**Course: IT4142E - INTRODUCTION TO DATA SCIENCE**

**Advisor: Assoc. Prof.Than Quang Khoat**

**Authors: Group 17**

**Nguyen Tong Minh - 20204885**
**Nguyen Cong Dat - 20200137**
**Ly Nhat Nam - 20204886**
**Hoang Long Vu - 20204897**

# DATA SCIENCE REPORT

## Starting Lineups & Match Results in English Premier League

SOICT
*Enlightening Your Digital Future*

**Academic semester: 2022.1**


# Project Report

# Staring Lineups and Match Results in English Premier League

**Course: IT4142E - Introduction to Data Science**

**Authors: Group 17**

**Nguyen Tong Minh - 20204885**
**Nguyen Cong Đat - 20200137**
**Ly Nhat Nam - 20204886**
**Hoang Long Vu - 20204897**

**Advisor: Assoc. Prof. Than Quang Khoat**

**Academic semester: 2022.1**

---

**Abstract**

One of the most popular sports in the world is soccer. Many people may try to use some pertinent information to forecast the outcome of a match, whether they are soccer fans, players of online soccer games, or even the professional coach of a soccer club. The data from the actual game, such as the total number of shots, yellow or red cards, fouls committed, etc., by the home and away sides, is often the foundation for these types of prediction models. However, using information from players who started for their teams during the English Premier League's first three seasons from 2019 to 2022, this study tried to forecast the outcome of soccer matches (win, draw, or loss). It covered their overall skills including attack, defence, teamplay, discipline and goalkeeping. We select some of valuable features by using unsupervised Machine Learning techniques, namely Principal Component Analysis, K-Means and supervised techniques, namely Support Vector Machine, ensemble methods such as Random Forest, Ada Boosting, Gradient Boosting. As a result of the research, it was determined that PCA and LDA preserve the most information and keep the relationship between instance. Random forest training on original data was the best classifier in this project. In addition to making predictions about the outcomes, this field offer coaches, supporters, and participants in online soccer games advice on what traits and player positions are most crucial to the outcome, which will effect how they choose the starting lineup.

We would like to express our deep gratitude to Prof. Than Quang Khoat, who gives us this golden opportunity to work on this wonderful project. Without his tremendous support and instruction, as well as his generosity, we cannot complete this project on time. We would like to extend our sincere thanks to Mr. Pham Quang Hieu, the teaching assistant of the class, for your enthusiasm with the course, your invaluable tutorial time has resulted in incalculable experiences.

# Contents

# 1 Introduction

## 1.1 Background

The English Premier League, commonly referred to as the EPL, is the top tier of professional football in England and one of the most popular and widely followed football leagues in the world. It was founded in 1992 and consists of 20 clubs playing 38 matches each, for a total of 380 matches in a season. The league operates on a promotion and relegation system, with the bottom three clubs being relegated to the second tier of English football and being replaced by the top three clubs from that division. Over the years, the Premier League has become known for its high levels of competitiveness, world-class players, and intense rivalries between clubs. The league attracts a huge following both domestically and internationally, and its matches are broadcast to millions of viewers in over 200 countries.

Therefore, predicting the outcome of matches in the Premier League is a popular topic among football fans and analysts. The analysis of match result prediction is a systematic approach to understanding the factors that influence the outcome of a football match and using that information to make more accurate predictions. It is used by fans, experts, and bookmakers to make predictions and place bets. It can also be used by teams and managers to identify areas for improvement and make more informed decisions about tactics and player selection. There are many different methods and approaches used to make predictions, ranging from simple intuition and gut feeling to complex models, which include statistical models, machine learning algorithms, and expert opinion, and the results can provide valuable insights into the performance of teams and players.And in this report, we show our analysis about match result prediction based on two main key: starting line-ups and player statistics using machine learning.

## 1.2 Problem Description

Predicting the outcome of a football match based on lineup and player information is a common approach used by experts and analysts. The idea is to use data and information about the players and their positions on the field to make predictions about the performance of the teams and how they may match up against each other. This can be particularly useful when there are key players missing or when the lineup of a team has changed, as it can give an indication of how these changes may impact the team's performance.

The data for this the research was collected from the 3 seasons 2021/2022 season, 2019/2020 season, 2020/2021 season of the English Premier League, involving a total of 1140 games (380 matches each season). Information regarding home and away teams' names, starting lineup lists and final results were acquired from `https://www.premierleague.com/results`. The personal data of the players was obtained from `https://fantasy.premierleague.com/`. We first propose some EDA for the players' positions on the field to understand how different positions and player combinations may influence the outcome of a match as well as the individual player data, such as their goals, assists, and shots on goal which provide the information that can be used to get a better understanding of a player's ability and how they may impact the match.

