decrease in team value of my real life FPL team. For each gameweek, each model was ran five times and the resulting predicted and actual team scores were averaged over the five runs. This was to account for the small amount of randomness within the HGBM and RF models, which caused team predictions to differ slightly between runs of the same model type on the same gameweek.

$$\text{Budget}(i) = \begin{cases} \text{Budget}(i-1) + X_i & i > 0 \\ 100 & i = 0 \end{cases} \tag{6}$$

where X_i is a random variable such :

$X_i \sim \text{Categorical}(\{-0.3, -0.2, -0.1, 0, 0.1, 0.2, 0.3, 0.4, 0.5\}, \{0.025, 0.025, 0.1, 0.1, 0.25, 0.25, 0.15, 0.1, 0.05\})$

To measure the accuracy of the optimal team predictions made by each model we introduce a new metric, one-sided team error (OSTE). OSTE is defined in Equation 7. The idea behind OSTE is that if the number of points that actually would have been scored by the predicted optimal team is actually greater than the predicted score, then there should be no error, but if predicted score was higher than the actual score then this should be penalised.

$$\text{OSTE}(s_i, \hat{s}_i) = \min(0, s_i - \hat{s_i}), \quad \text{where :} \tag{7}$$

s_i = Number of points that the predicted optimal team actually scored in gameweek i

$\hat{s}_i$ = Number of points that the predicted optimal team was predicted to score in gameweek i

In Figure 15, we can see that the optimal teams which were created using the player points predictions from the RF model, trained on the SMOGN dataset, have the highest average OSTE. The optimal teams created using the points predictions from the ensemble of position models have the lowest average OSTE.

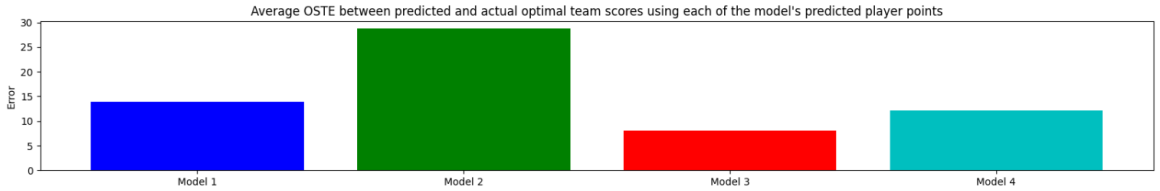

Figure 15: The average OSTE between predicted optimal team scores and actual optimal team scores, averaged over the optimal team predictions for gameweeks 1-30.

Figure 16 shows the number of times that each optimal team, created using each model's player points predictions was predicted to score highest, in each of the 30 gameweeks. In all 30 gameweeks, the the optimal team created using the points predictions from the RF model trained on the SMOGN data had the highest predicted scoring optimal team. However, Figure 17 shows the number of times that the optimal team created using each model's player points predictions actually scored highest, in each of the 30 gameweeks. The optimal teams created using the player points predictions from the HGBM model, trained on the original training dataset, actually score highest the most frequently. This tells us that although the RF model trained on the SMOGN data had the lowest TRMSE on predicted player points, the overall team's predicted points are being consistently overestimated. The HGBM model trained on the original dataset had the lowest RMSE and MAE on predicted player points, which reflects in the fact that the most frequent highest scoring optimal teams use the player points predictions from this model.

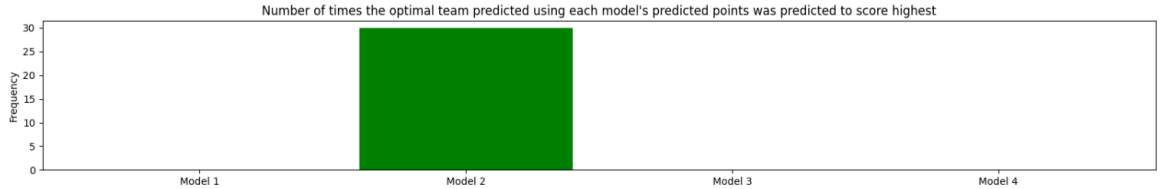

Figure 16: The number of times the optimal team predicted using each model's predicted points was predicted to score highest.

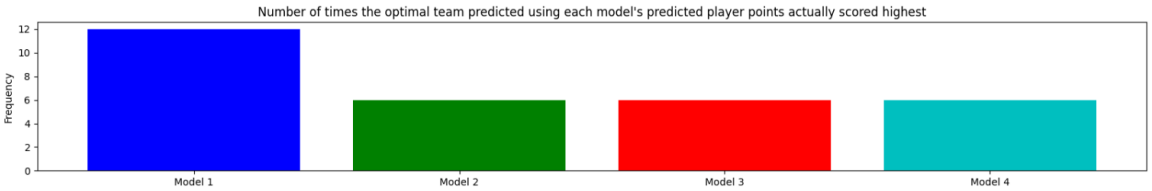

Figure 17: The number of times the optimal team predicted using each model's predicted points actually scored the highest.

8.3 Recommending Transfers

In the earlier Objectives section, it was mentioned that an additional project objective was defined towards the end of term two, which was to create a transfer recommendation algorithm. This algorithm takes in as input a user's team and the number of gameweeks, which we denote the "lookahead period", which should be considered in advance when recommending players to transfer in and out of the user's team, for each gameweek that falls within the lookahead period. The structure of the algorithm is described below in Algorithm 1.

Algorithm 1 Transfer Recommender

1: **Inputs :** Current FPL team and lookahead period
2: **Outputs :** Weekly transfer plan and expected points gain
3: Calculate the predicted player points for each gameweek within the lookahead period.
4: For each gameweek:
5: Determine the optimal transfers in and transfers out to maximise the expected points gain of the team.
6: Return the transfers for each gameweek and the weekly expected points gain.

Since this objective was defined as an extension task, there was only time to get the algorithm working for when the lookahead period was defined as one. This means that transfers can only be recommended for the upcoming gameweek. The advantage of having a larger lookahead period than one gameweek is that transfers can be recommended based on the long-term fixtures of players, rather than only recommending transfers based on the next match that a player is playing in. For example, a player may have an easy fixture in upcoming gameweek, in which they are likely to earn a high number of FPL points, but in the three gameweeks after they could have difficult fixtures in which they might struggle to score highly in. In this case, if a lookahead of one was used then that player may be recommended to be transfered in for the upcoming gameweek but then transfered out the next. However, if a lookahead of four was used than the algorithm would see that the player might not score highly in three of the next four games so there are better players that could be transfered in.

