Loughborough University

# Report

## Predicting Premier League matches using Machine Learning

(s) Joshua Quartey

Student no: F027815

Final Year Project

C0C255

Department of Computer Science

Loughborough University

## Acknowledgements

I would like to sincerely thank everyone that helped make this project possible, especially for Tao Chen my supervisor helping me structure my report and constantly providing feedback along the way. I would also like to thank my friends and family for supporting me and keeping me motivated throughout the year.

## Abstract

For my final year project, I will be attempting to predict premier league matches to evaluate the predictability of the league. I will do this by using machine learning and data I will collect from previous premier league seasons.

The idea behind this decision was after Manchester City won the league for the 4th time in 5 years and the usual top 6 remained. Is the premier league predictable? Using data collected from many different sources and machine learning I will attempt to make an algorithm to try to predict matches in the premier league and compare them with the actual games. By doing this I will be able to test my claims earlier on the predictability of the premier league.

Due to its versatility, simplicity and reliable tools, I have opted to use python as my sole language to construct my program. Python is seen as the best machine learning language, as stated by Christina Voskoglou from Towards DataScience, '57% of data scientists and machine learning developers using it' [1]. There are also hundreds of python libraries that would be very useful when incorporating machine learning into my code. This will enable me to integrate machine learning into my program effectively.

I feel this project will be very interesting to work on. As well as combining my passions for football, computing and mathematics, it also allows me to see how the premier league is viewed by AI and I can observe the different variables that make the league 'predictable'.

This report documents the progress I have made across the year and goes in-depth into explaining how I accomplished this project.

# Table of Contents

# 1. Introduction

## 1.1. Project Motivation

Is the premier league predictable? The premier league is the best league in the world, many would say that its due to its high level and intensity, others would say it's down to the unknown outcome of every game. Teams towards the bottom of the table like Southampton can oppose a high-end club such as Manchester united and Chelsea and take a point or even 3. However recently teams such as Manchester city and Liverpool have been finishing seasons with more than 90 points, losing a total of 2-4 games throughout the stretch of a 38-game season. Are teams just expected to lose now?

Which comes back to my original question, has it now become predictable? In order to investigate this matter, I intend to utilize a plethora of data sources and apply machine learning techniques to construct an algorithm with the objective of forecasting match results in the Premier League. This will allow me to evaluate the accuracy of my hypothesis at an earlier stage, with respect to the presumed predictability of this esteemed football league.

## 1.2. Project aim

With all this in mind, my aim for this project is to make predictions on Premier League matches and incorporate my own league table. The table in comparison to the real one will show a visible representation of how my program predicted the league. According to how accurate the table is will help decide my argument as well as being a useful tool to make predictions in the league. I will focus on the development of the precision on the machine learning as I improve my program.

## 1.3. Objectives

For my model to reach its full potential, I have created a list of objectives. This will not only be a measure of success for this project but the basis for my machine learning model. It will provide me with guidelines to check against in order to ensure that I am fulfilling the criteria. During the development stage of this project, the targets listed below, will be at the very core of the process.

- The CSV file should have at least 4 seasons of data.
- There should be multiple predictors to help precision of the machine learning built.
- The program should be able to predict a win, lose or draw.
- The program should be able to make precise predictions using recent form as well as previous seasons.
- The project should be able to simulate all possible matchups in the league.
- A premier league table should be made of all my predictions at the end.

# 2. Background

## 2.1. Best league in the world

“I think the Premier League is the most unpredictable league in the world. That’s why it’s the best league in the world. Anyone can beat anyone. You can have a team that’s not expected to win anything, but they can go on and win the league. That’s why the premier league is so special.” Former Arsenal player, and premier league hall of famer Thierry Henry on the English division. Although my claims and tests I will be running are slightly different to this quote I do completely agree with his statement. The quality in the premier league is unquestionable. It is the only league where 6 clubs are seen as title contenders every season, and the strength of these 6 clubs were shown in Europe in 2019 when four clubs (Liverpool, sours, arsenal and Chelsea) all met in European cup finals. This was the first time in football history. Outside the top 6 is also very competitive and can make life very difficult for every team from 1$^{st}$ to 20$^{th}$.

Some of the great players of today’s views on the premier league [11]:

- "The Premier League is the toughest league in the world. You have to be at your best every game, otherwise you will lose." Sergio Aguero, Manchester City.
- "It's not easy to come here and play. The pace and physicality of the game is something that you have to get used to." - N'Golo Kante, Chelsea.
- "Every game in the Premier League is like a cup final. You have to be prepared to battle for every point." - Virgil van Dijk, Liverpool.
- "The Premier League is very competitive. There are no easy games, and you have to be focused for 90 minutes every week." - Kevin De Bruyne, Manchester City.
- "The Premier League is tough, physically and mentally. You have to be strong and focused to succeed." - Harry Kane, Tottenham Hotspur.

## 2.2. Leicester 2015/16 champions

The phenomenon that is Leicester 2015/16 is the prime example of the premier league at its very best. Leicester City's league-winning season in 2015-2016 is considered one of the greatest upsets in football history. The team, which had narrowly avoided relegation the previous season, defied the odds and won the Premier League title by a comfortable margin, finishing 10 points ahead of second-placed Arsenal.

The mathematical calculation of Leicester City's chances of winning the Premier League title that season was based on bookmakers' odds. At the start of the season, Leicester City were considered 5000-1 outsiders to win the league, meaning that for every £1 bet on them to win, the bookmakers would pay out £5000 if they did. To put this into perspective, the odds of Elvis Presley being found alive in 2015 were also 5000-1, which gives an idea of just how unlikely Leicester's triumph was considered at the time [12].

| | Probability of finishing top in the English Premier League |
|---|---|
| Manchester United | 14% |
| Manchester City | 32% |
| Chelsea | 42% |
| Arsenal | 13% |
| Liverpool | 2% |
| Tottenham Hotspur | 2% |

(Figure 1.1)

(That season these were the probability that each of the top 6 would win the league.)

The actual probability of Leicester City winning the Premier League in the 2015-2016 season can be calculated by taking the reciprocal of their odds of winning. To calculate their actual probability of winning the league, we need to convert the odds to a fraction and take the reciprocal of that fraction. The odds of 5000-1 can be written as a fraction of 1/5000. So, the actual probability of Leicester City winning the Premier League in the 2015-2016 season was approximately 1 in 5000, or 0.02%. In comparison to the top 6 Chelsea and Manchester City, 42% and 32% respectively it would have been close to impossible to anticipate the winners.


(Figure 1.2)

The graph shows the probability of each team finishing in the top 4 that very season in comparison to their wage ratio. (The wage ratio is the wage spend of a club relative to the league average). Leicester alongside Crystal palace and Norwich and at the lowest for both. Generally speaking, teams with higher wage ratios have an advantage over those with lower ratios, as they are able to attract and retain better players. This is because top players are usually paid higher salaries than average or lower-level players, and teams with more financial resources can afford these high salaries. Simply

going off odds to not show the actual probability of Leicester winning as factors such as Wage ratio also come into play.

Leicester's success that season can be attributed to several factors, including the tactical genius of their manager, Claudio Ranieri, the excellent form of key players such as Jamie Vardy and Riyad Mahrez, and the team's ability to play as a cohesive unit and consistently perform to a high level throughout the season.

So, is the premier league unpredictable or was this season just an outlier?

## 2.3. Games and Gambling

However, despite this unpredictability, companies such as EA sports FIFA, Football Manager and gambling sites like Bet365 use statistical analysis and data to predict outcomes with a certain level of accuracy. These companies have access to a wealth of data, including player performance statistics, team form, injury news, and other factors that can influence the outcome of a game.

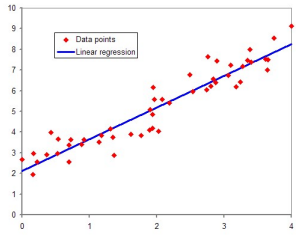
FIFA Football Manager, for example, uses sophisticated algorithms and data analysis to simulate football matches and predict outcomes. These simulations consider a wide range of variables, such as team strength, tactics, player form, and even weather conditions.

Similarly, gambling sites such as Bet365 use data analysis to offer odds on matches and predict the most likely outcome. This data is based on a variety of factors, including team form, player injuries, and head-to-head records. Despite the use of advanced data analysis and statistical models, the Premier League remains highly unpredictable. The league is known for its upsets and surprises, with even the most dominant teams suffering unexpected defeats. While companies such as FIFA Football Manager and Bet365 can offer predictions based on data analysis, there are no guarantees in football, and the outcome of any given match can never be fully predicted.

## 2.4. Maths and Machine Learning

Mathematics plays a crucial role in analysing the predictability of the Premier League. There are several mathematical techniques that can be used to analyse and predict the outcomes of football matches.

One common technique used in sports analytics is regression analysis. Regression analysis involves identifying a set of independent variables, such as team form, player statistics, and head-to-head records, and using these variables to predict the outcome of a game. Regression models can be used to estimate the probability of a certain outcome, such as a win or a draw. (Figure 1.3)




Another technique used in sports analytics and what I will be using is machine learning. Machine learning involves training a model using historical data, such as previous Premier League results, to

identify patterns and predict future outcomes. These models can incorporate a wide range of variables and can be continually updated to improve their accuracy.

Furthermore, mathematical models can be used to simulate football matches and predict outcomes. These models consider various factors such as the expected goals scored and the likelihood of a particular player scoring a goal. Monte Carlo simulations are one type of model that can be used to simulate matches and estimate the likelihood of different outcomes. A Monte Carlo simulation is a model used to predict the probability of a variety of outcomes when the potential for random variables is present. Monte Carlo simulations help to explain the impact of risk and uncertainty in prediction and forecasting models.

Finally, statistical analysis can be used to identify trends and patterns in the data that may not be immediately apparent. This analysis can be used to identify factors that are most strongly correlated with a team's performance and can be used to adjust predictions accordingly.

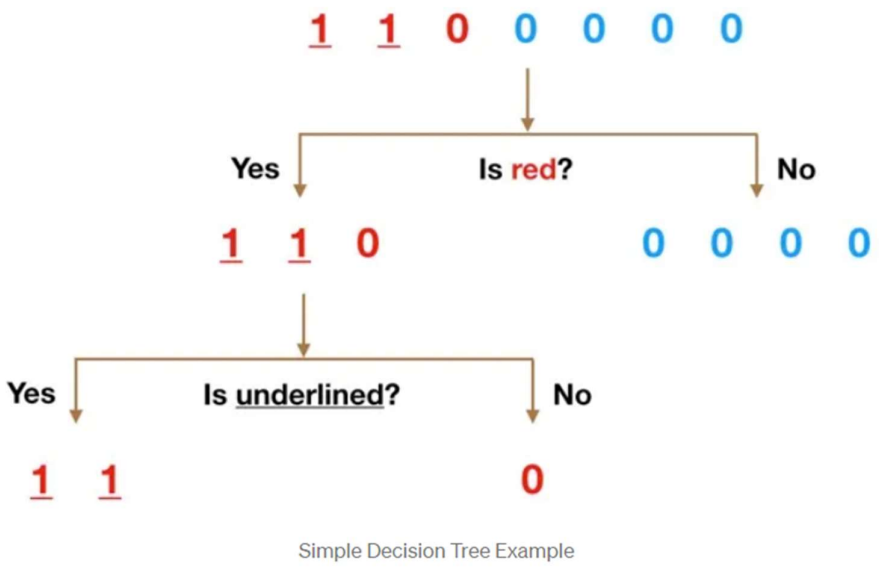
In conclusion, mathematics can be used to analyse the predictability of the Premier League by using techniques such as regression analysis, machine learning, and simulation models. These techniques can be used to identify patterns and predict outcomes based on a wide range of variables. By using mathematics to analyse the predictability of the Premier League, companies and analysts can gain a better understanding of the factors that contribute to a team's success and make more accurate predictions about the outcome of the matches.

# 3. Literature review

The Premier League is one of the most popular football leagues in the world, and predicting the outcomes of matches is of great interest to fans, gamblers, and football experts. With the advent of machine learning techniques, it is possible to develop models that can predict the results of Premier League matches accurately. In this literature review, we will examine the existing research on using machine learning to predict the outcomes of Premier League matches.