The study aims to determine the best classifier for constructing a training model by comparing the prediction accuracy of Random Forest, Support Vector Machine, AdaBoost, Gradient Boost, etc . Once the most accurate classifier is selected, it will be used to identify the most relevant features through feature selection. Finally, the model will assess and analyze the prediction results and highlight the significance of different player features in various positions. As a result, the ultimate model will offer recommendations and insights for online players, fans, and professional coaches. Overall, predicting the outcome of a football match based on lineup and player information is an important aspect of match analysis. By considering the different factors and data that can impact a team's performance, analysts can make more informed predictions about the outcome of a match and provide valuable insights for fans and bookmakers. The result is learned by 80% all total matches and used to evaluate the 20 % left.

However, despite the data we have owned, there are challenges hindering our objective such as incompletes in dimensions of data quality (e.g.,

data internal and external inconsistency, lack of player information for a match,...), nonsensical integration of matches and player statistics (i.e. players are put randomly in places of lineup representation) regardless of the football tactics. Furthermore, the problem itself is not a comfortable dinner as a match since unpredictability is the beauty of any sport. Nevertheless, we are obviously able to forecast its results in specific perspectives and circumstances.

### 1.3 Outline

We organize the report as follow:

**Chapter 2** Describe the way of collecting data from the source and how to clean them.

**Chapter 3** Summary how row data have been integrate for further analysis.

**Chapter 4** Demonstrate exploratory data analysis for getting the property and valuable insights helping build more effective model.

**Chapter 5** Outline the methodology used for each experiment in the project, providing a step-by-step explanation of the modeling, parameter-fixing, and prediction processes for each model used. The text also examines and evaluates the results and conclusions drawn from this series of experiments.

## 2 Data Collection & Cleaning

The desired information is collected from official archive of English Premier League (`https://www.premierleague.com/`), under the *JSON* format for flexibility (e.g., nested fields,...). We then convert the obtained data to a tabular form and perform cleaning, making it consistent across multiple sources and usable for further stages. Besides, cleaning is responsible for checking the missing records and if any, we re-collect them.

### 2.1 Data Collection

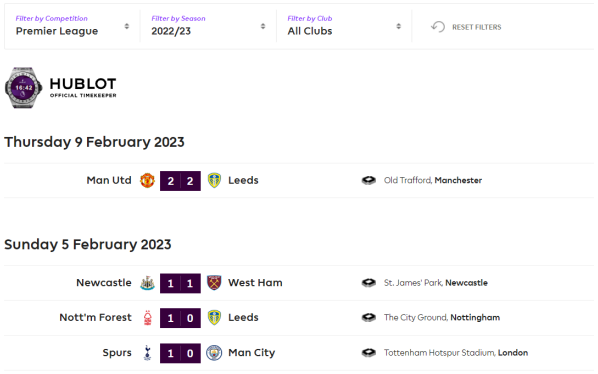


Figure 1: Match URLs from `www.premierleague.com/results`.

The dataset required for our objective consists of match information about starting lineups with the result, and player statistics by season. The two are later integrated to obtain desired data, which will be used to learn the relationship between lineups and match results.


Figure 2: Lineup information in a match web page from `www.premierleague.com/results`.

***Match information*** The former source, match data, is initially crawled from `www.premierleague.com/results` (figure 1), which is the official website of English

Premier League, providing the latest, specific and full statistics for each match per week. By one season, there are 20 teams battling it out for honour of being crowned champions. Each team has two matches (i.e. one on the home stadium, one on the away) with one another team in the rest 19 teams. In total, a team has to play 38 matches and a season 380 matches. Due to stability, we aim to assess the matches of the seasons that are complete, which has no ongoing weekly update on both matches and player statistics. Therefore, only URLs of the matches in the most recent seasons (approximately 1300 matches), from 2019/20 to the current, are crawled. We then scrape the content (i.e., lineup information, match result) from the web page of each URL (figure 2). One to note is that the lineup information consists of one player identifier for each player included and they can be used to fill the player performance to the slot in the match data.