To determine the optimal transfers in step five of the algorithm, another linear programming problem was defined, similar to in Section 7.1. The linear programming problem is detailed below :

Let `gw` denote the gameweek that we want to generate transfer recommendations for. Let $X = \{1, \cdots n\}$ denote the set of available players. Let $e \in \{0,1\}^n$ be a binary vector representing the current user's starting eleven, where $e_i = 1$ if and only if player i is in the starting eleven. Let $t \in \{0,1\}^n$ be a binary vector representing the user's entire team, where $t_i = 1$ if and only if player i is in the team.

Let $p : X \to \mathbb{R}$ be a function which maps a player to the number of points which they are predicted to score in the upcoming gameweek.

Let $c : X \to \mathbb{R}$ be a function which maps a player to the their current cost.

Let $s : X \to \{1, 2, 3, 4\}$ be a function which maps a player to their position. Each position is denoted by an integer between one and four, as done previously in the report.

Let $l : X \to \{1, 2 \cdots, 20\}$ be a function which maps a player to the club they play for. Each PL club is represented by a unique positive integer from one to twenty.

Then $\forall i \in \{1, n\}$ define the following binary variables :

- $t_i^{(f)}$ denotes if a free transfer is used to transfer in i player to the team for the next gameweek
- $t_i^{(p)}$ denotes if a paid transfer (a transfer which incurrs a -4 points deduction to the team's points total) is used to transfer in player i to the team for the next gameweek
- $t_i^{(o)}$ denotes if player i is transfered out of the team for the next gameweek
- s_i denotes if player i is in the starting eleven in the next gameweek
- c_i denotes if player i is chosen as captain for the next gameweek
- v_i denotes if player i is chosen as vice-captain for the next gameweek

Let u $\in \{0,1\}^n$ be a binary vector denoting the user's team for the upcoming gameweek, after the transfers have been made.

Let $\alpha \in [0,1]$ denote the importance of the bench players (see Section 8.1). Let b denote the budget that the user has available (this is not the total cost of the team but is any spare money not used on players in the team) and let k denote the number of free transfers that the user has for the upcoming gameweek.

The objective function and the set of constraints are defined on the following page.

Maximise $\sum_{i=1}^{n}((\alpha \cdot s_i + c_i + 0.001 \cdot v_i) + (1-\alpha) \cdot (u_i)) \cdot p(i)) - 4 \cdot t_i^{(p)}$

subject to :

$u = \sum_{i=1}^{n} e_i + t_i^{(f)} + t_i^{(p)} - t_i^{(o)}$ (the team for the upcoming gameweek is the team at the end of the previous gameweek, plus any players that have been transfered in or out of the team)

$\sum_{i:s(i)=1}^{n} u_i = 2$ (must have two goalkeepers in the team)

$\sum_{i:s(i)=2}^{n} u_i = 5$ (must have five defenders in the team)

$\sum_{i:s(i)=3}^{n} u_i = 5$ (must have five midfielders in the team)

$\sum_{i:s(i)=4}^{n} u_i = 3$ (must have three forwards in the team)

$\sum_{i:s(i)=1}^{n} s_i = 1$ (must have one goalkeeper in the starting eleven)

$3 \leq \sum_{i:s(i)=2}^{n} s_i \leq 5$ (must have between 3 and 5 defenders in the starting eleven)

$2 \leq \sum_{i:s(i)=3}^{n} s_i \leq 5$ (must have between two and five midfielders in the starting eleven)

$1 \leq \sum_{i:s(i)=4}^{n} s_i \leq 3$ (must have between one and three forwards in the starting eleven)

$\sum_{i:l(i)=j}^{n} l(i) \cdot u_i \leq 3 \ \ \forall j \in [1,20]$ (must have at most three players from each PL team)

$\sum_{i=1}^{n} t_i^{(f)} \leq k$ (can use at most k free transfers)

$\sum_{i=1}^{n} s_i = 11$ (must have 11 players in the starting 11)

$\sum_{i=1}^{n} u_i = 15$ (must have 15 players in the team)

$\sum_{i=1}^{n}((t_i^{(f)} + t_i^{(p)}) \cdot c(i)) - (t_i^{(o)} \cdot c(i)) \leq b$

(the total price of the players transfered in minus the total price of the players transfered out can be at most the budget available)

$u_i \geq s_i \ \forall i \in [1,n]$ (all players in the starting eleven must be in the team)

$u_i \geq c_i \ \forall i \in [1,n]$ (the captain must be in the starting eleven)

$u_i \geq v_i \ \forall i \in [1,n]$ (the vice-captain must be in the starting eleven)

$v_i + c_i \leq 1.5 \ \forall i \in [1,n]$ (the captain must be different from the vice-captain)

$t_i^{(f)} + t_i^{(p)} + t_i^{(o)} \leq 1 \ \ \forall i \in [1,n]$

(a player can't be both transfered in and transfered out of the team before the upcoming gameweek)

Then once the optimal solution to the linear program has been found, the values of the variables can be used to output which players have been transfered in and out of the team, which players have been swapped in and out of the starting eleven, the players selected in the starting eleven, and who to captain and vice-captain in the team. Figure 18 shows an example of the output returned by the transfer recommendation algorithm.

```
---Gameweek 21---
Budget : 0.2
-----Transfers In-----
Antoine Semenyo : 5.6

-----Transfers Out-----
Ismaïla Sarr : 3.38

-----Swapped Out-----

-----Swapped In-----

-----Expected Points Gain-----
2.22

----Captain-----
Mohamed Salah : 13.19

----Vice Captain-----
Antoine Semenyo : 5.6

-----Starting 11 for GW 21 -----
['Mark Flekken']
['Trent Alexander-Arnold', 'Antonee Robinson', 'Gabriel dos Santos Magalhães']
['Mohamed Salah', 'Antoine Semenyo', 'Cole Palmer', 'Gabriel Martinelli Silva', 'Anthony Gordon']
['Alexander Isak', 'Nicolas Jackson']

-----Bench-----
['Hákon Valdimarsson', 'João Pedro Junqueira de Jesus', 'Levi Colwill', 'Jacob Greaves']
```

Figure 18: Example output of the transfer recommendation algorithm

9 Web App

This section focuses on the development of a web application to display the results of the optimal team prediction and transfer recommendation algorithms in a more visually friendly manner. Currently, the output from the optimal team prediction algorithm is just raw text which lists the players in each position of the team which should be chosen, as well as the number of points that each player is predicted to score. Similarly, the transfer recommendation algorithm just lists the players which should be transfered in or out of the team, and the expected points gain from making these moves.