## 3.1. Decision trees

Decision trees are a type of machine learning algorithm that involves recursively partitioning the feature space into smaller regions in order to make predictions or classifications. The tree is built by selecting the feature that provides the most information gain at each split. Decision trees are easy to interpret and can handle both categorical and numerical data, but they are prone to overfitting and may not generalize well to new data.



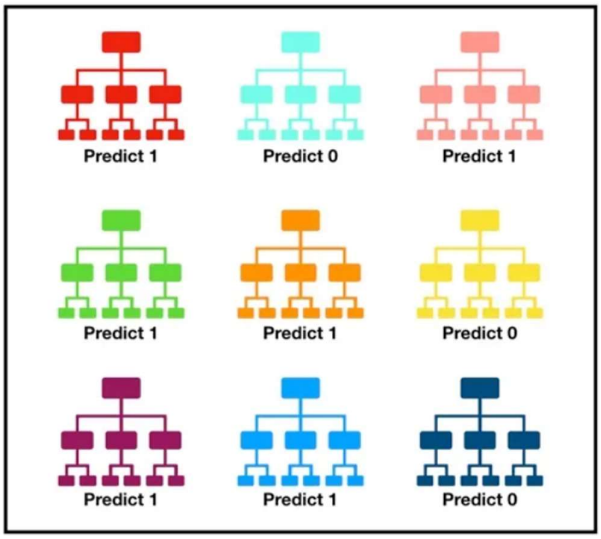

Simple Decision Tree Example

(Figure 2.1)

Decision trees are useful because they are simple to understand and interpret and can be used for both classification and regression tasks. They can handle both numerical and categorical data and can be used to identify important features in a dataset. One of the key advantages of decision trees is that they can handle non-linear relationships between features and the target variable. They are also able to handle missing data and outliers and can be easily modified or updated as new data becomes available. In addition, decision trees can be used to build more complex models such as random forests and gradient boosting, which can improve prediction accuracy and handle larger datasets.

### 3.1.1. Vs Random Forest

"Random forest is an ensemble method that combines multiple decision trees to produce a more accurate and robust model." In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone (Wikipedia). Each tree is trained on a random subset of the data and a random subset of the features. This randomness helps to reduce the variance and overfitting issues that decision trees may face and can improve prediction accuracy. Random forest can also handle missing values and can be used for feature selection. However, the resulting model can be more difficult to interpret than a single decision tree.




Visualization of a Random Forest Model Making a Prediction (Figure 2.2)

'Tony Yui states the fundamental concept behind random forest is a simple but powerful one — the wisdom of crowds. In data science speak, the reason that the random forest model works so well is a large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models. [14]

The algorithm procedures are:

1. Given there is a data set $D = \{x_{i1}, x_{i2}, \ldots, x_{in}, y_i\}\,(i \in [1, m])$ that has N number of features, the samples under bootstrap sample $(m * n)^{m*n}$ can generate sampling space.
2. Building a decision tree: sampling each one $d_j = \{x_{i1}, x_{i2}, \ldots, x_{ik}, y_i\}\,(i \in [1, m])$ (k < m) to generate the decision tree and record each result.
3. Training T times let $H(x) = max \sum_{t=1}^{T} \phi(h_j(x) = y)$ in the formula that $\phi(x)$ which is a type of decision tree (includes absolute majority, plurality and weighted voting, etc)

During steps (1) and (2), the input samples for each tree are not comprised of the entire sample set. Each decision tree is built using a fully split method, meaning that a leaf node cannot be further divided, and all samples belonging to the same class are grouped together. These methods ensure sample randomness without requiring branch pruning and help prevent overfitting issues.

In general, decision trees may be preferred if interpretability is a priority, while random forest may be preferred if prediction accuracy is a priority and interpretability is not as critical. Additionally, random forest may be more suitable for datasets with high dimensionality or many features, while decision trees may work better for smaller datasets with fewer features.

In "Predicting English Premier League Football Match Outcomes with Machine Learning Techniques" (Armatee, 2019) [3], the author used multiple machine learning algorithms, including decision tree, random forest, and support vector machine, to predict the outcomes of Premier League matches. The model was trained on data from the 2015/2016 season and tested on the 2016/2017 season.

The results showed that the random forest algorithm achieved the highest accuracy of 53.75% in predicting the outcomes of matches. The author found that including additional features, such as in-game statistics and team form, could improve the model's accuracy.

In "Portfolio Project: Predicting EPL Football Match Winners Using Machine Learning" (Dataquest, 2021), this time the author used logistic regression, decision tree, and random forest algorithms to predict the outcomes of Premier League matches. The model was trained on data from the 2016/2017 to 2019/2020 seasons and tested on the 2020/2021 season. Once again, the results showed that the random forest algorithm achieved the highest accuracy of 55.16% in predicting the outcomes of matches [5]. The author also found that including additional features, such as home advantage and betting odds, could improve the model's accuracy.

### 3.2. Neural network and gradient boosting

On the other hand, in "Machine Learning-Based Football Match Outcome Prediction for the English Premier League" (Islam et al., 2021) [4], the authors used a combination of neural network and gradient boosting algorithms to predict the outcomes. The model was trained on data from the 2015/2016 to 2019/2020 seasons and tested on the 2020/2021 season. The results showed that the model achieved an accuracy of 53.91% in predicting the outcomes of matches. The authors also found that including additional features, such as the distance between teams and the number of days between matches, could improve the model's accuracy [15].

Neural networks and gradient boosting are both popular machine learning techniques used for prediction and classification tasks. However, they have differences in terms of their architecture, training process, and strengths/weaknesses. Here is a general overview:

Differences:

- Architecture: Neural networks are composed of layers of interconnected nodes (neurons) organized into input, hidden, and output layers, while gradient boosting typically uses decision trees as base learners.
- Training process: Neural networks are typically trained using backpropagation, where weights are updated in each iteration, while gradient boosting uses an iterative approach that adds new trees to correct the errors of previous trees.
- Interpretability: Neural networks can be complex and difficult to interpret, while gradient boosting models can be more interpretable as they are composed of decision trees.
- Scalability: Neural networks can handle large datasets and high-dimensional data, while gradient boosting can be computationally expensive and may not be as suitable for very large datasets.

Similarities:

- Ensemble methods: Both neural networks and gradient boosting are considered as ensemble methods that combine multiple models to improve prediction accuracy.
- Non-linearity: Both techniques can capture non-linear relationships between features, which can be beneficial in capturing complex patterns in data.
- Prediction accuracy: Both neural networks and gradient boosting can achieve high prediction accuracy, often outperforming traditional linear models in many cases.

### 3.3. Summary

The above studies demonstrate that machine learning techniques can be used to predict the outcomes of Premier League matches with a reasonable degree of accuracy. While some of the models had lower accuracy levels than others, including additional features, such as player ratings, team statistics, and betting odds, can improve the accuracy of the models. Furthermore, the studies show that combining multiple machine learning algorithms can also result in higher accuracy levels. Overall, the research suggests that machine learning can be a useful tool for predicting Premier League match outcomes and can potentially provide valuable insights for football enthusiasts and industry expert.

## 4. Collecting Data

The forthcoming chapter aims to explicate the process of acquiring data for machine learning through an array of diverse methodologies. The primary objective is to show how collected data for my machine learning model with a comprehensive understanding of the intricacies involved in data gathering and preparation, including techniques such as web scraping, data cleaning, and data augmentation.

### 4.1. Data Scraping with requests

Data scraping refers to the automated process of extracting data from websites or other sources of information. This process can be used to collect large amounts of data, which can then be analysed and used for various purposes. Scraping data is an important step before machine learning because machine learning algorithms require large amounts of high-quality data to train and develop predictive models. However, it's important to note that not all data is suitable for machine learning. The quality of the data is important, as is its relevance to the problem being solved. Therefore, it's important to carefully select and pre-process the data before using it to train machine learning models.

Premier League History | 2022-2023 Premier League Overview | Scores & Fixtures | Squad & Player Stats ▾ | Nationalities | Squad & Player Wages | Other 2022-2023 Leagues ▾

| Rk | Squad | MP | W | D | L | GF | GA | GD | Pts | Pts/MP | xG | xGA | xGD | xGD/90 | Last 5 | Attendance | Top Team Scorer | Goalkeeper | Notes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Arsenal | 30 | 23 | 4 | 3 | 72 | 29 | +43 | 73 | 2.43 | 58.9 | 31.6 | +27.3 | +0.91 | W W W W D | 60,205 | Martinelli - 14 | Aaron Ramsdale | |
| 2 | Manchester City | 29 | 21 | 4 | 4 | 75 | 27 | +48 | 67 | 2.31 | 61.9 | 22.2 | +39.7 | +1.37 | W W W W W | 53,194 | Erling Haaland - 30 | Ederson | |
| 3 | Newcastle Utd | 29 | 15 | 11 | 3 | 48 | 21 | +27 | 56 | 1.93 | 49.9 | 28.8 | +21.1 | +0.73 | W W W W W | 52,243 | Miguel Almirón - 11 | Nick Pope | |
| 4 | Manchester Utd | 29 | 17 | 5 | 7 | 44 | 37 | +7 | 56 | 1.93 | 45.8 | 38.0 | +7.8 | +0.27 | L D L W W | 73,704 | Marcus Rashford - 15 | David de Gea | |
| 5 | Tottenham | 30 | 16 | 5 | 9 | 55 | 42 | +13 | 53 | 1.77 | 43.2 | 36.1 | +7.1 | +0.24 | L W D D W | 57,543 | Harry Kane - 23 | Hugo Lloris | |
| 6 | Aston Villa | 30 | 14 | 5 | 11 | 41 | 40 | +1 | 47 | 1.57 | 38.4 | 43.7 | -5.3 | -0.18 | D W W W W | 41,664 | Ollie Watkins - 12 | Emiliano Martínez | |
| 7 | Brighton | 28 | 13 | 7 | 8 | 52 | 36 | +16 | 46 | 1.64 | 51.0 | 33.0 | +18.0 | +0.64 | D W D W L | 31,460 | Alexis Mac Allister - 8 | Robert Sánchez | |
| 8 | Liverpool | 29 | 12 | 8 | 9 | 50 | 35 | +15 | 44 | 1.52 | 52.0 | 41.2 | +10.8 | +0.37 | W L L D D | 49,397 | Mohamed Salah - 13 | Alisson | |
| 9 | Brentford | 30 | 10 | 13 | 7 | 47 | 40 | +7 | 43 | 1.43 | 44.0 | 39.8 | +4.2 | +0.14 | W D D L L | 17,073 | Ivan Toney - 18 | David Raya | |
| 10 | Fulham | 29 | 11 | 6 | 12 | 39 | 40 | -1 | 39 | 1.34 | 35.2 | 48.6 | -13.5 | -0.46 | D L L L L | 23,568 | Aleksandar Mitrović - 11 | Bernd Leno | |
| 11 | Chelsea | 30 | 10 | 9 | 11 | 29 | 31 | -2 | 39 | 1.30 | 39.0 | 36.2 | +2.8 | +0.09 | W D L D L | 39,988 | Kai Havertz - 7 | Kepa Arrizabalaga | |
| 12 | Crystal Palace | 30 | 8 | 9 | 13 | 29 | 40 | -11 | 33 | 1.10 | 30.2 | 40.5 | -10.3 | -0.34 | L L L W W | 24,900 | Wilfried Zaha - 6 | Vicente Guaita | |
| 13 | Wolves | 30 | 8 | 7 | 15 | 24 | 42 | -18 | 31 | 1.03 | 30.0 | 42.9 | -13.0 | -0.43 | W L L D W | 31,585 | Daniel Podence - 6 | José Sá | |
| 14 | West Ham | 29 | 8 | 6 | 15 | 27 | 39 | -12 | 30 | 1.03 | 36.9 | 35.7 | +1.3 | +0.04 | L D W L W | 62,459 | Saïd Benrahma, Jarrod Bowen - 4 | Łukasz Fabiański | |
| 15 | Bournemouth | 30 | 8 | 6 | 16 | 28 | 57 | -29 | 30 | 1.00 | 29.9 | 51.8 | -22.0 | -0.73 | W L W L W | 10,359 | Philip Billing - 7 | Neto | |
| 16 | Leeds United | 30 | 7 | 8 | 15 | 39 | 54 | -15 | 29 | 0.97 | 37.5 | 49.5 | -12.0 | -0.40 | D W L W L | 36,507 | Rodrigo - 11 | Illan Meslier | |
| 17 | Everton | 30 | 6 | 9 | 15 | 23 | 43 | -20 | 27 | 0.90 | 32.5 | 51.7 | -19.2 | -0.64 | D W D D L | 39,237 | Demarai Gray - 4 | Jordan Pickford | |
| ▼ 18 | Nott'ham Forest | 30 | 6 | 9 | 15 | 24 | 54 | -30 | 27 | 0.90 | 30.5 | 47.7 | -17.1 | -0.57 | L L D L L | 29,177 | Brennan Johnson - 8 | Dean Henderson | |
| ▼ 19 | Leicester City | 30 | 7 | 4 | 19 | 40 | 52 | -12 | 25 | 0.83 | 35.6 | 49.1 | -13.5 | -0.45 | L D L L L | 31,815 | Harvey Barnes - 10 | Danny Ward | |
| ▼ 20 | Southampton | 30 | 6 | 5 | 19 | 24 | 51 | -27 | 23 | 0.77 | 28.5 | 42.7 | -14.3 | -0.48 | D L D L L | 30,502 | James Ward-Prowse - 7 | Gavin Bazunu | |

(Figure 3.1)

For this project I will be scraping data from the FBref website [6]. FBref is a website that provides comprehensive data on fooball matches, including detailed statistics on teams, players, and individual matches. One of the key advantages of using FBref for machine learning is the vast amount of data that the website provides. With detailed statistics on hundreds of teams and thousands of players, FBref provides a wealth of data that can be used to train and test machine learning models. Additionally, the data is structured and organized, making it easier to pre-process and prepare the data for use in machine learning models.