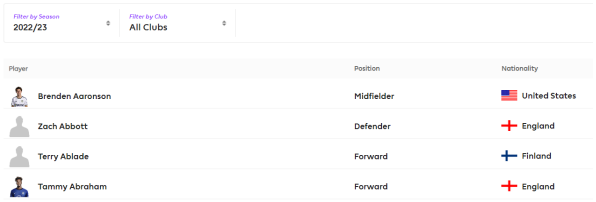


Figure 3: Player URLs from `www.premierleague.com/players`.

***Player statistics*** The latter source, player data, is crawled from `www.premierleague.com/players` (figure 3), which is also the official database of English Premier League, instead of sub-links of players from match web page of lineups. The reason is the high dynamic player market of the league across months of a year, in which one player is possibly in a different team after the transfer period of summer and winter.

Later, we scrape each player's statistics as the content of the corresponding web page from an URL. The data consists of the non-nested data fields, which can be transformed directly into tabular format, involving player's name, identifier, season and performance indices (e.g., goal kicks, blocked shots,...) (figure 4). Cautiously, we investigate the features recorded of players in different positions (i.e., forward, defender, midfielder, goalkeeper) and find out the fact that a position does not include all features of an another one. For example, forwards such as Rashford do not have a record of accurate long balls in team play like defenders. In this case, we fill such non-record features with 0.0, which is intuitive as a forward, from the example above, either rarely does long balls or do with distinct purposes.

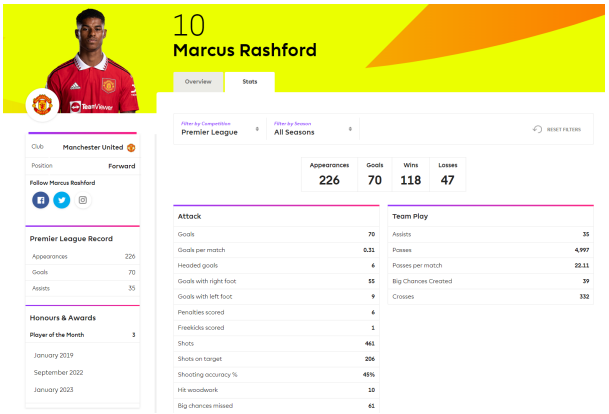


Figure 4: An example of statistics of a player in English Premier League.

There are 51 features to evaluate a player in games (e.g., recoveries, own goals, shots on target,...), which are classified into 05 different skill groups including *defence*, *team play*, *discipline*, *attack* and *goalkeeping*. The player data on the website of the league is recorded per season and only the current season receives weekly updates on the performance.

One issue during data collection is that the structure of modern websites like Premier League is dynamic, which means in order to get whole content of the web pages, clients must interact with their interface. Therefore, *Scrapy* itself cannot handle the Premier League website but we has to combine it with *Selenium* which provides interactive mechanisms for scrapy spiders.

## 2.2 Data Cleaning

The match and player data are loaded from *JSON* files, then converted into tabular form. Since the fields of match data are nested, as

shown in the figure 5, we parse it by flattening all the fields with players' name being excluded. The *id* is kept on the purpose of mapping to the player data.

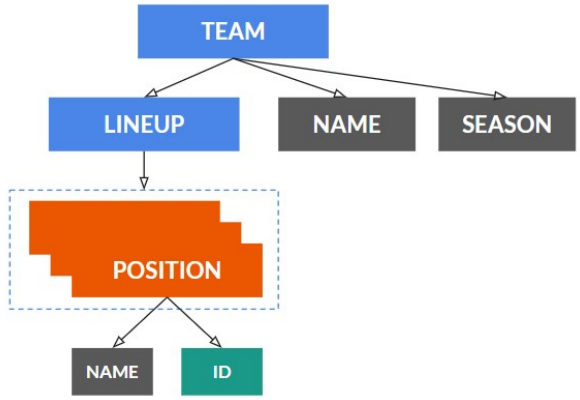


Figure 5: *JSON* structure of the match data.

Obviously, a match has 02 teams and a team has 11 players in the starting lineup, resulting in 22 players in total per match. However, each of them has different arrangements and strategies, which raises the concern about the importance in position order. The following seven possible line-ups are found during the matches across 03 Premier League seasons from 2018/19:

- 1-3-4-3 represents 1 Goalkeeper, 3 Defenders, 4 Midfielders and 3 Forwards
- 1-3-5-2 represents 1 Goalkeeper, 3 Defenders, 5 Midfielders and 2 Forwards
- 1-4-3-3 represents 1 Goalkeeper, 4 Defenders, 3 Midfielders and 3 Forwards
- 1-4-4-2 represents 1 Goalkeeper, 4 Defenders, 4 Midfielders and 2 Forwards
- 1-4-5-1 represents 1 Goalkeeper, 4 Defenders, 4 Midfielders and 2 Forwards
- 1-5-3-2 represents 1 Goalkeeper, 5 Defenders, 3 Midfielders and 2 Forwards
- 1-5-4-1 represents 1 Goalkeeper, 5 Defenders, 4 Midfielders and 1 Forwards