In the Section 2.2 of the Literature Review, we introduced the Fantasy Football Hub [18] website. The web app that has been developed takes inspiration from the AI Transfers and AI Team features of Fantasy Football Hub. It is not designed to provide competition to the tools provided by Fantasy Football Hub, since these are the state of the art in the FPL field, but rather act as a platform to show the main results from the models and algorithms that have been developed throughout this report. In the future, the web application could be developed further to include more features and to improve its overall appearance and functionality.

9.1 Design

The web app has two main features :

1. Optimal Team Predictor - Allows the user to enter the gameweek that they want to generate the predicted optimal team for and the budget that they have to build the team. The optimal predicted team is then displayed, along with the individual predicted points of the players within the team and the total predicted points score of the team.
2. Transfer Recommender - Allows the user to enter their team's unique ID (every manager's team that is registered in FPL has a unique ID) which is used to get their current FPL team. The user also enters the number of gameweeks in advance that they would like to plan transfers for (the lookahead period). Then the transfer recommendation algorithm will output the players which should be transfered in and out of the user's team, for each of the gameweeks in advance that were specified. As mentioned previously, only functionality for a lookahead value of one has been implemented in the project.

9.2 Implementation

The web app was created using the Flask [25] web application framework. Flask was chosen since it is written in Python, so the machine learning models and algorithms that had already been written could be easily be implemented within the back-end of the web app. The front-end was developed using HTML and CSS code due to its familiarity before the project. The Jinja template engine [36] was used to integrate the base HTML code of the web page with the outputs of the team generator or transfer recommendation algorithms, to render the optimal team or recommended transfers on the web page for the user.

9.2.1 Optimal Team Predictor

Figure 19 shows the page that is displayed when the optimal team predictor tool is loaded. On the left hand side, there are two input boxes which allow the user to enter the gameweek that they want to generate the predicted optimal team for, and the budget that they have available to build the team.

Figure 19: A screenshot of the optimal team predictor page before the "generate team" button is pressed

Figure 20 shows the resulting predicted optimal team that corresponds to the inputs made by the user. The starting eleven is shown in the centre of the page, overlaid on the football pitch. On the right hand side, the four players which are selected to form the bench are shown. Each player in the starting eleven is displayed with their photo, their full name, the number of points which they are predicted to score in the specified gameweek and the number of points that they actually scored. Below the starting eleven, the total number of points that the team is predicted to score is displayed, as well as the total number of points that the team actually scored and the average score for the gameweek. Note that the number of points that each player actually scored and the team's total score would not be shown for a gameweek in the future, since these values are only known for gameweeks that have already finished.

Figure 20: A screenshot of the optimal team predictor page after the "Generate Team" button is pressed and the user inputs are submitted

9.2.2 Transfer Recommender

Figure 21 shows the page that is displayed when the transfer recommender page is loaded. On the left hand side, there are two input boxes in which the user can enter their team's FPL ID, and the gameweek which they want to create transfer recommendations from. In order to get the user's FPL team from its ID, the FPL API was used. The FPL API can be publicly accessed at the URL `https://fantasy.premierleague.com/api/`. The API endpoint which returns a user's team for a given gameweek is `https://fantasy.premierleague.com/api/entry/{team_id}/event/{gw}/picks/`. On the right hand side, there are two more input boxes. The "Lookahead Period" input allows the user to enter the number of weeks in advance that they want to generate transfer recommendations for. The "Number of free transfers" input allows the user to input how many free transfers that they currently have available.

Figure 21: A screenshot of the transfer recommender page before the user's team is imported and the "Generate Transfer Recommendations" button is pressed.

Figure 22 shows the page which is loaded after the user's FPL team has been imported and the transfer recommendations have been generated. Overlaid on the football pitch is the user's current starting eleven and below are the players that are currently on their bench. Then on the right hand side, under the "Generate Transfer Recommendations" button, the transfer recommendations are shown. For each of the upcoming gameweeks specified by the lookahead period input, the names of the player's which should be transfered in and out of the team are output first. Then any substitutions (players who should be swapped between the starting eleven and bench) are output. Finally, the predicted points gain of making these transfers and substitutions is output.

Figure 22: A screenshot of the transfer recommender page after the user's team is imported and the "Generate Transfer Recommendations" button is pressed.

10 Analysis of Social Media Data

After the development of the models in Sections 4,5 and 6, two key limitations were identified which were likely affecting the predictive accuracy of the player points' predictions generated by the models developed so far.

Firstly, if a player becomes injured before a match, this will have a significant impact on the number of minutes which they are likely to play in the match. In football, players typically become injured in two scenarios. Firstly, they can become injured during a match due to the impact of an opponent player's tackle, or by tearing/straining a muscle in their body. Secondly, they can obtain an injury during a training session, for the same reasons as during a match. López-Valenciano et al found that the number of injuries obtained by professional male footballers during a match was almost ten times higher than the number of injuries obtained during training [37]. Depending on the severity of the injury, a player may be fine to play in their next match, or have to miss multiple of their upcoming matches in order to recover from the injury. Currently our models have no way of knowing if a player has recently become injured since the features in our datasets only give an indication of a player and their team's recent performance, as well as data about their upcoming match. Therefore, if a player has been playing well recently, but they pick up an injury in their previous game which causes them to miss their upcoming match, they will score zero points but the model may predict them to score a much greater number of points. Clearly this will increase the error of average player points predictions made by the models, and could reduce the number of points scored by a predicted optimal team of players, if injured players are included in the team.