FBref also provides access to historical data, which can be used to train machine learning models and test their accuracy. By training models on historical data and testing their predictions against actual outcomes, researchers and practitioners can evaluate the effectiveness of their models and refine their approach over time.

Finally, FBref is user-friendly and easy to navigate, making it easy to access and collect the data needed for machine learning. With an intuitive interface and powerful search tools, researchers and practitioners can quickly find and collect the data they need to build and test machine learning models.

To extract this data from FBref I will be using requests. Requests is a popular Python library used to send HTTP/1.1 requests easily. It allows you to interact with web services and retrieve data from websites by making requests to their APIs or scraping data from their HTML pages. One of the main benefits of using Requests is its ease of use. With just a few lines of code, you can send a request and receive a response from a web server. Requests also makes it easy to customize requests with headers, authentication, and other parameters to fit your needs.

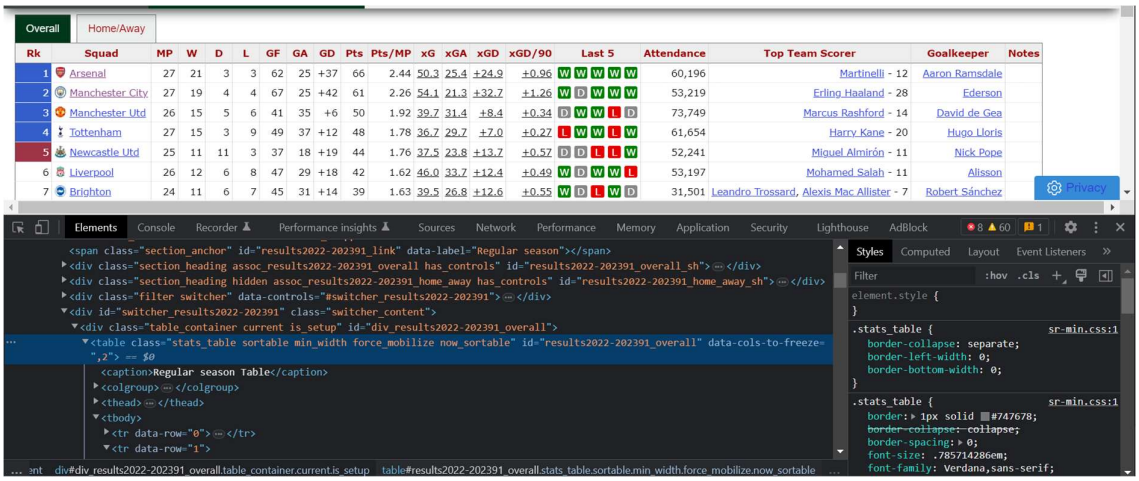
```
import requests
standings_url = "https://fbref.com/en/comps/9/Premier-League-Stats"
data = requests.get(standings_url)
```

Here is an example of one of many ways I will use requests, in this case using a GET request to the premier league statistics page and storing the response in the ‘data’ variable.

### 4.2. Parsing match data with beautiful soup

As well as using requests I have opted to also using beautiful soup. Beautiful Soup is a separate Python library that serves a different purpose, but together they can be used effectively for web scraping. Beautiful Soup is a Python library that allows you to parse and extract data from HTML and XML files. It is a powerful tool for web scraping and can be used to extract data from websites for use in machine learning models. One of the main benefits of using Beautiful Soup is its simplicity. It provides a simple and easy-to-use API for parsing HTML and XML files and extracting data from them. You can easily navigate the HTML structure of a web page and extract specific elements or data using a variety of methods such as ‘find ()’, ‘find_all()’, ‘select()’, and more.

Requests can be used to send an HTTP request to a webpage and get its HTML content, and Beautiful Soup can be used to parse and extract specific data from the HTML. This makes the process of web scraping much easier and more efficient.



(Figure 3.2)

Before I could use any of the beautiful soup commands, I had to inspect on the href website to filter out the data needed for my code.

```
standings_table = soup.select('table.stats_table')[0]
links = standings_table.find_all('a')
links = [l.get("href") for l in links]
links = [l for l in links if '/squads/' in l]
team_urls = [f"https://fbref.com{l}" for l in links]
team_urls
```

```
✓ 1.0s

'https://fbref.com/en/squads/18bb7c10/Arsenal-Stats',
'https://fbref.com/en/squads/b8fd03ef/Manchester-City-Stats',
'https://fbref.com/en/squads/19538871/Manchester-United-Stats',
'https://fbref.com/en/squads/361ca564/Tottenham-Hotspur-Stats',
'https://fbref.com/en/squads/b2b47a98/Newcastle-United-Stats',
'https://fbref.com/en/squads/822bd0ba/Liverpool-Stats',
```

After using the requests.Get() method to send an HTTP GET request to the URL, in order for it to retrieve the HTML content of the webpage. Beautiful soup was imported and then the code uses the soup.select() method to find the first HTML table element on the page that has a class of stats_table. This HTML table contains the data that we want to extract from the webpage. The find_all() method is then used to find all the HTML anchor elements ('a') inside the standings_table variable, which represent hyperlinks to other pages.

Finally, the code uses a list comprehension to iterate over the links list and extract the href attribute of each anchor element. The href attribute contains the URL of the page that the hyperlink points to. The resulting list contains only the URLs that are present in the href attribute of each anchor element. In summary, this code demonstrates how to use the Requests and Beautiful Soup libraries to scrape data from a webpage. It retrieves the HTML content of a webpage, parses it using Beautiful Soup, and extracts all the URLs present in the hyperlinks of a specific HTML table on the page.

## 4.3. Extracting match stats with pandas

To extract important data, I have opted to use pandas. Pandas is a powerful library for data manipulation and analysis in Python. It provides a wide range of functions and methods for reading, cleaning, transforming, and analysing data, which can make the data extraction and processing steps much easier and more efficient. For example, pandas have functions for reading data from various sources, including HTML tables, CSV files, Excel spreadsheets, SQL databases, and more. It also has functions for cleaning and transforming data, such as removing missing values, filtering rows and columns, aggregating data, and merging datasets.

Pandas also provides a flexible and intuitive data structure called a Data Frame, which is similar to a table or spreadsheet. The Data Frame allows for easy manipulation and analysis of data, including indexing, slicing, filtering, sorting, grouping, and more. This can make it easier to extract the relevant information from the scraped data and transform it into a format that is suitable for machine learning. For example, it can make difficult to read html such as:

```
data = requests.get(team_url)
data.text
```

✓ 1.3s Python

```
'    \n      \n<!DOCTYPE html>\n<html data-version="klecko-" data-root="/home/fb/deploy/www/base" lang="en" class="no-js" >\n<head>\n    <meta
charset="utf-8">\n    <meta http-equiv="x-ua-compatible" content="ie=edge">\n    <meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=2.0" />\n    <link rel="dns-prefetch" href="https://cdn.ssref.net/req/202303021" />\n<!-- Quantcast Choice. Consent Manager Tag v2.0
(for TCF 2.0) -->\n<script type="text/javascript" async=true>\n    (function() {\n\tvar host = window.location.hostname;\n\tvar element =
document.createElement(\'script\');\n\tvar firstScript = document.getElementsByTagName(\'script\')[0];\n\tvar url = \'https://cmp.quantcast.com\'\n\t
.concat(\'/choice/\', \'XwNYEpNeFfhfr\', \'/\', host, \n\t\t    \'/choice.js?tag_version=V2\');\n\tvar uspTries = 0;\n\tvar uspTriesLimit =
```

And parse it to make it easier understand as shown here:

```
import pandas as pd
matches = pd.read_html(data.text, match="Scores & Fixtures ")
matches
```

✓ 2.8s

Output exceeds the size limit. Open the full output data in a text editor

```
[          Date   Time            Comp               Round  Day Venue Result
 0   2022-08-05  20:00  Premier League          Matchweek 1  Fri  Away      W
 1   2022-08-13  15:00  Premier League          Matchweek 2  Sat  Home      W
 2   2022-08-20  17:30  Premier League          Matchweek 3  Sat  Away      W
 3   2022-08-27  17:30  Premier League          Matchweek 4  Sat  Home      W
 4   2022-08-31  19:30  Premier League          Matchweek 5  Wed  Home      W
```

Additionally, pandas can handle different types of data, including numerical, categorical, and textual data, which can be useful for analysing and predicting different aspects of match stats.

After successfully extracting and parsing the data using requests and pandas, we now have easy and clear table as shown above. In this table we have all the match data from the current 2022/23 season for each club that was located in the ‘Scores & Fixtures’ table on FBref . This includes data such as the date, time, round, result etc. Collecting these types of data is important for machine learning data when predicting Premier League results for several reasons:

Temporal dynamics: The date of the match can provide important temporal context to the data, which can help capture trends and patterns over time. For example, a team's performance at the beginning of the season may differ from their performance towards the end of the season. By including the date of the match in the dataset, machine learning models can account for these temporal dynamics and adjust their predictions accordingly.

Home and away advantage: Home and away advantage can have a significant impact on the outcome of a match. Home teams tend to perform better due to factors such as familiar surroundings, crowd support, and lack of travel fatigue. By including the information on whether a team was playing at home or away in the dataset, machine learning models can capture the impact of these factors and adjust their predictions accordingly.

Outcome variable: The result of the match (e.g., win, draw, or loss) is the primary outcome variable in predicting Premier League results. By including this information in the dataset, machine learning models can learn from past matches and use the information to make predictions about future matches.

Feature engineering: The information extracted can also be used as features in machine learning models to improve their accuracy. For example, feature engineering techniques can be used to create new features from the existing data that capture important relationships between the variables. This can help improve the accuracy of the model and make more accurate predictions.

## 4.4. Extracting shooting stats with pandas

Much like the match stats I also opted to extracting shooting stats as well. These include Shots taken, shots on target, expected goals, penalty kicks and many more. Adding these shooting stats will be important for my model as well as the match stats firstly because it will lead to better prediction accuracy. Shooting stats provide insights into a team's attacking ability and their potential to score goals. This information can be combined with other match stats to create a more comprehensive view of team performance. Collecting shooting stats will also help identify the stronger teams. A team can lose a game despite being the overall better team so just taking down match results can lead to incorrect analysis. For example, a team with a high number of shots on target and a high expected goals value suggests that they are strong in attack and may have a higher chance of winning matches. Conversely, a team with a low number of shots on target and a low expected goals value suggests that they struggle to create chances and may be weaker in attack.