In order to satisfy all of these possible lineups, we provide 28 places for players, instead of 22, that assigns maximum places for each position in a team, which means 01 place for goalkeeper, 05 places for defenders, 05 places for midfielders and 03 places for forwards. If the starting lineup does not have a full number of players in one position, the empty place will be substituted by using $NaN$. The figure 6 describes the basic features and general structure of the tabular match data.

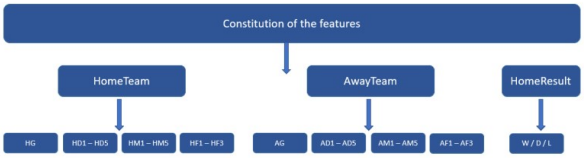


Figure 6: Constitution of the features of match data.

For player data, since its structure is as simple as described in section 2.1, we then just have to perform an usual conventions.

### 2.2.1 Data Consistency

Data consistency is one of ten dimensions of data quality, which is, in our scope, the measurement of compliance with required formats and values, representing the same former and latter of a record that is stored internally or distributed across storage. As a result of this statement, we assess the two datasets in order to correct structural errors and provide well-built interface so that we can learn the information more conveniently and exactly.

***Match information*** Cells in the match data, except for the primary key (i.e., season and teams' name), all contain either players' ID or empty, that they are of the same data field in other words. Unfortunately, the types of columns are not identical (i.e., either *int*, *float* or *object*) Usually, data science frameworks (e.g., *Pandas*) handle such issues implicitly provided that the value representations are no distinct (e.g., 1884, 1884.0 and "1884" are recognized identically despite type difference). However, since we are not sure that they would have identical interface, all columns that express positions have their data

type transformed into $float$. The work is executed successfully, claiming that there is no error in value representation with the formats being synchronized as well. Later, we check for the number of teams involved per season, that of seasons and players per team in a match so as to detect any logic anomalies (e.g., more than 20 teams per season). As a result, we find that some matches scraped have a team with just 10 players in the starting lineup. The external references claim that the error may come from the scraping systems but not itself, that we hence decide to drop those irregular matches due to the small size of false data (approximately 70 matches). Last but not least, the uniqueness (by using primary key) is checked to ensure that there is no duplicated records.

***Player statistics*** Similarly, the player data poses the issue of inconsistent data types. Unfortunately, some object-typed columns (e.g., $teamplay_passes$) have irregular value representations such as "1,448" which cannot be recognized as numbers automatically by the software. Therefore, we transform such cases following common annotations (i.e. "1448"), then convert them into $float$. Also, the uniqueness is examined by using $id$ and $season$ as the primary key, that passes.

On the other hand, players' ID in the player data is of object-typed while that in the match data of float-typed. Converting that of the former is thus performed to synchronize the formats and values.

### 2.2.2 Missing Data

As shown in figure 7, the extreme positions such as $home/fw_2$, $dhome/df_4$ are empty in most matches. It could be an insight that coaches did not prefer using tactical lineups that are extreme to a role. For handling such "natural" missing slots, we opt to leave them with $NaN$ and later fill it with a certain value that maps to a representation of empty slots in the player data.

For the player data, there is no missing value. However, it does not contains all the players that appear in match data. For example, player with $id$ 441192 has his data not collected at first. Therefore, we manually retrieve corresponding URLs for the missing cases since the number of records is count on fingers. Those URLs are then packed and fed into the spider for re-scraping. Re-checking later shows no missing record.

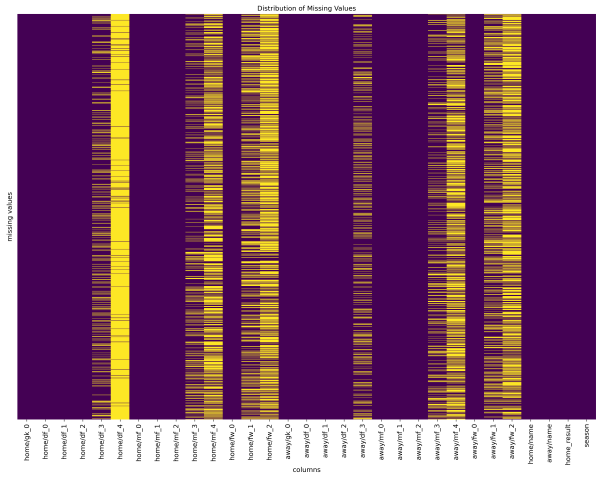


Figure 7: Missing distribution of the match data.