Secondly, players often play in football competitions seperate from the Premier League. These include European competitions, such as the Champions League, in which teams from Europe's top footballing leagues can play each other, and international matches, in which players will play for their nation's football team against other national teams. The features in our datasets only contain data for player's and team's performances in Premier League matches. This means that if a player has been playing well in a competition different from the Premier League then there is no way for the models to be aware of this currently.

The aim of this section is to investigate how data obtained from social media can be added as features to our datasets and address the limitations mentioned above. The initial objective was to try and collect as much data from social media as possible about each player prior to each gameweek of the 2023-24 and 2024-25 Premier League seasons.

10.1 Sources Considered

In this Section we explore the social media sources that were considered to collect the player data from, as well as the challenges that were faced in collecting this data.

10.1.1 X (Formally Named Twitter)

X is a social media platform where users share short posts which can contain text,images, or videos. X has around 600 million monthly users [38]. X is a popular online platform for sharing FPL related information such as news on players or teams, and recommendations on which players you should transfer in or out of your FPL team. In order to filter posts which are only relevant to FPL, the #fpl tag can be used.

A significant challenge faced was the volume of posts that could be read from X. To read posts from X, the official X API can be used, or external libraries such as Tweepy [39] can also be used. The X API includes a multiple access tiers which range from free to use to costing $5000 a month to use. Unfortunately, the free API tier does not allow any posts to be read from X. Therefore, the Tweepy library was the only feasible way to read posts from X. The `search_tweet` function in the library allows posts to be read from X which contain a specific search term. These posts can then be filtered further, such filtering by posts that were posted between a specified start and end date. However, X has strict rate limits imposed on the number of searches that can be performed within a given time period, which significantly restricts the volume of posts that can be read. Specifically, It was found that only 50 searches could be made every 15 minutes. To overcome this, five seperate X accounts were created which were rotated between, so while the rate limit period had been exceeded on one account, another account could be used to perform the searches instead. This helped to minimise

the time spent waiting for the rate limit period to reset on each account and allowed more posts to be read.

Rotating between X accounts and using Tweepy was an effective strategy, until all the accounts that were made got suspended by X for "violating our rules against inauthentic accounts". This meant that all of the accounts could no longer be used and it was no longer possible to read any more posts from X. As a result of this, the social media platforms Bluesky and Reddit were considered to see if they would provide a better source of data than X.

10.1.2 Bluesky

Bluesky is another social media platform similar to X where users share short posts which can contain text, images, or videos. Bluesky was originally an initiative developed by Twitter which aimed to create a decentralised version of Twitter. However, when Twitter was bought by Elon Musk in 2022 and renamed to X, all ties with Bluesky were cut and it became an independent platform.

Bluesky has a freely available API which can be accessed using the `atproto` Python library [40]. This meant that it was easy to read posts that had been posted on Bluesky. Similar to X, FPL related posts can be filtered by using the #fpl tag. However, Bluesky is a very new social media platform compared to X. Bluesky was only publicly launched in February 2024 and currently has approximately 35 million total users. It was found that there were significantly less posts about Premier League players, which contained the FPL tag, on Bluesky compared to X. This is likely due to the much lower user count of Bluesky, compared to X, and the higher popularity of FPL on X compared to Bluesky. Therefore it was decided that Bluesky was not a suitable source due to insufficient data on Premier League players.

10.1.3 Reddit

Reddit is a social media platform where users discuss content in individual communities. Each community is called a "subreddit". On Reddit, there is a dedicated FPL subreddit named FantasyPL which has over 750,000 members. The main difference between Reddit and X/Bluesky is the average length of posts made by users. On X and Bluesky, posts can contain a maximum of 280 and 300 characters respectively for standard users (Premium users have increased character limits for posts on X). However Reddit allows posts to have up to 40000 characters. Posts on X will usually contain a short piece of information about a single player whereas Reddit posts are much longer and often contain a long discussion about multiple players. In addition, the Reddit API is monetised and only allows access to the last 1000 posts from a subreddit. Therefore, it would not be possible to collect data for each player prior to each gameweek of the 2023/24 season, as the API would not be able to fetch posts from nearly two years ago.

10.1.4 Summary

The most appropriate source of FPL related social media data was found to be X. As mentioned previously, there were challenges faced with the amount of posts that could be fetched from X, but the volume of posts that were collected before the account suspensions occurred was deemed sufficient to provide a proof of concept about how the analysis of these posts would affect the player points predictions of the models that have been explored earlier. In the earlier Literature Review section, the paper by Bonello et al [14] was explored, which investigated the addition of sentiment analysis of data from Twitter to improve optimal FPL team predictions. It was found that sentiment analysis did not provide an improvement due to the poor grammar of tweets, so it is important that the techniques we will explore in the next section are more robust in tackling this issue.

10.2 Analysis of X Data

10.2.1 Data Collected

For every player that played in the 2023/24 and 2024/25 Premier League seasons, the original plan was to collect the top 20 posts, which contained the #fpl tag, that were posted about them prior to each game that they played in the season. The word "top" here refers to the posts that are returned in the "Top" section of a search query on X. These posts are ranked by a combination of the post's engagement score, health score, and relevance score [41]. In order to filter posts about a specific player, the search query passed to the Tweepy `search_tweet` function consisted of the player's web name and the #fpl tag. A player's web name refers to the name to which they are most commonly known by online. This was done to maximise the number of search queries that would be returned about each player, since players are not often referred to by their full name on social media platforms such as X. The search query also consisted of a string to ensure that the posts returned about a player were posted between a specified start and end date. This was to ensure that the posts returned about a player only contained recent information or opinions which were relevant to their upcoming match, and not previous matches that they had played in. Specifically, posts were filtered so that only posts that had been posted one day after a player's previous match and one day before their upcoming match were collected.

The aim of collecting these posts was to try and capture posts which mentioned a player's recent form or a possible injury that they had recently picked up. In the Premier League, before each team plays their match in a certain gameweek, the team's manager will give a press conference in which answer questions from the media. As part of this, they will usually outline any recent injuries that have occurred in the team and an estimate of how long it will take the injured players to recover. These quotes are often posted on X, so if there are popular posts referencing a player's injury then it is a good indication that they may not play in their next match. Figure 23 shows an example of a post which describes an injury for Manchester City forward Erling Haaland and tells us that he will not play in his team's upcoming game.