To obtain these stats I will use the same method I used to extract the match stats. However, this time instead of using `links = [l for l in links if '/squads/' in l]` with header `matches = pd.read_html(data.text, match="Scores & Fixtures ")` I will be using `links = [l for l in links if l and 'all_comps/shooting/' in l]` with header `shooting = pd.read_html(data.text, match="Shooting")[0]`. To obtain the shooting stats shown:

```
shooting = pd.read_html(data.text, match="Shooting")[0]
shooting.head()
```

✓ 2.8s Python

| | For Arsenal | | | | | | | | | | ... | Standard | | | | Expected | | | | | Unnamed: 25_level_0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Date | Time | Comp | Round | Day | Venue | Result | GF | GA | Opponent | ... | Dist | FK | PK | PKatt | xG | npxG | npxG/Sh | G-xG | np:G-xG | Match Report |
| 0 | 2022-08-05 | 20:00 | Premier League | Matchweek 1 | Fri | Away | W | 2 | 0 | Crystal Palace | ... | 14.6 | 1.0 | 0 | 0 | 1.0 | 1.0 | 0.10 | 0.0 | 0.0 | Match Report |
| 1 | 2022-08-13 | 15:00 | Premier League | Matchweek 2 | Sat | Home | W | 4 | 2 | Leicester City | ... | 13.0 | 0.0 | 0 | 0 | 2.7 | 2.7 | 0.16 | 1.3 | 1.3 | Match Report |

## 4.5. Cleaning and merging scraped data

Both match and shooting data have been extracted and will now have to be merged. To do this I used the '. merge()' method of the pandas library. The merge() method combines two Data Frames based on one or more common columns. In this case, you specified the common column to be "Date", which means that the method will merge the two Data Frames based on the values in their "Date" column.

```
team_data = matches[0].merge(shooting[["Date","Sh","SoT","Dist","FK","PK","PKatt"]], on="Date")
team_data.head()
```

✓ 3.4s Python

| | Date | Time | Comp | Round | Day | Venue | Result | GF | GA | Opponent | ... | Formation | Referee | Match Report | Notes | Sh | SoT | Dist | FK | PK | PKatt |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2022-08-05 | 20:00 | Premier League | Matchweek 1 | Fri | Away | W | 2.0 | 0.0 | Crystal Palace | ... | 4-3-3 | Anthony Taylor | Match Report | NaN | 10 | 2 | 14.6 | 1.0 | 0 | 0 |
| 1 | 2022-08-13 | 15:00 | Premier League | Matchweek 2 | Sat | Home | W | 4.0 | 2.0 | Leicester City | ... | 4-3-3 | Darren England | Match Report | NaN | 19 | 7 | 13.0 | 0.0 | 0 | 0 |

I also cleaned my data here as shown from the list of shooting stats I used. This was because if I added all shooting columns data such as xG would overlap with the match data. Overlapping data can cause a problem because it can introduce bias into my analysis and modelling. When you have overlapping data, you essentially have duplicate or very similar observations in your dataset, which can skew your results and make your analysis less reliable. I also opted to certain shooting stats and not all due to some for example non penalty expected goals seeming redundant in comparison to other more important stats.

## 4.6. Scarping data from multiple seasons

The next thing I need to do is scale my method up and scrape data for every team from multiple years. This is important because it increases my sample size, which can improve the accuracy and reliability of my model. A larger data size can help capture more patterns and trends in the data that can lead to higher predictive power. Premier league matches can also change over time dure to various factors such as changes in team strategies, player transfers and rule changes. scraping data from multiple seasons can help capture these changes and improve accuracy.

To capture these seasons, I will be using a loop. This loop will iterate through years 2023 to 2017 (in reverse order).

```
years = list(range(2023, 2017, -1))
years
[2023, 2022, 2021, 2020, 2019, 2018]
```

I chose these years because they're the most recent seasons. At first, I opted to use just the 2020/21 season and the 2019/2020 seasons

for my data, however during those seasons many changes occurred. This was due to COVID-19 pandemic, which had an impact on the scheduling and dynamics of the league. Due to this I went even further back and captured more season, despite my code taking longer the results should be a lot more accurate.

```
for year in years:
    data = requests.get(standings_url)
    soup = BeautifulSoup(data.text)
    standings_table = soup.select('table.stats_table')[0]

    links = [l.get("href") for l in standings_table.find_all('a')]
    links = [l for l in links if '/squads/' in l]
    team_urls = [f"https://fbref.com{l}" for l in links]

    previous_season = soup.select("a.prev")[0].get("href")
    standings_url = f"https://fbref.com{previous_season}"
```

The for loop used first sends a request to the standings_url and retrieves the HTML content using the request library. It then uses beautiful soup library to parse the HTML and extract the first table with class “stats_table”. It then finds all the links within the table and selects only those that contain the string “/squads/” and creates a list of full URL’S for each team. Next, it retrieves the previous season’s standings_url by selecting the first link with the class “prev” and appending it.

Then with the use of another for loop, for each URL, it retrieves the HTML content ansd reads the table with the string “scores & fixtures” and then the same for the “all_comps/shooting/” string. The merging strategy I used before is then repeated and all the data is extracted.

In my code you will notice the use of ‘time.sleep()’ function. I used this to pause for 5 seconds between each loop iteration, this is included because I wanted to avoid overloading the website with excessive amounts of requests too quickly.

## 4.7. Data collected.

```
match_df = pd.concat(all_matches)
match_df.columns = [c.lower() for c in match_df.columns]
match_df.to_csv("premierleague.csv")
match_df
```

After concatenating and saving all my data scraped I am left with 5 seasons of data including match stats such as result, venue and teams playing and shooting stats which include, shots taken, expected goals,etc.

| | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2022-08-20 | 17:30 | Premier League | Matchweek 3 | Sat | Away | W | 3 | 0 | Bournemouth | ... | Match Report | NaN | 14.0 | 6.0 | 14.8 | 0.0 | 0.0 | 0.0 | 2023 | Arsenal |
| 3 | 2022-08-27 | 17:30 | Premier League | Matchweek 4 | Sat | Home | W | 2 | 1 | Fulham | ... | Match Report | NaN | 22.0 | 8.0 | 15.5 | 1.0 | 0.0 | 0.0 | 2023 | Arsenal |
| 4 | 2022-08-31 | 19:30 | Premier League | Matchweek 5 | Wed | Home | W | 2 | 1 | Aston Villa | ... | Match Report | NaN | 22.0 | 8.0 | 16.3 | 1.0 | 0.0 | 0.0 | 2023 | Arsenal |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 38 | 2018-04-15 | 16:00 | Premier League | Matchweek 34 | Sun | Away | W | 1 | 0 | Manchester Utd | ... | Match Report | NaN | 10.0 | 4.0 | 18.1 | 0.0 | 0.0 | 0.0 | 2018 | West Bromwich Albion |
| 39 | 2018-04-21 | 12:30 | Premier League | Matchweek 35 | Sat | Home | D | 2 | 2 | Liverpool | ... | Match Report | NaN | 13.0 | 6.0 | 17.7 | 0.0 | 0.0 | 0.0 | 2018 | West Bromwich Albion |
| 40 | 2018-04-28 | 15:00 | Premier League | Matchweek 36 | Sat | Away | W | 1 | 0 | Newcastle Utd | ... | Match Report | NaN | 9.0 | 2.0 | 20.1 | 0.0 | 0.0 | 0.0 | 2018 | West Bromwich Albion |
| 41 | 2018-05-05 | 15:00 | Premier League | Matchweek 37 | Sat | Home | W | 1 | 0 | Tottenham | ... | Match Report | NaN | 9.0 | 1.0 | 10.2 | 0.0 | 0.0 | 0.0 | 2018 | West Bromwich Albion |
| 42 | 2018-05-13 | 15:00 | Premier League | Matchweek 38 | Sun | Away | L | 0 | 2 | Crystal Palace | ... | Match Report | NaN | 7.0 | 1.0 | 24.8 | 1.0 | 0.0 | 0.0 | 2018 | West Bromwich Albion |

4340 rows × 27 columns

(Figure 3.3)

# 5. Implementation

In the upcoming chapter, I will delve into the implementation of my highly intricate and advanced machine learning model, using a diverse range of cutting-edge techniques. My primary objective is to explore effective methods of deploying the model to achieve optimal performance. I will investigate various strategies for enhancing model accuracy, including fine-tuning hyperparameters, predictors, and employing ensemble methods.

## 5.1. Pandas

I started by importing pandas into my code. Pandas is a high-performance data analysis library that makes it easy to work with data frames, arrays, and lists. Data frames are great way to store and analyse your data, because they let you group related data together in a tabular format. The data I will be using is in a csv file of previous premier league season fixtures holding the result, venue (whether its home/away), goals, shots, xg, etc. With Pandas, you can import data from a wide variety of sources, organize your data by columns, analyse your data with a variety of functions, and present your results in a meaningful way. In the beginning will be using a csv file I downloaded online containing only 2022/21 and 2021/20 seasons.

Example of how the data is saved and stored:

```python
import pandas as pd #importing pandas library
matches = pd.read_csv("plmatches.csv", index_col=0) #reading in match data
matches.head()
```

✓ 2.8s Python

| | date | time | comp | round | day | venue | result | gf | ga | opponent | ... | match report | notes | sh | sot | dist | fk | pk | pkatt | season | team |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2021-08-15 | 16:30 | Premier League | Matchweek 1 | Sun | Away | L | 0.0 | 1.0 | Tottenham | ... | Match Report | NaN | 18.0 | 4.0 | 16.9 | 1.0 | 0.0 | 0.0 | 2022 | Manchester City |
| 2 | 2021-08-21 | 15:00 | Premier League | Matchweek 2 | Sat | Home | W | 5.0 | 0.0 | Norwich City | ... | Match Report | NaN | 16.0 | 4.0 | 17.3 | 1.0 | 0.0 | 0.0 | 2022 | Manchester City |
| 3 | 2021-08-28 | 12:30 | Premier League | Matchweek 3 | Sat | Home | W | 5.0 | 0.0 | Arsenal | ... | Match Report | NaN | 25.0 | 10.0 | 14.3 | 0.0 | 0.0 | 0.0 | 2022 | Manchester City |
| 4 | 2021-09-11 | 15:00 | Premier League | Matchweek 4 | Sat | Away | W | 1.0 | 0.0 | Leicester City | ... | Match Report | NaN | 25.0 | 8.0 | 14.0 | 0.0 | 0.0 | 0.0 | 2022 | Manchester City |
| 6 | 2021-09-18 | 15:00 | Premier League | Matchweek 5 | Sat | Home | D | 0.0 | 0.0 | Southampton | ... | Match Report | NaN | 16.0 | 1.0 | 15.7 | 1.0 | 0.0 | 0.0 | 2022 | Manchester City |

5 rows × 27 columns

## 5.2. Cleaning Data

Cleaning data is a vital step in any machine learning project. The ability to clean data is key in successfully applying machine learning algorithms. There are several steps involved in the cleaning process. Some of these include removing outliers, missing data, data formatting, and normalizing the data. First I checked for any missing data in the 2 season of data I used at the start:

```python
import pandas as pd #importing pandas library
matches = pd.read_csv("plmatches.csv", index_col=0) #reading in match data
matches.shape
```

✓ 0.7s

```
(1389, 27)
```

We can see there are 1389 rows of data meaning 1389 matches recorded. Here lies a problem as there are 38 games in a season, 20 teams and 2 seasons altogether meaning there should be 38*20*2= 1520 rows. I will improve on this by adding more seasons of data as well as more recent data.

Other ways of cleaning will also be implemented. Machine learning algorithms can only work with numeric data, this would mean the data that is in float64 or int64. All the data types from each of the categories of stored data:

```
date            object
time            object
comp            object
round           object
day             object
venue           object
result          object
gf             float64
ga             float64
opponent        object
xg             float64
xga            float64
poss           float64
attendance     float64
captain         object
formation       object
referee         object
match report    object
notes          float64
sh             float64
sot            float64
dist           float64
fk             float64
pk             float64
```

```
pkatt          float64
season           int64
team            object
```

(Figure4.1)

The algorithm cant work with data that is an object. So I will need to convert useful data into these data types so they can be used as predictors. For example I changed the date datatype as shown here:

```
matches["date"] = pd.to_datetime(matches["date"])
```

```
date           datetime64[ns]
```

## 5.3. Predictors

Prediction is the process of making an educated guess about the outcome of future events based on the analysis of past events (Wikipedia). One of the main components of making predictions with machine learning is the predictor itself. In order to make good predictions, it is important for us to choose the right predictors. There are many different predictors to choose from, each with its own advantages and disadvantages.