### 2.2.3 Outlier Handling

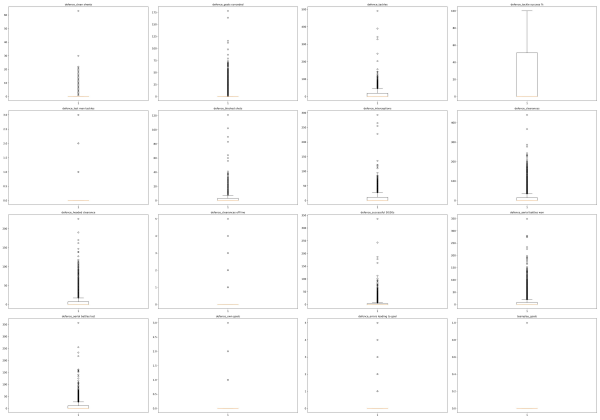

Figure 8: Some box plots for each performance measurements of the players.

As the match data contains solely players' id, we concentrate assessing this aspect of the player data. The box plots of the figure 8 on performance measurements partly describe the critical spreading out of data (e.g., $shooting\_accuracy\_\%$ with 24% records being out of the quantile range).

This scattering may be a sign of that the data collected is fault due to either systematic errors (e.g., wrong scraping algorithms,...) or external sources. Fortunately, the manual verification on a number of extreme cases raises our belief in that there is no such errors; in other words, the data is telling the truth. For example, Tony Springett in the season 2021/22 achieved 100% shooting accuracy but he just played less games (03 appearances throughout the whole season) and, as a midfielder, he made a small number of shooting. Consequently, the outliers are left since they are the truth of English Premier League, which we want our modelling to deal with.

#### 2.2.4 Redundant Data

Likewise, we pay attention to player statistics. One expected issue is the sparsity due to the high-dimensional data and the number of zero values in each feature which are filled for positions that do not have its record. Also, some numeric features are discrete and have a small range (e.g., $red_cards$ is in range from 0 to 1 and just a few players get red cards). All of them are shown in the figure 9. This means either dimensionality reduction or feature selection can improve the quality of data, summarizing and extracting the truly impactful data fields as well as alleviating the natural noise detected.

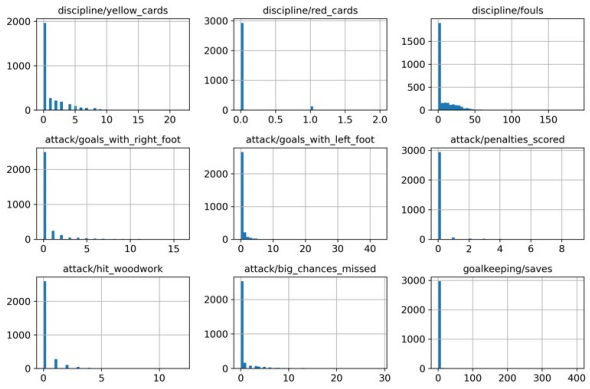


Figure 9: Histogram of some player features.

Besides, we check for the seasons appearing on both data. The player data has the season 2018/19 while the match data has the season 2022/23 but not vice versa. Thus, we delete the records of these both seasons on the two tables since they are not integratable.

## 3 Data Integration

Data integration is the process of combining data from different sources into a single, unified view. We found that data integration in our situation plays an important role. In order to make accurate predictions, various types of data must be gathered and integrated from various sources such as player performance statistics and team tactics. We have to process and analyze data in order to identify patterns and trends that can provide valuable insights into the outcome of a match. After integraion, this data will be use for explanatory analysis and then be used to make predictions based on the most valuable information available. By utilizing data integration, the scores such as accuracy and ROC AUC can be significantly improved and hence, we can make more informed decisions about betting and other related activities.

### 3.1 Player data

Original player dataset has 54 features, arrange into 5 primary skills, which are attack, defence, goalkeeping, teamplay, discipline. We scale the range of value for a single skill, as its interval and value fluctuate between each.


Figure 10: Skill of player.