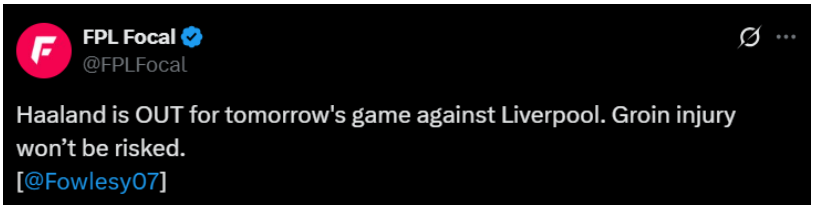

Figure 23: A post from X highlighting a recent injury for Haaland

After facing the challenges previously mentioned, the data described above was only able to be collected for players that played for Chelsea, Arsenal, and Wolverhampton Wanderers in the 2023/24 and 2024/25 seasons. To reduce the number of unnecessary searches made by the `search_tweet` function, a simple binary classifier model was trained to predict whether a player will score more than zero points in their upcoming game. This was implemented using an instance of the `sklearn.svm.LinearSVC` class, trained on the training dataset created in Section 4.1.5, but instead of having the number of FPL points that the player scored as the label, the label was 0 if the player scored zero FPL points and 1 otherwise. Note that players that played for Chelsea, Arsenal or Wolverhampton Wanderers were removed from the training dataset to prevent test data (players that we are collecting X posts for) appearing in the training dataset. The search tweet function was only called on players that were predicted to above 0 points, since player's predicted to score zero points are likely to not play at all during the match, or play very little.

The raw text from each set of player posts was then processed into a numerical format, so that information captured within each post could be added as new features to the training and test datasets that were created in Section 4.1.5 (for players that played for Chelsea, Arsenal, or Wolverhampton Wanderers). The three different processing methods that were tested are explored in the upcoming sections.

10.2.2 Sentiment Analysis

Sentiment analysis is the process of classifying the overall sentiment of a piece of text as a specific emotional tone. A standard sentiment analysis task aims to classify text as positive, negative, or neutral. In order to classify the sentiment of the X posts, a variant of the RoBERTa language model [42], which had been fine tuned for sentiment analysis and trained on twitter data, was chosen to be used [43]. This model was chosen due its improved performance over other sentiment analysis models and the fact that it had been trained on posts from twitter so the model was more likely to be robust at handling sentiment in FPL related posts on X.

The `transformers` Python library was used to import the pre-trained fine-tuned RoBERTa model. The steps that were taken in order to compute the sentiment of an X post are detailed below :

1. First, the text in the post was pre-processed. Any line breaks within the post were removed and replaced with a full stop. Then any hyperlinks and hashtags were removed from the text since they are not relevant to the post's sentiment.
2. After pre-processing, the text is tokenised. Tokenisation refers to the process of breaking down the text into small numerical units, named tokens. Tokenisation was done using the `transformers.AutoTokenizer` class.
3. The tokenised text is then passed as an input to the RoBERTa model, which outputs a score of how likely the text is positive, negative or neutral in sentiment.
4. The sentiment scores are converted to probabilities using the Softmax function. The probability that post z has sentiment i is defined by Equation 8, where z_j is the sentiment score for sentiment $j \in \{1, 2, 3\} = \{\text{Positive, Neutral, Negative}\}$.

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{3} e^{z_j}} \quad \text{for } i \in \{1, 2, 3\} \tag{8}$$

Then once the sentiment probabilities of all posts for a player have been computed, we calculate the sum of all the probabilities for the positive, neutral and negative sentiment classes respectively. These sums will be used as additional features in each player's feature vector.

10.2.3 Vector Embeddings

An embedding is a vector of floating point numbers which can be used to represent data such as text or images. Embeddings can be used to capture the sentiment of text and the contextual relationship between the words within the text, in a very high number of dimensions. An embedding model aims to capture the relationships between the words in a text and then represent this in a multi-dimensional vector. Before the embeddings of the posts were calculated, the text contained within the posts was pre-processed by removing line break, hashtags, and hyperlinks from the post text. In order to calculate the corresponding embedding vector of the text contained in an X post, the `text-embedding-3-small` OpenAI embedding model was used [44]. The `embeddings.create` function was used, as part of the `openai` Python library, to calculate the embedding vectors for each of the X posts that were collected. The default dimension of a vector returned by the text-embedding-3-small model is 1536. For each post, an embedding of dimension 1536 was calculated, as well as an embedding of dimension 512. Then for all the embedding vectors of the same dimension, calculated from the text in posts made about a player prior to a match they played in, the average of the embeddings were calculated to give a final embedding vector for that player. This average was computed to for the vector embeddings of dimension 1536 and 512. Each component of the averaged vector embeddings will be used as an additional feature in each player's feature vector.

10.2.4 LLM Categorisation

The final method that was tested was using a large language model (LLM) to categorise posts into one of four categories. The LLM that was used was the OpenAI `GPT-4o-mini` model [45] since it is a fast and affordable model, and it provides enough intelligence to be able to reliably understand the small amount of information contained within an X post. The `chat.completions.create` function, as part of openai Python library, was used to import the model. Again, the text contained within

each post was pre-processed, in the same way as the previous two sections, before being passed to the LLM. The four categories (A-D) that posts were categorised into are listed below :

A. The post tells us that the player has been recently, or is currently injured.

B. The post tells us that the player is currently in good form

C. The post tells us that the player is currently in bad form

D. The post does not fall into any of the first three categories.

In order to categorise each post into one of the four categories, a prompt was created which was given to the LLM, along with the pre-processed text from each model. The prompt used was "You will be provided with a tweet about {`player name`}, a football player. I want you to explain to me if you think he is currently injured (A), in bad form (B), if he is playing well (C) or if you are unsure (D). Tweets may contain information related to fantasy football, such as a player being transfered in or out of someone's team". Then once all of the posts for a player had been categorised, the total number of posts that belonged to each category was calculated. The frequency of each category will then be used as an additional feature in each player's feature vector (four new features, each representing the number of posts belonging to each category).