The first predictor I chose to create was based on if the team was playing home or away. The effect of playing home and away in football has been debated for many years. It has been suggested that playing away from home harms team morale and can cause players to lose confidence. According to footystats.org the home win % in the premier league is 45% in the ongoing 22/23 season whereas the away win % is only 32% [8]. When creating this predictor, we notice the category venue which decided if the team was home or away is listed as an object for its data type. A simple fix was to create a new category called venue_code which took each matches venue and gave a code 0 and 1 for away and home respectively.

```
import pandas as pd #importing pandas library
matches = pd.read_csv("plmatches.csv", index_col=0) #reading in match data
matches["date"] = pd.to_datetime(matches["date"])#c
matches["venue_code"] = matches["venue"].astype("category").cat.codes
matches
```

✓ 0.1s Python

| | date | time | comp | round | day | venue | result | gf | ga | opponent | ... | notes | sh | sot | dist | fk | pk | pkatt | season | team | venue_code |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2021-08-15 | 16:30 | Premier League | Matchweek 1 | Sun | Away | L | 0.0 | 1.0 | Tottenham | ... | NaN | 18.0 | 4.0 | 16.9 | 1.0 | 0.0 | 0.0 | 2022 | Manchester City | 0 |
| 2 | 2021-08-21 | 15:00 | Premier League | Matchweek 2 | Sat | Home | W | 5.0 | 0.0 | Norwich City | ... | NaN | 16.0 | 4.0 | 17.3 | 1.0 | 0.0 | 0.0 | 2022 | Manchester City | 1 |
| 3 | 2021-08-28 | 12:30 | Premier League | Matchweek 3 | Sat | Home | W | 5.0 | 0.0 | Arsenal | ... | NaN | 25.0 | 10.0 | 14.3 | 0.0 | 0.0 | 0.0 | 2022 | Manchester City | 1 |
| 4 | 2021-09-11 | 15:00 | Premier League | Matchweek 4 | Sat | Away | W | 1.0 | 0.0 | Leicester City | ... | NaN | 25.0 | 8.0 | 14.0 | 0.0 | 0.0 | 0.0 | 2022 | Manchester City | 0 |
| 6 | 2021-09-18 | 15:00 | Premier League | Matchweek 5 | Sat | Home | D | 0.0 | 0.0 | Southampton | ... | NaN | 16.0 | 1.0 | 15.7 | 1.0 | 0.0 | 0.0 | 2022 | Manchester City | 1 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 38 | 2021-05-02 | 19:15 | Premier League | Matchweek 34 | Sun | Away | L | 0.0 | 4.0 | Tottenham | ... | NaN | 8.0 | 1.0 | 17.4 | 0.0 | 0.0 | 0.0 | 2021 | Sheffield United | 0 |
| 39 | 2021- | 15:00 | Premier | Matchweek | Sat | Home | L | 0.0 | 2.0 | Crystal Palace | | NaN | 7.0 | 0.0 | 11.4 | 1.0 | 0.0 | 0.0 | 2021 | Sheffield | 1 |

Oppositions will obviously influence the match result, so I will also use this as a predictor. You are much more likely to lose to a team like Manchester City than Southampton especially if previous data against said opponents shows this. Like I did with the venue I will do the same for the opponent. Each team in the league will be allocated and number and put in the category opp_code.

```
import pandas as pd #importing pandas library
matches = pd.read_csv("plmatches.csv", index_col=0) #reading in match data
matches["date"] = pd.to_datetime(matches["date"])#c
matches["venue_code"] = matches["venue"].astype("category").cat.codes
matches["opp_code"] = matches["opponent"].astype("category").cat.codes
matches
```

✓ 0.7s Python

| date | time | comp | round | day | venue | result | gf | ga | opponent | ... | sh | sot | dist | fk | pk | pkatt | season | team | venue_code | opp_code |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2021-08-15 | 16:30 | Premier League | Matchweek 1 | Sun | Away | L | 0.0 | 1.0 | Tottenham | ... | 18.0 | 4.0 | 16.9 | 1.0 | 0.0 | 0.0 | 2022 | Manchester City | 0 | 18 |
| 2021-08-21 | 15:00 | Premier League | Matchweek 2 | Sat | Home | W | 5.0 | 0.0 | Norwich City | ... | 16.0 | 4.0 | 17.3 | 1.0 | 0.0 | 0.0 | 2022 | Manchester City | 1 | 15 |
| 2021-08-28 | 12:30 | Premier League | Matchweek 3 | Sat | Home | W | 5.0 | 0.0 | Arsenal | ... | 25.0 | 10.0 | 14.3 | 0.0 | 0.0 | 0.0 | 2022 | Manchester City | 1 | 0 |
| 2021-09-11 | 15:00 | Premier League | Matchweek 4 | Sat | Away | W | 1.0 | 0.0 | Leicester City | ... | 25.0 | 8.0 | 14.0 | 0.0 | 0.0 | 0.0 | 2022 | Manchester City | 0 | 10 |
| 2021-09-18 | 15:00 | Premier League | Matchweek 5 | Sat | Home | D | 0.0 | 0.0 | Southampton | ... | 16.0 | 1.0 | 15.7 | 1.0 | 0.0 | 0.0 | 2022 | Manchester City | 1 | 17 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 2021-05-02 | 19:15 | Premier League | Matchweek 34 | Sun | Away | L | 0.0 | 4.0 | Tottenham | ... | 8.0 | 1.0 | 17.4 | 0.0 | 0.0 | 0.0 | 2021 | Sheffield United | 0 | 18 |

-we can see here Tottenham have opp_code = 18

For the start of my model, I have used the day_code, and hour as my final two predictors to get it started as I improve my model, I intend to add more efficient predictors.

Result of the matches W, L and D would also need to be changed to integers. We call these our targets. This is because before we start to train our data, we need a target we are going to try and predict. This will be very important when testing the precision and accuracy of our model.

```
matches["target"] = matches["result"].astype("category").cat.codes#target changes to 0,1,2 whether team draws,losses or wins respectively
matches
```

✓ 0.2s Python

| time | comp | round | day | venue | result | gf | ga | opponent | ... | fk | pk | pkatt | season | team | venue_code | opp_code | hour | day_code | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 16:30 | Premier League | Matchweek 1 | Sun | Away | L | 0.0 | 1.0 | Tottenham | ... | 1.0 | 0.0 | 0.0 | 2022 | Manchester City | 0 | 18 | 16 | 6 | 1 |
| 15:00 | Premier League | Matchweek 2 | Sat | Home | W | 5.0 | 0.0 | Norwich City | ... | 1.0 | 0.0 | 0.0 | 2022 | Manchester City | 1 | 15 | 15 | 5 | 2 |
| 12:30 | Premier League | Matchweek 3 | Sat | Home | W | 5.0 | 0.0 | Arsenal | ... | 0.0 | 0.0 | 0.0 | 2022 | Manchester City | 1 | 0 | 12 | 5 | 2 |
| 15:00 | Premier League | Matchweek 4 | Sat | Away | W | 1.0 | 0.0 | Leicester City | ... | 0.0 | 0.0 | 0.0 | 2022 | Manchester City | 0 | 10 | 15 | 5 | 2 |
| 15:00 | Premier League | Matchweek 5 | Sat | Home | D | 0.0 | 0.0 | Southampton | ... | 1.0 | 0.0 | 0.0 | 2022 | Manchester City | 1 | 17 | 15 | 5 | 0 |

We have W, L and D as 2,1 and 0 respectively.

Predictors such as attendance and formation would not make sense to use as they are recorded on the day of the fixture therefore predicting with them would negligible.

## 5.4. Initial Machine Learning Model

To start with I imported a model called Random Forest Classifier. The Random Forest Classifier is a machine learning algorithm that can be used to solve a wide range of classification problems. It can also be used to classify different types of data, such as text documents or images, based on a set of predefined categories. The model consists of several decision trees, each of which performs a specific task using the input data to generate an outcome. These trees are connected to form a forest, which is used to create the final prediction.

In order to import the Random Forest, I used the python library sklearn or scikit-learn. Scikit-learn is probably the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality. (Analyticsvidhya)

For this model I will need to split my data in training and test sets. A training set is implemented to build up a model, while a test set is to validate the model built. Data points in the training set are excluded from the test set, this data is time sensitive I need to ensure that the test set takes place after the training set. In machine learning, we basically try to create a model to predict the test data. So, we use the training data to fit the model and testing data to test it. The models generated are to predict the results unknown which is named as the test set. We do this to in order to check accuracies, precisions by training and testing it on it.

My predictors are then applied, and I fit in my Random Forest model. Essentially, I will be training the Random Forest model using my predictors in order to predict my target. The target being as stated before 2, 1 and 0 for win, lose and draw respectively.

```
rf.fit(train[predictors], train["target"])
```

For my initial machine learning model, I then predicted my accuracy:

```
from sklearn.metrics import accuracy_score#this
acc = accuracy_score(test["target"], preds)#tes
acc
```

✓ 0.1s

```
0.4384057971014493
```

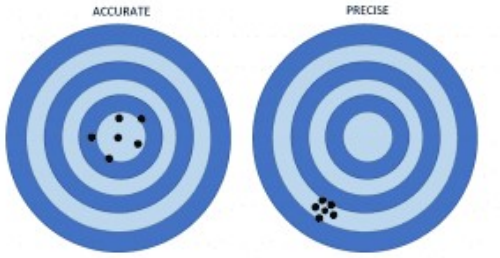
My current accuracy is not very good. I will need to add more predictors as well as finding more data (adding more seasons to my csv file). As I retrain my model and improve upon it the accuracy should increase.

By creating a data frame, I can delve into how close my predictions were for each result:

| prediction | 0 | 1 | 2 |
|---|---|---|---|
| actual | | | |
| 0 | 9 | 21 | 26 |
| 1 | 29 | 62 | 25 |
| 2 | 18 | 36 | 50 |

We can see 9 times we predicted a draw right, 62 times we predicted a loss right and 50 times we predicted a win right. So, the draws are the least accurate.

Precision is another metric to base the model upon. This will tell us when I predicted a win what percentage of the time was it right. Precision is not to be mistaken by accuracy. Accuracy refers to how close a measurement is to the true or accepted value. Precision refers to how close measurements of the same item are to each other. [10]




(Figure 4.1)

However, when calculating precision, I encountered a problem.

```
ValueError: Target is multiclass but average='binary'. Please choose another average setting, one of [None, 'micro', 'macro', 'weighted'].
```

What happened was my model was producing 3 outcomes 2, 1 and 0, this caused a value error because the precision I'm trying to calculate requires binary outcomes. Rather than precision of a win lose or draw I needed to change my program to check the precision of a win to not winning at all (draw and loss) . The solution was simply to change my outcomes from 2, 1, 0 from win, lose, draw respectively to 1, 0 for win and not win.

```
#matches["target"] = matches["result"].astype("category").cat.codes
matches["target"] = (matches["result"] =="W").astype("int")
```

Precision of my model: `0.4745762711864407` A team is predicted to win right 47.5% of the time

This precision is not very good but I intend to improve on it.