The case is complicate with original data, as many attributes have sparse values, which is common in the player data, because a player often play a single role, hence the score of him in the skill of that role might be high but it will be very low in other skills. Therefore, dimension reduction is very much needed. It is ideal to have only one feature represent the whole property of a skill.

***Dimension reduction for features compression*** Principal component analysis (PCA) is a flexible and popular method for linear data, as we can choose target dimension number that we want, in this case is 5. We also experiment with Linear discriminant analysis in the case data is non-linear. After reducing dimensions number using PCA and LDA which will be refered in 4, we have final dataset which each player represented by 5 and 3 attributes, respectively. By using dimension reduction algorithm, it becomes easier to visualize the data and understand the relationships between the features. As working with a large number of features can slow down the training process of machine learning algorithms, but by reducing the number of features, the algorithms can be trained faster, which can be seen in our problem when there are 27 positions, with 5 attributes for each player, we already have total 135 features in final training data, whereas some of value can be very sparse or high-correlation, furthermore, it is hard for us to choose which features should be kept in the features selection process.

## 3.2 Match data

Our dataset contains 27 columns for possible player positions, specifically 1/1 home/away goalkeeper, 5/4 home/away defender, 5/5 home/away midfielder, 3/3 home/away forward. Each element in a record is the "id" of that player in the "players" dataset. This dataset will be the main dataset to integrate for the training phase.

***Features order*** Common tabular dataset has attributes determined since the beginning and we collect data base on solely those attributes, hence each feature distribution will be different from the others, while we also want all of the value in an attribute has some meanings together. Unlike any others tabular data, football match attributes are slightly different, since we only have the positions as well as the name of player who play in that line-up. So that, the main problem is, which player should we put in one feature? As we have 27 positions for players, there are 27! ways to order them, each way result in different outcome!

***Propose solution*** To overcome this problem, we have some observations into the data. A football team is divided into five roles and the rival team should not have any ties to the home team, thereby allow us to merely rearrange players of that role in either home team or away team, yielding a smaller number of arrangements need to take into account. After doing this, we arrange order of player in each position index. To do this, we hypothesize that the player with highest overall performance should stand together. Then, the up-rising issue is that, how can we find the overall performance of player, because the performance of each player base on several skills such as attack or defence. Therefore, we consider using the most significant skill which describe type of position of the player. For example, as can be seen from 11, attack is the main role of a forward player, defence correspond to a defender, team play is for the midfielder.

| | attack | defence | teamplay | discipline | goalkeeping |
|---|---|---|---|---|---|
| 18 | 73.982708 | 15.335281 | 7.175765 | 3.066969 | 0.439277 |
| 8 | 1.767565 | 87.948065 | 6.350099 | 3.579535 | 0.354736 |
| 6 | 7.260360 | 13.107760 | 74.386371 | 2.272490 | 2.973019 |
| 7 | 6.231148 | 17.950586 | 1.845017 | 73.800671 | 0.172578 |
| 11 | 1.457117 | 10.625710 | 1.015231 | 0.350746 | 86.551196 |

**NOTE:** 5 PCs could be a representative for each role (e.g., $22^{th}$ PC represents the attack ability of a player).

Figure 11: Variance of features in PCA player's data.

## 3.3 Player and match integration

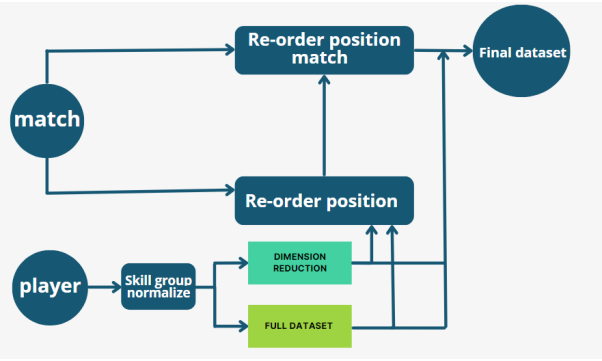


Figure 12: Pipeline of the integration process.

Here is the overall pipeline for our integrate process. We will describe clearly each step below.

1. match, player: Our datasets after cleaned.

2. Skill group normalize: We use standard normalizer for attributes in a specific skill, scale it to normal distribution.

3. After normalize, we divide it into full data (data preserving 54 attributes) and reduction data (data using dimension reduction method like PCA or LDA).

4. Re-order position: We replace player id in each match's record with the significant score of player in player data that we propose above. For each role of a single record, we re-order the score from highest to lowest, clearly the role that don't have enough player, any missing player element will be leftover and be assumed to be the worst player.