10.2.5 Results

After the three processing methods had been applied to the X posts for each player, six new training and test datasets were created, each which contain a combination of the new features obtained from the processing methods, and the features from th. The six new datasets and their features are listed below. Note that all datasets only contain players from Chelsea, Arsenal or Wolverhampton Wanderers, due to reasons previously mentioned. The training datasets contain players from the 2023/24 PL season, and the test datasets contain players from the 2024/25 season (up to gameweek 30). If there were no posts made about a player prior to a certain gameweek, then the values of the new features in each dataset were set to zero for that player. The six datasets created are listed below :

1. Base dataset. This dataset contains all of the features in the original datasets from Section 4.1.5 and no new features.

2. Sentiment dataset. This dataset contains all the features from the base dataset, plus a feature for each of the total positive, neutral and negative sentiment scores of the posts for the player.

3. Large embedding dataset. This dataset contains all the features from the base dataset, plus 1536 features, where each feature is a component of the averaged 1536 dimension vector embedding of all the posts for the player.

4. Small embedding dataset. This dataset contains all the features from the base dataset, plus 512 features, where each features is a component of the averaged 512 dimension vector embedding of all the posts for the player.

5. LLM categorisation dataset. This dataset contains all the features from the base dataset, plus a feature for number of total number of player posts that were categorised into each of the four categories.

6. Embedding and LLM categorisation dataset. This dataset contains all the features from the base dataset, plus the additional features from the large embedding and LLM categorisation datasets.

We now compare the metrics for the test predictions made by the HGBM model, from Section 5.3, trained and tested on each dataset type listed above. The results are shown below in Table 27. We can see that the model trained and tested on the LLM categorisation data has the lowest MSE and MAE, and the highest R^2 score. The model trained on the sentiment dataset had least reduction in error. This is consistent with the conclusions from Bonello et al [14], so it is likely that the poor grammar of X posts is still impactful, despite the RoBERTa sentiment model being fine tuned on tweets. This highlights that the `gpt_4o-mini` LLM was able to better understand the content of the posts better than the sentiment model. The models which were trained on the embedding datasets provided a larger reduction in error than the model trained on the sentiment data, but were narrowly higher in error than the model trained on the LLM categorisation dataset.

Dataset	RMSE	R^2	MAE
Dataset 1	2.541	0.198	1.664
Dataset 2	2.538	0.200	1.657
Dataset 3	2.394	0.235	1.528
Dataset 4	2.396	0.234	1.542
Dataset 5	**2.390**	**0.237**	**1.527**
Dataset 6	2.394	0.235	1.528

Table 27: Performance metrics (MSE, MAE, R^2) of the model on six different datasets.

11 Evaluation

In this section we summarise the main results of the project, and compare them to some of the results of the existing work that was discussed in the Literature Review section.

11.1 Summary of Results

During the course of the project, many types of models and datasets were explored to try and get the lowest error in predicting individual players and entire team FPL points.

The model with the lowest error for predicting player points was the HGBM model, trained and tested on the the original training and test datasets which were created in Section 4.1.5. Re-balancing the class imbalance in the training dataset provided an increase in the average player points predictions, and consequently a reduction in error for predicting players which were predicted to score, or actually scored above three points. It was found that the RF model was more suited to predicting the higher player points, than the HGBM model. This aligns with the conclusions found by Branco et al [33], where it was shown that applying the SMOGN algorithm to the training data for random forest models had one of the greatest improvements in model performance out of all the models which were tested.

The approach to use individual models to predict points for each player position (Section 7.1) did not reduce the error in player points predictions, compared to the preceding models. This highlighted that by having a feature for the player position in the datasets, the preceding models were already robust in handling points predictions for players in different positions, and that there was in fact no noise generated by features that were not relevant to certain player positions. Similarly, the approach to use individual models to predict the main factors which contribute to the number of points that a player scores did not provide a reduction in prediction error, compared to the preceding models (Sections 7.2). It was surprising to find that this approach gave the highest error in player points predictions, especially given the success that Valouxis [8] had with employing a similar strategy. One reason for this could be the difference in approaches to predicting the number of minutes that a player will play in their upcoming match. Valouxis does not explain in detail how the predicted minutes for each player are calculated but says : "we take into account the player's past minutes in the team, the player's form, his injury status, and important news from the press conferences the teams' managers provide before the matches". In Section 10, we explored the fact that the aim of analysing posts from X about each player was to introduce the capability for our models to detect player injuries and information from manager's press conferences. It was shown in Section 10.2.5 that even with the limited number of posts that we were able to collect, there was a reduction in error from categorising X posts using the `gpt_4o_mini` model. Therefore, if we were able to collect X posts for every player then I am confident it would provide an improvement to the player minutes predictions, and consequently reduce the error in player predicted points.

In Section 8, it was found that the optimal teams generated for gameweeks 1-30 of the 2024/25 PL season using the player points predictions from the HGBM model, trained on the original training data, scored the highest most frequently out of all the models developed during the project. Using the points predictions generated from the RF model trained on the SMOGN data produced the highest predicted scoring teams in all gameweeks, but only 25% of these teams actually scored highest in a certain gameweek. This suggests that using models with higher overall accuracy in player predicted points is the most important factor when trying to accurately predict the optimal team for a gameweek.

11.2 Comparison to Existing Work

Table 28 compares the best performing model that was created during the project (The HGBM model trained and tested on the original training and test datasets), with the best models developed by Bangdiwala et al [6] and Valouxis [8]. It is important to note that the training and test data for each model is different, since the training and test datasets used are comprised of data from different Premier League seasons and have different feature sets. Although our model does not beat the existing work in all metrics, it is fairly close to the results that were achieved. However, as mentioned previously, with the addition of a complete set of analysis of X posts for all players, in all of the seasons that data was collected for, I am confident that the metrics of our models would improve further and possibly even surpass the results achieved by the existing work.

	Model		
Metric	Bangdiwala et al.'s Model [6]	Valouxis' Model [8]	My Model
RMSE	2.146	2.300	2.254
R^2	N/A	0.333	0.255
MAE	1.215	1.290	1.369

Table 28: Comparison of the best performing models from the literature review literature and my best performing model

12 Conclusions and Future Work

12.1 Project Conclusion

As defined in Section 1.4, the main objectives of the project were :

1. **Dataset Creation**
   Create the training and test datasets required to train and test the machine learning models, by using existing sources of data available online.
2. **Machine Learning Model Development**
   Develop and test different machine learning models and identify the best performing model for predicting player and team points.
3. **Analysis of Social Media Data**
   Obtain data from a suitable social media source and test different methods of analysing the data. Investigate how the addition of this data to the training and test datasets affects the points predictions of the machine learning models.
4. **Development of Web Application**
   Create a web application to be able to display the optimal team for a given gameweek.