## 5.5. Rolling Averages

A rolling average is an average calculated as a weighted average of past data. In other words, it averages the latest values in a series with a previous value in order to come up with a current value. I will incorporate these rolling averages into my model to enhance my predictions precision. To do this I will need to split my data into their respective teams. This is so we can see the teams matches prior to the game we are trying to predict so the model adjusts according to form. If a team losses 2 matches in a row prior to the prediction, chances are they may lose again. The predictors I'll be using for the rolling averages are `["gf","ga","sh","sot","dist","fk","pk","pkatt"]`. Variables such as gf (goals forward) and ga (goals against0 are very good averages to use as predictors before the game. It shows how creative and successful the team is going forward as well

as how stable they are in defence in the previous games. These will be very useful when calculating the prediction for the coming match.

```
rolling_averages(group, cols, new_cols)
matches_rolling = matches.groupby("team").apply(lambda x: rolling_averages(x, cols,new_cols))#we made rolling averages for every team and their matc
matches_rolling
```

✓ 0.3s Python

| team | | date | time | comp | round | day | venue | result | gf | ga | opponent | ... | day_code | target | gf_rolling | ga_rolling | sh_rolling | sot_rolling |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Arsenal | 6 | 2020-10-04 | 14:00 | Premier League | Matchweek 4 | Sun | Home | W | 2.0 | 1.0 | Sheffield Utd | ... | 6 | 2 | 2.000000 | 1.333333 | 7.666667 | 3.666667 |
| | 7 | 2020-10-17 | 17:30 | Premier League | Matchweek 5 | Sat | Away | L | 0.0 | 1.0 | Manchester City | ... | 5 | 1 | 1.666667 | 1.666667 | 5.333333 | 3.666667 |
| | 9 | 2020-10-25 | 19:15 | Premier League | Matchweek 6 | Sun | Home | L | 0.0 | 1.0 | Leicester City | ... | 6 | 1 | 1.000000 | 1.666667 | 7.000000 | 3.666667 |
| | 11 | 2020-11-01 | 16:30 | Premier League | Matchweek 7 | Sun | Away | W | 1.0 | 0.0 | Manchester Utd | ... | 6 | 2 | 0.666667 | 1.000000 | 9.666667 | 4.000000 |
| | 13 | 2020-11-08 | 19:15 | Premier League | Matchweek 8 | Sun | Home | L | 0.0 | 3.0 | Aston Villa | ... | 6 | 1 | 0.333333 | 0.666667 | 9.666667 | 2.666667 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| olverhampton Wanderers | 32 | 2022-03-13 | 14:00 | Premier League | Matchweek 29 | Sun | Away | W | 1.0 | 0.0 | Everton | ... | 6 | 2 | 1.333333 | 1.000000 | 12.333333 | 3.666667 |
| | 33 | 2022-03-18 | 20:00 | Premier League | Matchweek 30 | Fri | Home | L | 2.0 | 3.0 | Leeds United | ... | 4 | 1 | 1.666667 | 0.666667 | 12.333333 | 4.333333 |

## 5.6. Improving Machine Learning model

### 5.6.1. Rolling averages

The goal of this project is to test the predictability of the league, in order to do this the model needs to be as precise as possible. By retraining my model with the new parameters, I have included the precision should increase. Retraining the model is very important for the success of the project. Like I did with the initial machine learning model I will use the training and test set and use the new rolling averages to enhance it. The new precision of my model is: `0.625` A team is now predicted to win right 62.5% of the time

From the rolling averages alone, the model has increased its precision by 15%. This is much better. Rolling averages are a statistical technique used to smooth out fluctuations in a dataset by taking the average of a specific number of consecutive data points. The rolling average can increase precision by reducing the impact of random noise and short-term variations in the data, revealing longer-term trends and patterns.

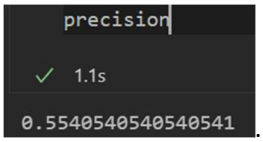
### 5.6.2. Larger data set

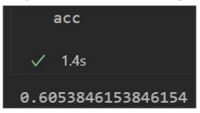
On the other hand, once I added my larger data with increased training and test sets:

```
train = data[data["date"] < '2021-08-12']
test = data[data["date"] > '2021-08-12']
```

training set -2017-2021

test set – 2021-2023

I noticed my precision had decreased from 0.625 to 

. This is because by adding more seasons of data I am introducing more noise to my data. Despite this change and increase in accuracy that was previously recorded at 0.438 has increased to 

which is a very positive increase. Adding data from additional seasons can help to capture any new or changing patterns that emerge over time, this helped improve my validity and reliability of my model. Even so I don't intend to add more seasons of data as including extreme amounts of data to my training and test sets can also increase the risk of overfitting and data leakage, which can decrease accuracy.

### 5.6.3. Predictors

```
Index(['date', 'time', 'comp', 'round', 'day', 'venue', 'result', 'gf', 'ga',
       'opponent', 'xg', 'xga', 'poss', 'attendance', 'captain', 'formation',
       'referee', 'match report', 'notes', 'sh', 'sot', 'dist', 'fk', 'pk',
       'pkatt', 'season', 'team', 'venue_code', 'opp_code', 'hour', 'day_code',
       'target'],
      dtype='object')
```

Additionally, I added possession and season to my predictors as well as xg and possession again to my rolling averages. Possession refers to the amount of time a team controls the ball during a game. By including possession data in my machine learning model, it can potentially provide insights into a team's offensive and defensive strategies, their overall performance, and their likelihood of winning.

Similarly, incorporating season data can also enhance my model's predictions. In sports, different seasons can have varying effects on team performance, due to factors such as weather conditions, player injuries, and changes in team dynamics. By including season data in your model, it can potentially identify patterns and trends that are specific to a particular season, which can be used to improve the accuracy of my predictions.

Expected goals (xG) is another metric that can be used to enhance my predictors and rolling averages. xG measures the likelihood of a shot resulting in a goal, based on factors such as shot distance, angle, and shot type. By incorporating xG data into my rolling averages, I can gain a more accurate assessment of a team's offensive performance, as it accounts for the quality of their scoring opportunities rather than just the quantity of shots taken.

To incorporate expected goals in to my predictors I had to first round each value into a whole number as my predictors can only take integers. `matches["expected goals"] = matches["xg"].round(0)`

My new predictors:

```
predictors = ["venue_code","opp_code","hour","day_code","poss","season","expected goals"]
```

### 5.6.4. Hyperparameter tuning.

Hyperparameter tuning is the process of optimizing the settings, or hyperparameters, of a machine learning algorithm in order to improve its performance. In the case of a random forest model for predicting Premier League matches, hyperparameter tuning can be used to improve the accuracy of your predictions by fine-tuning the model to the specific data and problem at hand. Random forest models have several hyperparameters that can significantly affect their performance, such as the number of trees in the forest, the depth of the trees, and the minimum number of samples required to split a node. The optimal values of these hyperparameters can depend on the specific data and problem at hand. Therefore, hyperparameter tuning is crucial to finding the optimal hyperparameters for your random forest model. This involves trying different combinations of hyperparameters, training and evaluating the model on a validation set, and selecting the combination of hyperparameters that results in the best performance.

I used the GridSearchCV function from the sklearn.model_selection module to perform hyperparameter tuning for my Random Forest Classifier model and created a variable that contains a dictionary of hyperparameters and their values to test. The hyperparameters being tested are:

```
param_grid = {
    'n_estimators': [50, 100, 150, 200],
    'min_samples_split': [2, 5, 10, 20, 40],
    'max_depth': [None, 10, 20, 30],
    'min_samples_leaf': [1, 2, 4, 8]
}
```

- n_estimators: The number of trees in the forest.
- min_samples_split: The minimum number of samples required to split an internal node.
- max_depth: The maximum depth of each tree.
- min_samples_leaf: The minimum number of samples required to be at a leaf node.

Once I had my parameters, I set my Random Forest Classifier to a random state value of 1 to ensure reproducibility of my results. This means each time my model is run, the same results will be obtained. A GridSearchCV function is then used to perform the tuning, it takes the classifier, hyperparameters with their values and a cross-validation of 5. To evaluate the performance of the model is used the accuracy as a scoring function. The best hyperparameters found by the function are then printed along with the overall accuracy of my model.

Results:

```
Best hyperparameters: {'max_depth': 20, 'min_samples_leaf': 8, 'min_samples_split': 2, 'n_estimators': 200}
Accuracy of best model: 0.7026813880126183
```

I added these parameters to my original model and as well as my accuracy increasing to my precision took suit also.

```
rf = RandomForestClassifier(max_depth=20, min_samples_leaf=8, min_samples_split=2, n_estimators=200, random_state=1 )
```

Results:

Accuracy: `0.7026813880126183` precision `0.6798866855524079`

# 6. Results

In this section, I will discuss the results of my study on the 2020/21 premier league season. I chose this season as it is the most recent completed season at this current date. Another reasons I chose to focus on the 2021/22 season was its unique context. The COVID-19 pandemic had disrupted the previous season, leading to an irregular schedule and reduced fan attendance. This had a significant impact on team performance and created several uncertainties for the following season. By predicting the outcome of the 2021/22 season, I was able to better understand the factors that contributed to team success in this unique context and provide insights that could inform future research and decision-making in the sport.

## 6.1. Accuracy and Precision

As stated in chapter 5.6.4 I ended with an accuracy of 70.2% and precision of 68%. My machine learning model has performed commendably in predicting the outcome of games in the Premier League for the 2021/2022 season. The accuracy rate of 70.2% is well above the chance level of 33.3% and indicates that my model was able to correctly predict the outcome of a match more than two-thirds of the time. This is a significant achievement, given the complex and unpredictable nature of football matches, and suggests that my model has the potential to offer valuable insights into game outcomes.

Precision is another essential metric for evaluating the performance of machine learning models. My model's precision rate of 67.99% implies that it was able to correctly predict the winning team nearly 68% of the time, which is quite impressive. This indicates that my model has a good ability to identify the team that is more likely to win a given game.

However, it is important to note that there are limitations to the predictability of my model, as even the most accurate model cannot account for unexpected events such as injuries, weather conditions, or sudden changes in team dynamics. These factors can significantly affect the outcome of a game and reduce the predictability of my model. Despite these limitations, the performance of my model suggests that it can be a valuable tool for predicting game outcomes in the Premier League. With further refinement and optimization, my model may be able to improve its accuracy even further and provide more accurate predictions.

It is essential to recognize the potential of machine learning in enhancing our understanding of sports, and in particular, football. As the field of sports analytics continues to evolve, models like mine can help us gain insights into the intricacies of athletic performance and improve our predictions of game outcomes. According to James and Gray (2019), machine learning has the potential to revolutionize sports analytics. Additionally, Guzman and Schwing (2018) developed a probabilistic model for football forecasting, demonstrating the importance of machine learning in sports analytics.

Overall, the results of my model are promising, and I am excited to see how it will develop and contribute to the advancement of sports analytics in the future.

## 6.2. Creating my Table and results

In chapter 1.3 one of the objectives stated was to create my own table using the predictions I created. To implement this, I will be using excel. As someone who works with data regularly, I have found that Microsoft Excel is an excellent tool for organizing and analysing data. With its powerful features and user-friendly interface, Excel allows me to quickly and easily manipulate and visualize data in a variety of ways. I have found Excel to be a convenient place to create my Premier League table, given that I had stored my data in a predictions.csv file on excel.

| | actual_x | predicted_x | date | team_x | opponent_ | result_x | new_team | actual_y | predicted_y | team_y | opponent_ | result_y | new_team_y |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 13/08/2021 | Arsenal | Brentford | L | Arsenal | 1 | 0 | Brentford | Arsenal | W | Brentford |
| 155 | 1 | 0 | 13/08/2021 | Brentford | Arsenal | W | Brentford | 0 | 0 | Arsenal | Brentford | L | Arsenal |
| 65 | 0 | 0 | 14/08/2021 | Aston Villa | Watford | L | Aston Villa | 1 | 0 | Watford | Aston Villa | W | Watford |
| 215 | 1 | 0 | 14/08/2021 | Brighton a | Burnley | W | Brighton | 0 | 1 | Burnley | Brighton | L | Burnley |
| 276 | 0 | 1 | 14/08/2021 | Burnley | Brighton | L | Burnley | 1 | 0 | Brighton a | Burnley | W | Brighton |
| 313 | 1 | 0 | 14/08/2021 | Chelsea | Crystal Pal | W | Chelsea | 0 | 0 | Crystal Pal | Chelsea | L | Crystal Palace |
| 376 | 0 | 0 | 14/08/2021 | Crystal Pal | Chelsea | L | Crystal Pal | 1 | 0 | Chelsea | Crystal Pal | W | Chelsea |
| 441 | 1 | 1 | 14/08/2021 | Everton | Southamp | W | Everton | 0 | 0 | Southamp | Everton | L | Southampton |
| 534 | 0 | 0 | 14/08/2021 | Leeds Unit | Manchest | L | Leeds Unit | 1 | 0 | Manchest | Leeds Unit | W | Manchester Utd |
| 597 | 1 | 0 | 14/08/2021 | Leicester C | Wolves | W | Leicester C | 0 | 0 | Wolverhar | Leicester C | L | Wolves |
| 659 | 1 | 1 | 14/08/2021 | Liverpool | Norwich C | W | Liverpool | 0 | 0 | Norwich C | Liverpool | L | Norwich City |
| 783 | 1 | 0 | 14/08/2021 | Manchest | Leeds Unit | W | Manchest | 0 | 0 | Leeds Unit | Manchest | L | Leeds United |
| 911 | 0 | 0 | 14/08/2021 | Norwich C | Liverpool | L | Norwich C | 1 | 1 | Liverpool | Norwich C | W | Liverpool |
| 946 | 0 | 0 | 14/08/2021 | Southamp | Everton | L | Southamp | 1 | 1 | Everton | Southamp | W | Everton |
| 1074 | 1 | 0 | 14/08/2021 | Watford | Aston Villa | W | Watford | 0 | 0 | Aston Villa | Watford | L | Aston Villa |
| 1170 | 0 | 0 | 14/08/2021 | Wolverhar | Leicester C | L | Wolves | 1 | 0 | Leicester C | Wolves | W | Leicester City |
| 720 | 0 | 1 | 15/08/2021 | Manchest | Tottenhan | L | Manchest | 1 | 0 | Tottenhan | Manchest | W | Tottenham |
| 847 | 0 | 1 | 15/08/2021 | Newcastle | West Ham | L | Newcastle | 1 | 1 | West Ham | Newcastle | W | West Ham |
| 1011 | 1 | 0 | 15/08/2021 | Tottenhan | Manchest | W | Tottenhan | 0 | 1 | Manchest | Tottenhan | L | Manchester City |
| 1108 | 1 | 1 | 15/08/2021 | West Ham | Newcastle | W | West Ham | 0 | 1 | Newcastle | West Ham | L | Newcastle Utd |
| 66 | 1 | 0 | 21/08/2021 | Aston Villa | Newcastle | W | Aston Villa | 0 | 0 | Newcastle | Aston Villa | L | Newcastle Utd |
| 156 | 0 | 0 | 21/08/2021 | Brentford | Crystal Pal | D | Brentford | 0 | 0 | Crystal Pal | Brentford | D | Crystal Palace |
| 216 | 1 | 0 | 21/08/2021 | Brighton a | Watford | W | Brighton | 0 | 0 | Watford | Brighton | L | Watford |