5. Re-order position match: We use the index of re-order score to re-organize the player id in match data.

6. Finally, as we have organized data, we will merge both dataset, matching id and season of each player in each record of match data with player's corresponding features in either full or reduction player data.

To summary, with our integration process, we have 3 final dataset, which is LDA data, PCA data and full data. All of them contain purely numerical value and are ready to go to the EDA phase and modelling phase.

# 4 Exploratory Data Analysis

Exploratory data analysis (EDA) aims to analyze datasets we have, summarizing their main characteristics by using statistical graphics, data visualization and hypothesis testing if any. These steps possibly grant insights into data storytelling beyond the formal modelling, which may enhance our model fitting and data experiments afterwards.

## 4.1 Player Data

### 4.1.1 Univariate analysis

The player statistics are dispersed and spread out over large ranges, according to the section 2.2.4, but the statistical description (figure 13) brings distinct perspectives for such an irregular characteristic. In particular, quantile ranges of many performance measurements (e.g., *clean_sheets*, *goal_conceded*,...) contains solely zeros, which are definitely the consequence of non-record features in each position mentioned in section 2.1. As a result, the outliers of a feature are actually not "anomalous"; in other words, most of them are of the players whose position have departments track that feature. The processing of statistical outliers are then left on the integrated data, handled by using advanced algorithms (e.g., isolation tree,..) assess data points on multi-dimensional spaces.

Out[6]:

| | defence/clean_sheets | defence/goals_conceded | defence/tackles | defence/tackle_success_% |
|---|---|---|---|---|
| count | 3058.000000 | 3058.000000 | 3058.000000 | 3058.000000 |
| mean | 1.011445 | 5.318509 | 12.274689 | 19.432309 |
| std | 2.857000 | 13.408723 | 21.738711 | 28.817012 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 50% | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 75% | 0.000000 | 0.000000 | 18.000000 | 50.000000 |
| max | 30.000000 | 178.000000 | 390.000000 | 100.000000 |

8 rows × 51 columns

Figure 13: Table of statistical description for player features.

### 4.1.2 Multivariate analysis

As shown in the figure 14, nearly half area of the matrix is either light out or vice versa, which describe the strong pairwise monotonic correlation across features of the players. In fact, there are approximately 36% having more than 0.4 of correlation coefficient, which means more than $\frac{1}{3}$ feature pairs point out a moderate relationship. To explain appropriately about this phenomenon, we look at some top strongly correlated pairs such as *saves* - *goal_kicks* (0.9919) and *saves* - *throw_outs* (0.9918), which intuitively hints the backstory under the iceberg. Explicitly, when a goalkeeper catch a ball in his hands, he has to initiate the ball back to a place on the

grass so as to continue the match, by basically two ways with one being a ball kick and the another a throw. That is why such pairs can get a uncommonly strong correlation, which consequently claims the redundancy in the statistics for players. Therefore, it is a great practice for us to apply dimensionality reduction directly on the player data before integration, comprehending all important information of a player, instead of from multiple players of a lineup that is difficult to understand and possibly mixes irrelevant features from different players together.

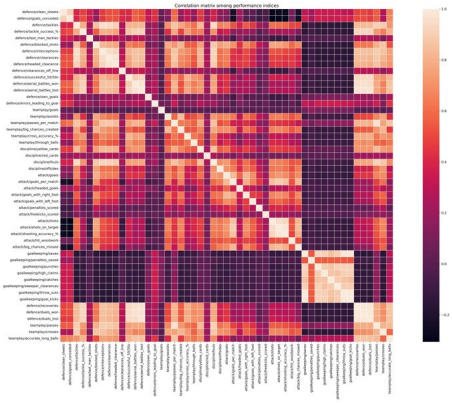

Figure 14: Correlation matrix between performance indices.

#### 4.1.3 Dimensionality reduction

Most points in a high-dimensional hypercube are very close to the border. Points in the high-dimensional hypercube are far apart even if they lie within the same unit hypercube. This deduces a fact about high-dimensional datasets: They are at risk of being very sparse. In other words, new instances fed into models are usually far away from the training samples, making the models struggle to recognize the identical characteristics from the learned reference. Thus, the overall modelling comes to a failure, being said overfitting. Since our resource and data size are limited, that we are not capable of collecting more data for the sake of the quatity scaling exponentially with the dimensions, we opt to break the so-called curse of dimensionality [2] by preparing other versions of the player data, whose dimensions are reduced, along with the original data so that model fitting could try both of them.