All of these objectives have been met to an acceptable standard. The datasets contain a wide range of features from multiple sources, which capture the historical performance of a player and their team over the current and previous seasons, as well data relevant to their upcoming match in each gameweek. Several machine learning models were developed to predict individual player FPL points and many experiments were carried out to try and improve these predictions, some more successful than others. These individual points predictions were then used to predict the optimal team for each gameweek of the 2024/25 PL season, by using linear programming. The additional objective of creating an algorithm to recommend player transfers for upcoming gameweeks was partially completed. The time constraints of the project meant that there was only time to implement the algorithm for a lookahead period of one.

The analysis of social media data was successful, despite the challenges faced with collecting posts from X. The results that were obtained, on the data that was collected, showed that if the original planned volume of data was collected then it would improve the accuracy of the predicted points produced by the machine learning models developed during the project. Finally, a web application was developed which allowed the optimal team to be displayed for a given gameweek, and transfers to be recommended for an upcoming gameweek (only for a lookahead value of one).

12.2 Future Work

I believe that with the addition of some of the improvements suggested below, the success of some of the project objectives could be increased further, and therefore improve the overall quality results obtained during the project. Many of the improvements were not implemented due to the natural time constraints of the project.

1. Increase the amount of training data and include more features in the datasets - the training data was chosen to include three seasons of data because the data in Vastaav's dataset had a slightly different set of features for seasons before the 2021/22 season. It would likely be possible to get the missing features from other online data sources. Increasing the volume of training data would increase the number of samples of players scoring a high number of FPL points and would increase the accuracy of the models predictions on high scoring players. Also, other features could be added to the datasets such as if a player takes penalties, free kicks or corners for their team. Players which take penalties for their team have a higher chance of scoring in matches, and if a player takes free kicks or corners then this can often lead to them getting more assists in matches.
2. Try to collect more posts from X - with more time available, more X accounts could have been created to collect posts, and more careful measures taken to reduce the chances of getting the accounts suspended. For example, rotating proxy servers could be used to repeatedly change the IP address of the device making the request to X. By collecting posts for every player in the training and test datasets, the error of the models' player points predictions would likely be reduced.

3. Use other sources of social media data such as blog posts or news articles - as explored in the literature review, Bonello et al [14] found that the use of sentiment analysis on news articles and blog posts was more successful than twitter posts. In Section 10, we showed that the vector embedding and LLM categorisation methods were more robust at handling the information in X posts, so these methods could also be applied to the content in blog posts and news articles. The length of the content in blog posts and news articles is significantly longer than the content in X posts, so methods such as named entity recognition would need to be applied in order to extract the relevant data for each player mentioned within the content.
4. Explore the use of deep learning for player points predictions - instead of using traditional machine learning models, neural networks could be used. This may lead to a reduction in error of the player points predictions.
5. View the problem as a classification problem rather than a regression problem - although the number of FPL points that a player scores can technically be any integer value, it is very rare that a player scores less than -2 points, or greater than 25 points in a match. Therefore, by restricting predicted player points to integers within this range, the task of predicting player FPL points could be modelled as a classification task.

12.3 Project Self Assessment

On reflection, I am pleased with the work I have completed during the course of the project and have enjoyed the entire process. The greatest contributions which were made during the project include using the SMOGN algorithm to re-balance the dataset and improve the predictive capabilities of the models for high scoring players, and exploring different analysis methods to extract the information contained X posts which were related to FPL. I am also proud of the web app which was created, and there is plenty of room for further development which could see it match the state of the art solutions, such as Fantasy Football Hub [18]. The project explores multiple concepts in machine learning and natural language processing (NLP), which are at the forefront of modern computer science. Before the project, I had very limited knowledge of machine learning and NLP and this has helped to develop my understanding of many important concepts and encourage me to explore these fields further in the future.

The work done in the project could be built by others in many different ways. The training and test datasets created would provide a natural foundation to begin a similar type of project, and could be developed even further by adding more data or features. As mentioned in Section 12.2, collecting a larger volume of posts from X or applying the methods in Section 11.2 to other social media sources would be ways to build on the contributions of the project. The key limitation of the project were the issues faced with collecting the posts from X. With more time available, more posts could have been collected, or data collected from other sources instead.

References

[1] Rick Burton, Kevin Hall, and Rodney Paul. "The historical development and marketing of fantasy sports leagues". In: *The Journal of Sport* 2.2 (2013), pp. 185–215. (accessed : 14.04.2025).

[2] Fantasy Sports and Gaming Association. *Industry Demographics*. URL: https://thefsga.org/industry-demographics/. (accessed : 14.04.2025).

[3] Premier League. *Fantasy Premier League*. URL: https://fantasy.premierleague.com/. (accessed : 02.10.2024).

[4] Premier League. *Rules*. URL: https://fantasy.premierleague.com/help/rules. (accessed : 02.10.2024).

[5] Fantasy Premier League. *How the FPL Bonus Point System works*. URL: https://www.premierleague.com/news/106533. (accessed: 17.11.2024).

[6] Malhar Bangdiwala et al. "Using ML Models to Predict Points in Fantasy Premier League". In: Oct. 2022. DOI: 10.1109/ASIANCON55314.2022.9909447. (accessed : 10.10.2024.

[7] Vaastav Anand. *Fantasy Premier League*. URL: https://github.com/vaastav/Fantasy-Premier-League. (accessed: 2.10.2024).

[8] Spiros Valouxis. "Machine Learning in Fantasy Premier League". In: (June 2023). DOI: https://dspace.lib.ntua.gr/xmlui/bitstream/handle/123456789/58026/Machine_Learning_in_Fantasy_Premier_League.pdf?sequence=1&isAllowed=y. (accessed : 11.10.2024).

[9] Understat. *Understat*. URL: https://understat.com. (accessed : 2.10.2024).