(Figure 5.1)

My predictions show a 1 team_x is predicted to win and a 0 if not, and vice versa for the away team team_y. Every match is considered as we can see Arsenal vs Brentford on the first row Arsenal is team_x and Brentford is team_y and the row below the opposite. For this experiment if both teams are predicted to not win, I will assume the program has predicted a draw and on the odd chance both are predicted to win I will predict a draw also. Along with my predicted data the actual result of each game is put along side them. This allows me to have a visual representation on the accuracy of my model.

With Excel, I can easily import my match results data and use functions such as COUNTIF and SUMIF to calculate the total number of wins, losses, and draws for each team. I can then use conditional formatting to highlight the teams with the most wins, the fewest losses.

For example, when creating my table is used:

```
=IF(AND(C2=1,J2=0),3,IF(AND(C2=0,J2=1),0,IF(AND(C2=0,J2=0),1,IF(AND(C2=1,J2=1),1,""))))
```

Here's what each of these conditions means:

Condition 1: The home team (in cell C2) won the game and the away team (in cell J2) lost.

Condition 2: The home team lost, and the away team won.

Condition 3: The game ended in a draw.

Condition 4: Neither condition 1 nor 2 nor 3 apply, i.e., one team won, and the other team also won (this is not possible in a soccer match, so it should never happen in this context).

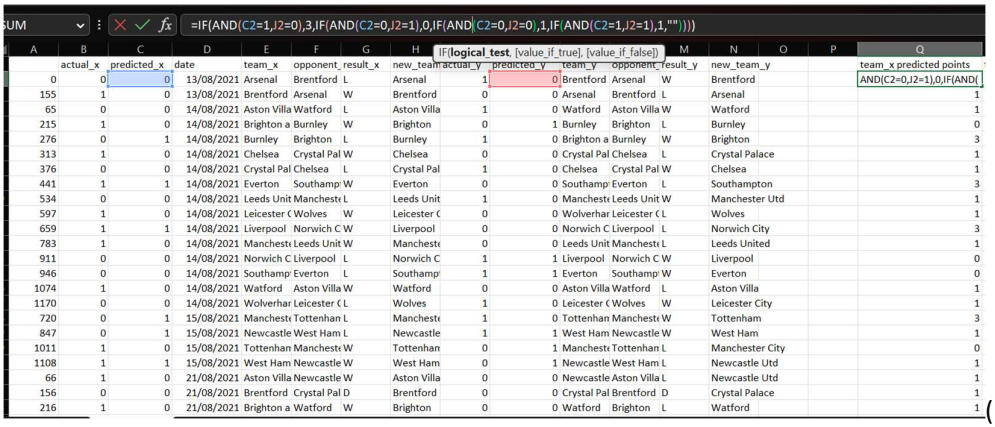
And here's what each of the return values means:

Return value if condition 1 is true: The home team won, so they get 3 points.

Return value if condition 2 is true: The home team lost, so they get 0 points.

Return value if condition 3 is true: The game ended in a draw, so each team gets 1 point.

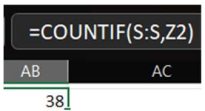
Return value if condition 4 is true: This should never happen, so the function returns a draw, so each team gets 1 point.




| | actual_x | predicted_x | date | team_x | opponent | result_x | new_team | actual_y | predicted_y | team_y | opponent | result_y | new_team_y | team_x predicted points |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 13/08/2021 | Arsenal | Brentford | L | Arsenal | 1 | 0 | Brentford | Arsenal | W | Brentford | AND(C2=0,J2=1),0,IF(AND( |
| 155 | 1 | 0 | 13/08/2021 | Brentford | Arsenal | W | Brentford | 0 | 0 | Arsenal | Brentford | L | Arsenal | 1 |
| 65 | 0 | 0 | 14/08/2021 | Aston Villa | Watford | L | Aston Villa | 1 | 0 | Watford | Aston Villa | W | Watford | 1 |
| 215 | 1 | 0 | 14/08/2021 | Brighton a | Burnley | W | Brighton | 0 | 1 | Burnley | Brighton | L | Burnley | 0 |
| 276 | 0 | 1 | 14/08/2021 | Burnley | Brighton | L | Burnley | 1 | 0 | Brighton a | Burnley | W | Brighton | 3 |
| 313 | 1 | 0 | 14/08/2021 | Chelsea | Crystal Pal | W | Chelsea | 0 | 0 | Crystal Pal | Chelsea | L | Crystal Palace | 1 |
| 376 | 0 | 0 | 14/08/2021 | Crystal Pal | Chelsea | L | Crystal Pal | 1 | 0 | Chelsea | Crystal Pal | W | Chelsea | 1 |
| 441 | 1 | 1 | 14/08/2021 | Everton | Southamp | W | Everton | 0 | 0 | Southamp | Everton | L | Southampton | 3 |
| 534 | 0 | 0 | 14/08/2021 | Leeds Unit | Manchest | L | Leeds Unit | 1 | 0 | Manchest | Leeds Unit | W | Manchester Utd | 1 |
| 597 | 1 | 0 | 14/08/2021 | Leicester C | Wolves | W | Leicester C | 0 | 0 | Wolverhar | Leicester C | L | Wolves | 1 |
| 659 | 1 | 1 | 14/08/2021 | Liverpool | Norwich C | W | Liverpool | 0 | 0 | Norwich C | Liverpool | L | Norwich City | 3 |
| 783 | 1 | 0 | 14/08/2021 | Manchest | Leeds Unit | W | Manchest | 0 | 0 | Leeds Unit | Manchest | L | Leeds United | 1 |
| 911 | 0 | 0 | 14/08/2021 | Norwich C | Liverpool | L | Norwich C | 1 | 1 | Liverpool | Norwich C | W | Liverpool | 0 |
| 946 | 0 | 0 | 14/08/2021 | Southamp | Everton | L | Southamp | 1 | 1 | Everton | Southamp | W | Everton | 0 |
| 1074 | 1 | 0 | 14/08/2021 | Watford | Aston Villa | W | Watford | 0 | 0 | Aston Villa | Watford | L | Aston Villa | 1 |
| 1170 | 0 | 0 | 14/08/2021 | Wolverhar | Leicester C | L | Wolves | 1 | 0 | Leicester C | Wolves | W | Leicester City | 1 |
| 720 | 0 | 1 | 15/08/2021 | Manchest | Tottenhan | L | Manchest | 1 | 0 | Tottenhan | Manchest | W | Tottenham | 3 |
| 847 | 0 | 1 | 15/08/2021 | Newcastle | West Ham | L | Newcastle | 1 | 1 | West Ham | Newcastle | W | West Ham | 1 |
| 1011 | 1 | 0 | 15/08/2021 | Tottenhan | Manchest | W | Tottenhan | 0 | 1 | Manchest | Tottenhan | L | Manchester City | 0 |
| 1108 | 1 | 1 | 15/08/2021 | West Ham | Newcastle | W | West Ham | 0 | 1 | Newcastle | West Ham | L | Newcastle Utd | 1 |
| 66 | 1 | 0 | 21/08/2021 | Aston Villa | Newcastle | W | Aston Villa | 0 | 0 | Newcastle | Aston Villa | L | Newcastle Utd | 1 |
| 156 | 0 | 0 | 21/08/2021 | Brentford | Crystal Pal | D | Brentford | 0 | 0 | Crystal Pal | Brentford | D | Crystal Palace | 1 |
| 216 | 1 | 0 | 21/08/2021 | Brighton a | Watford | W | Brighton | 0 | 0 | Watford | Brighton | L | Watford | 1 |

(Figure 5.2)

To create my table the `=SUMIF(S:S,Z2,Q:Q)` function was used to add all points for each team.




Alongside this I used the function to make sure each team had played 38 games to simulate the whole season.

My results:

| Club | Points | Games |
|---|---|---|
| Manchester City | 104 | 38 |
| Liverpool | 96 | 38 |
| Chelsea | 70 | 38 |
| Arsenal | 63 | 38 |
| Manchester United | 57 | 38 |
| Tottenham Hotspur | 57 | 38 |
| Brighton and Hove Albion | 47 | 38 |
| Newcastle United | 46 | 38 |
| Leicester City | 41 | 38 |
| Everton | 37 | 38 |
| Leeds United | 37 | 38 |
| Burnley | 40 | 38 |
| Watford | 38 | 38 |
| Crystal Palace | 35 | 38 |
| Aston Villa | 35 | 38 |
| Southampton | 32 | 38 |
| West Ham United | 31 | 38 |
| Wolverhampton Wanderers | 30 | 38 |
| Brentford | 29 | 38 |
| Norwich City | 22 | 38 |

(Figure 5.3)

## 6.3. Prediction Analysis

When we compare your machine learning model's predictions to the actual 2021/22 Premier League table, we can see that there are some similarities but also some significant differences.

| Club | Points | Club | Actual |
|---|---|---|---|
| Manchester City | 104 | Manchester City | 93 |
| Liverpool | 96 | Liverpool | 92 |
| Chelsea | 70 | Chelsea | 74 |
| Arsenal | 63 | Tottenham Hotspur | 71 |
| Manchester United | 57 | Arsenal | 69 |
| Tottenham Hotspur | 57 | Manchester United | 58 |
| Brighton and Hove Albion | 47 | West Ham United | 56 |
| Newcastle United | 46 | Leicester City | 52 |
| Leicester City | 41 | Wolverhampton Wanderers | 51 |
| Everton | 37 | Brighton and Hove Albion | 51 |
| Leeds United | 37 | Newcastle United | 49 |
| Burnley | 40 | Crystal Palace | 48 |
| Watford | 38 | Brentford | 46 |
| Crystal Palace | 35 | Aston Villa | 45 |
| Aston Villa | 35 | Southampton | 40 |
| Southampton | 32 | Everton | 39 |
| West Ham United | 31 | Leeds United | 38 |
| Wolverhampton Wanderers | 30 | Burnley | 35 |
| Brentford | 29 | Watford | 23 |
| Norwich City | 22 | Norwich City | 22 |

(Figure 5.4)

### 6.3.1. Top 6

First, it is worth noting that my model's prediction was very accurate in predicting the top 3 teams, as Manchester City, Liverpool, and Chelsea all finished in the top 3 in both my prediction and the actual standings. The entirety of the top 6 was also included however the order was a off with Arsenal, Manchester united and Spurs finishing 4$^{th}$,5$^{th}$ and 6$^{th}$ respectively in my model but 5$^{th}$,6$^{th}$ and 4$^{th}$ respectively in the actual table.