[10] FiveThirtyEight. *Club Soccer Predictions*. Archive : https://web.archive.org/web/20230813053417/https://projects.fivethirtyeight.com/soccer-predictions/. URL: https://projects.fivethirtyeight.com/soccer-predictions/. (accessed : 25.04.2025).

[11] Internet Archive. *Wayback Machine*. URL: https://web.archive.org/. (accessed : 25.04.2025).

[12] Opta Analyst. *What is Expected Goals (xG)?* URL: https://theanalyst.com/2023/08/what-is-expected-goals-xg. (accessed : 22.04.2025).

[13] Marina Dunbar. *Political poll news site 538 to close amid larger shuttering across ABC and Disney*. URL: https://www.theguardian.com/us-news/2025/mar/05/abc-news-538-shutdown. (accessed : 25.04.2025).

[14] Nicholas Bonello et al. "Multi-stream Data Analytics for Enhanced Performance Prediction in Fantasy Football". In: (Dec. 2019). DOI: 10.48550/arXiv.1912.07441. (accessed : 23.10.2024).

[15] Harmon, Joshua Roesslein, and other contributors. *Tweepy*. DOI: 10.5281/zenodo.7259945. URL: https://github.com/tweepy/tweepy.

[16] API-SPORTS. *API-FOOTBALL*. URL: https://rapidapi.com/api-sports/api/api-football. (accessed : 10.04.2025.

[17] Fréderic Godin et al. "Beating the Bookmakers: Leveraging Statistics and Twitter Microposts for Predicting Soccer Results". In: Aug. 2014. DOI: 10.13140/2.1.2168.0000. (accessed : 11.10.2024).

[18] Will Thomas. *Fantasy Football Hub*. URL: https://www.fantasyfootballhub.co.uk/. (accessed: 17.11.2024).

[19] Python Software Foundation. *Python*. URL: https://www.python.org/. (accessed : 06.04.2025).

[20] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830. (accessed : 6.04.2025).

[21] The pandas development team. *pandas-dev/pandas: Pandas*. Version 2.23. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134. (accessed : 05.04.2025).

[22] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55. (accessed : 6.04.2025).

[23] Google. *Google Colab*. URL: https://colab.research.google.com/. (accessed : 06.04.2025).

[24] Richard Cunningham. *Batch Compute System*. URL: https://warwick.ac.uk/fac/sci/dcs/intranet/user_guide/batch_compute/. (accessed : 06.04.2025).

[25] Pallets. *Flask*. URL: https://palletsprojects.com/projects/flask. (accessed : 06.04.2025).

[26] Fantasy Premier League. *How the ICT index in Fantasy works*. URL: https://www.premierleague.com/news/65567. (accessed: 17.11.2024).

[27] Amos Bastian. *A Python package for Understat*. URL: https://understat.readthedocs.io/en/latest/index.html. (accessed : 16.11.24).

[28] Joseph Buchdahl. *Football-Data*. URL: https://www.football-data.co.uk/. (accessed : 07.04.2025).

[29] Shankar Rao Pandala. *Lazy Predict*. URL: https://pypi.org/project/lazypredict/. (accessed : 17.11.24).

[30] Roger L.Berger George Casella. *Statistical Inference (Second Edition)*. Brooks/Cole, Cengage Learning, 2002. (accessed : 21.11.24).

[31] Alois Knoll Alexey Natekin. "Gradient boosting machines, a tutorial". In: *Frontiers in Neurorobotics* (2013). DOI: 10.3389/fnbot.2013.00021. (accessed : 20.11.24).

[32] scikit learn. *HistGradientBoostingRegressor*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.HistGradientBoostingRegressor.html. (accessed : 09.04.2025).

[33] Paula Branco, Luís Torgo, and Rita P. Ribeiro. "SMOGN: a Pre-processing Approach for Imbalanced Regression". In: *Proceedings of the First International Workshop on Learning with Imbalanced Domains: Theory and Applications*. Ed. by Paula Branco Luís Torgo and Nuno Moniz. Vol. 74. Proceedings of Machine Learning Research. PMLR, 2017, pp. 36–50. (accessed : 21.11.24).

[34] Nick Kunz. *A Python implementation of Synthetic Minority Over-Sampling Technique for Regression with Gaussian Noise (SMOGN)*. URL: https://pypi.org/project/smogn/. (accessed : 21.11.24).

[35] Stuart Mitchell and contributors. *Optimization with PuLP*. URL: https://coin-or.github.io/pulp/. (accessed : 20.02.2025).

[36] Armin Ronacher. *Jinja*. URL: https://jinja.palletsprojects.com/en/stable/. (accessed : 11.04.2025).

[37] Alejandro et al López-Valenciano. "Epidemiology of injuries in professional football: a systematic review and meta-analysis". In: *British journal of sports medicine vol. 54,12* (2020), pp. 711–718. (accessed : 10.04.2025).

[38] Fabio Duarte. *X (Formerly Twitter) User Age, Gender, Demographic Stats (2025)*. URL: https://explodingtopics.com/blog/x-user-stats. (accessed : 20.04.2025).

[39] Harmon, Joshua Roesslein, and other contributors. *Tweepy*. DOI: 10.5281/zenodo.7259945. URL: https://github.com/tweepy/tweepy. (accessed : 14.02.2025).

[40] Ilya Siamionau. *The AT Protocol SDK*. URL: https://pypi.org/project/atproto/. (accessed : 15.02.2025).

[41] X. *Search Recommendations*. URL: https://help.x.com/en/resources/recommender-systems/search-recommendations. (accessed : 13.04.2025).

[42] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. arXiv: 1907.11692 [cs.CL]. URL: https://arxiv.org/abs/1907.11692. (accessed : 20.04.2025).

[43] Francesco Barbieri et al. "TweetEval:Unified Benchmark and Comparative Evaluation for Tweet Classification". In: *Proceedings of Findings of EMNLP*. 2020. (accessed : 21.04.2025).

[44] OpenAI. *Vector Embeddings*. URL: https://platform.openai.com/docs/guides/embeddings. (accessed : 27.03.2025).

[45] OpenAI. *GPT-4o mini*. URL: https://platform.openai.com/docs/models/gpt-4o-mini. (accessed : 27.03.2025).