When predicting the top 6 the model was far more accurate than other portions of the league, this could be because the top 6 teams in the Premier League tend to have a strong track record of consistently performing well over the years. They have the financial resources to attract the best players and have top-tier managers who can develop successful team strategies. For example, teams such as Manchester City and Liverpool from 2017/18 onwards (from where I collected my data onwards) have been the most dominant teams in the league. During this period, both teams have consistently finished in the top two places in the league and have won the title between them. Manchester City won the league in 2017/18, 2018/19, 2019/20, and 2020/21, while Liverpool won the league in 2019/20. The dominance of these two teams has had a significant impact on the accuracy of my model when predicting the top teams in the league. By including data from 2017/18

onwards, my model has been able to take into account the sustained success of Manchester City and Liverpool and predict that they would finish in the top two places once again in the 2021/22 season.

This dominance may also explain the overestimate of the total points, my model predicted that Manchester City would achieve a total of 104 points, but they managed 93 points, indicating an 11-point deviation from the predicted value. Similarly, Liverpool was anticipated to finish with 96 points, yet they finished with 92 points, leading to a discrepancy of 4 points.

### 6.3.2. Mid-Table

Predicting the mid-table teams in the English Premier League becomes increasingly harder because there is less of a gap in quality between these teams. Unlike the top teams, which have more financial resources to sign top players and have more consistent performances, the mid-table teams tend to have more varied performances from season to season. Additionally, the difference in points between mid-table teams can be very small, making it harder to predict which teams will finish in the top half of the table. Furthermore, mid-table teams can have unexpected performances, such as a strong finish to the season or a sudden loss of form, making it even harder to predict their final position.

For example, West Ham United had a standout season, finishing in seventh place with 56 points. They were one of the surprise teams of the season, with an impressive attacking line led by Jesse Lingard, who joined on loan from Manchester United in January. Whereas my projected with collected and trained data from the past predicted them to battle for relegation merely escaping with 1 point of bottom 3 with 31 points. My model has not been adapted enough to understand transfers such as Lingard to west ham that ultimately had a great change to their season.

Therefore, predicting mid-table teams in the English Premier League requires more sophisticated analysis and models that can consider multiple variables, including team form, player performance, injuries, and other factors.

### 6.3.3. Relegation

Predicting relegation teams is challenging due to the nature of the relegation battle. At the start of the season, there are usually several teams that could potentially be relegated, and predicting which of these teams will ultimately end up in the relegation zone can be difficult. Additionally, many of the teams that are fighting against relegation are often closely matched in terms of talent, making it challenging to predict which team will ultimately come out on top. This explains why despite Watford being relegated that season from data collected my model had them safe at 38 points 4 spots off relegation.

The prediction was increasingly even more difficult with teams such as Brentford that Watford swapped places on the tables. The algorithm predicted to get relegated with 29 points. The reason why it was considerably harder to predict teams like Brentford using your machine learning algorithm could be due to various factors. One possible explanation is that Brentford is a newly

promoted team to the Premier League, and there is not much data available about their performance in the top-flight league. This lack of historical data makes it more challenging to train your algorithm accurately and predict their performance in the current season. Furthermore, Brentford's playing style and strategies may differ significantly from the teams they faced in the lower leagues, making it harder to predict their performance. Moreover, the Premier League is a highly competitive league, and the difference in quality between teams is relatively narrow. Small variations in player form, injuries, or tactical approaches can have a considerable impact on a team's performance, making it harder to predict their outcome accurately.

In contrast, predicting teams like Norwich may be easier because they have already played in the Premier League in recent years, giving us access to their historical data, player performances, and other relevant factors. Additionally, some teams tend to maintain their playing style and tactics, which can be easier to predict. Teams such as Norwich, that have recently been promoted from the Championship, have a higher probability of being relegated due to the level of competition in the Premier League. These teams may have had success in the lower division but may struggle to compete against the more established teams in the Premier League. Additionally, they may have a smaller budget and fewer resources, which can further limit their ability to compete at the highest level. On this prediction was 100% accurate the points of 22 and position of $20^{th}$ being the exact same as the actual table.

# 7. Discussion

For this chapter, I will provide a comprehensive discussion of my project, encompassing a range of critical elements such as personal development, success, improvements, and limitations. Firstly, I will reflect on the impact of the project on my personal growth and the new skills and knowledge that I have acquired through this experience. I will then analyse the overall success of the project, including its effectiveness in achieving the desired outcomes and its impact on the broader context in which it was deployed. Furthermore, I will explore potential avenues for improvement, highlighting areas where the project could be optimized for greater efficacy. Finally, I will critically assess the limitations of the project, considering the factors that may have impeded its performance and any potential opportunities for further development. Through this comprehensive discussion, I aim to provide a nuanced and insightful evaluation of the project, drawing on both personal experience and objective analysis to offer a multifaceted perspective.

## 7.1. Personal Development

The project required skills in planning, research analysis, learning new technologies and developing and explaining complex software. The work was devised with what started at a novice level of the python language but then grew to become far more comfortable and confident. This can also be applied to machine learning and data scraping. The ongoing module of data mining and machine learning contributed greatly to my newfound skill and my improvement was recognisable throughout the project. Huge amounts of time investment were also required to design and create my model and analysis as well as the research done alongside.

However, my lack of experience with said skills did cause me to inevitably miss certain deadlines and self-allocated time frames (Gantt chart). Improvements can also be made with increased confidence and skill for my model and my results can be much more promising. Despite these slight self-inflicted errors, the success of my project was unhindered.

## 7.2. Project success

In summary, I deem the project as a triumph. I have carried out a comprehensive exploration into the predictability of the English Premier League, utilizing a sophisticated machine learning algorithm, coupled with data sourced from multiple seasons. Additionally, I have presented a detailed examination of the outcomes, delineating both the robust points and inadequacies of my model, thus enhancing the overall transparency and validity of my findings.

Upon completing this project, I can confidently say that all of the objectives outlined in chapter 1.3 have been successfully achieved. Firstly, the CSV file utilized contained data from a minimum of four seasons, providing a robust and diverse data set for analysis. This is crucial for the development of an accurate machine learning model, as a larger data set allows for patterns to emerge and increases the likelihood of accurate predictions. Multiple predictors were implemented, which not only increased the accuracy of the algorithm but also allowed for a more nuanced analysis of the factors influencing team performance. The program was designed to predict not only win, but also draw or

loss outcomes. This is particularly important in a league like the Premier League, where results can often be unpredictable. Incorporating recent form and previous season data in the analysis enabled more precise predictions. This was particularly useful in predicting the positions of the top teams, as their dominance in recent seasons made their positions relatively predictable. Conducting a comprehensive simulation that accounted for all possible matchups in the league was essential in generating a complete and accurate Premier League table. This allowed for a comprehensive analysis of each team's predicted performance throughout the season, providing valuable insights into the relative strengths and weaknesses of each team.

In short, I believe that my project was successful in achieving its objectives, providing a comprehensive and accurate prediction of the Premier League season.

## 7.3. Project Limitations and improvements

There are many limitations and areas where I can improve my project for example:

Data Quality: The accuracy and reliability of predictions depend heavily on the quality of the data used. In this case, while I used data from multiple seasons, it's important to ensure that the data is up-to-date, complete, and accurate. Additionally, incorporating more data sources, such as player and team statistics, could provide a more comprehensive view of team performance.

Model Complexity: While my model performed well in predicting the top six teams, it struggled with mid-table and relegation predictions. Improving the complexity of the model could help capture more nuances in team performance, such as the impact of individual players, tactical changes, and team morale.

Randomness of Sports: Sports are inherently unpredictable, and unexpected outcomes can occur due to various factors such as injuries, referee decisions, and weather conditions. It's important to acknowledge and account for this randomness in any sports prediction model.

Comparative Analysis: In evaluating the success of my model, it's important to compare its performance to other existing prediction models, such as bookmakers' odds or other statistical models. This can help provide a benchmark for the accuracy of my model and identify areas for improvement.

Overall, my project is a good start in exploring the predictability of the English Premier League. By addressing the limitations and incorporating feedback, I can improve the accuracy and applicability of your model in the future.

# 8. Conclusion

Is the premier league predictable, my investigation into the predictability of the English Premier League has revealed a nuanced and complex picture. While the dominance of certain top teams, such as Manchester City and Liverpool, makes predicting their positions relatively accurate, mid-table teams present a much greater challenge. Factors such as injuries, transfers, and changes in coaching staff can all have significant impacts on a team's performance, and predicting these outcomes is a formidable task.

Furthermore, my machine learning model using data from the 2017/18 season onwards provided an accurate prediction of the top six teams but struggled with mid-table and relegation predictions. This highlights the importance of using a range of methods and data sources to achieve the most accurate results.

In comparison to the predictions of a major bookmaker like Betfred, my model provided similar results in terms of the top six teams but showed notable differences in the mid-table and relegation predictions. This underscores the inherent unpredictability of the Premier League, and the need for careful analysis and consideration of multiple factors when making predictions.

In conclusion, while the Premier League may not be entirely predictable, a combination of thorough analysis, advanced machine learning techniques, and a nuanced understanding of the complexities of football can provide valuable insights into the outcomes of this highly competitive and exciting league.

# References


1) [1] Towards Data Science. (2021). What is the best programming language for machine learning? Retrieved from https://towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7
2) [2] Towards Data Science. (2019). O Jogo bonito: Predicting the Premier League with a random model. Retrieved from https://towardsdatascience.com/o-jogo-bonito-predicting-the-premier-league-with-a-random-model-1b02fa3a7e5a
3) [3] Arman E. (2022). Predicting the English Premier League with Machine Learning. Retrieved from https://armantee.github.io/predicting/
4) [4] Zhen X., Xiaowei W., Xingkai Z., & Xueying H. (2021). A Match Result Prediction System for English Premier League Using Machine Learning Techniques. IEEE Access, 9, 64232-64240. doi: 10.1109/ACCESS.2021.3076931
5) [5] Dataquest. (n.d.). Predicting EPL Football Match Winners Using Machine Learning. Retrieved from https://app.dataquest.io/m/99992/portfolio-project%3A-predicting-epl-football-match-winners-using-machine-learning/1/project-overview
6) [6] FBref. (n.d.). Premier League Stats. Retrieved from https://fbref.com/en/comps/9/Premier-League-Stats
7) [7] Keith Galli. (2020). Machine Learning Tutorial Python - 6: Training and Testing Data. Retrieved from https://www.youtube.com/watch?v=vmEHCJofslg&ab_channel=KeithGalli
8) [8] FootyStats. (n.d.). Premier League Home/Away League Table. Retrieved from https://footystats.org/england/premier-league/home-away-league-table
9) [9] Analytics Vidhya. (2015). Scikit-Learn: Your Go-To Machine Learning Library. Retrieved from https://www.analyticsvidhya.com/blog/2015/01/scikit-learn-python-machine-learning-tool/
10) [10] Exploring Our Fluid Earth. (n.d.). Practices of Science: Precision vs Accuracy. Retrieved from https://manoa.hawaii.edu/exploringourfluidearth/physical/world-ocean/map-distortion/practices-science-precision-vs-accuracy#:~:text=Accuracy%20refers%20to%20how%20close,item%20are%20to%20each%20other.
11) [11] How They Play. (n.d.). 10 Reasons Why All The World Loves The EPL. Retrieved from https://howtheyplay.com/team-sports/10-Reasons-Why-All-The-World-Loves-The-EPL
12) [12] Oxera. (2016). Odds on: What was the probability of Leicester City's 2016 success? Retrieved from https://www.oxera.com/insights/agenda/articles/odds-on-what-was-the-probability-of-leicester-citys-2016-success/
13) [13] Scikit-learn. (n.d.). GridSearchCV. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
14) [14] Towards Data Science. (2021). Understanding Random Forest. Retrieved from https://towardsdatascience.com/understanding-random-forest-58381e0602d2
15) [15] Lundberg, S. & Lee, S. (2017). Gradient Boosting Machine Learning: Gradient Boosted Tree Models Can Be More Accurate Than Neural Networks and More https://www.researchgate.net/figure/Gradient-boosted-tree-models-can-be-more-accurate-than-neural-networks-and-more_fig8_338662630