As observed, our player instances are not spread out uniformly across all dimensions but many features are almost constant, while the others are highly correlated. Then, those instances potentially lie within (or close to) a much lower dimensional subspace of the high-dimensional space. Besides, many player statistics are linearly correlated, implying that the data representation appears not to twist and turn but plain. Therefore, principal component analysis (PCA) with its linear projection is experimented initially. To choose the ideal number of principal axes, we use the elbow method (figure 15) and find out 24 dimensions are the best choice. Nonetheless, if the data is reduced to 24 dimensions, the variance ratio explained is just 69%, which drops too much information. The another option, which is officially used, is to choose the number of dimensions that preserve 95% of information and it is 24. Lastly, the relationship between original fields and principal axes are evaluated. Conclusively, five axes (18, 8, 6, 7, 11) purely contain the most information from skill groups - attack, defence, team play, discipline and goalkeeping, respectively and all the axes have the positive mapping coefficients, except for defence vs. the $8^{th}$ axis.

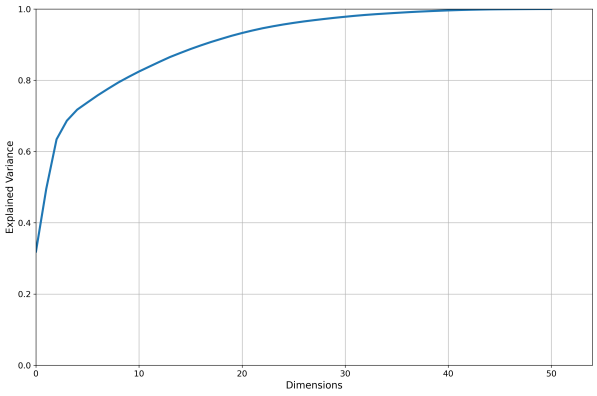


Figure 15: Explained variance ratio of PCA on dimensions.

We also try some other advanced dimensionality reduction techniques, out of linear projection, including manifold learning (i.e. multidimensional scaling (MDS), t-distributed stochas-

tic neighbor embedding (t-SNE) and isomap) and discriminant analysis (i.e., linear discriminant analysis (LDA)). One common characteristic of those manifold learning algorithms are their dependence on similarity preservation; while MDS and isomap maintain the distances between the instances, t-SNE keeps similar instances close and dissimilar instances apart. On the other hand, LDA is actually an classification algorithm learning the most discriminative axes between the classes, which defines a hyperplane of the projection; it is a great technique to reduce dimensionality before running another classification algorithm.

Through experiments, LDA, trained to discriminate positions, remains the only method that is capable of visualizing acceptably separating regions for each position and playing style in a three-dimensional feature space (figure 16).

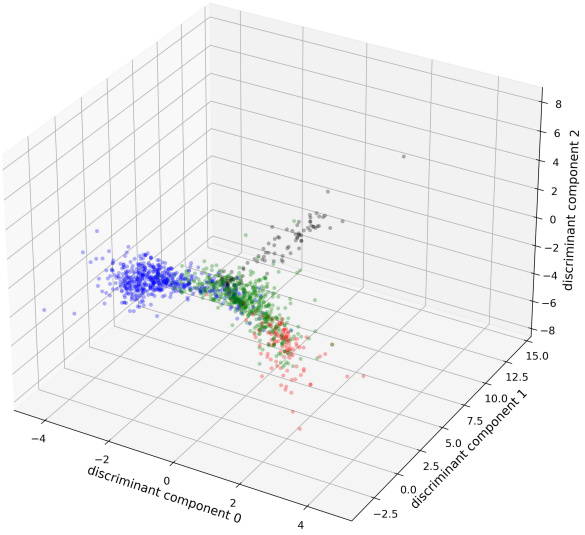


Figure 16: The feature space of LDA. Blue points are defender, red ones are forwards, black goalkeepers and green midfielders.

Unfortunately, the relationships between discriminant components and original axes are not intuitive and we have to estimate it using clustering and correlation. For the first component (C0), we fit a simple algorithm, K-Means, resulting in two clusters as shown in the figure 17. Interestingly, only defenders are spread across two clusters while the other positions stay mostly (over 95% each) in the first cluster, which may mean that the component could point out differences among defenders. Therefore, the values of C0 could be assessed for arranging defenders in lineup during integration. Likewise, the second component (C1) is able to differentiate characteristics of goalkeepers but not of other positions. The last one, however, just explains a small amount of variance, which is then not utilized to arrange any position.

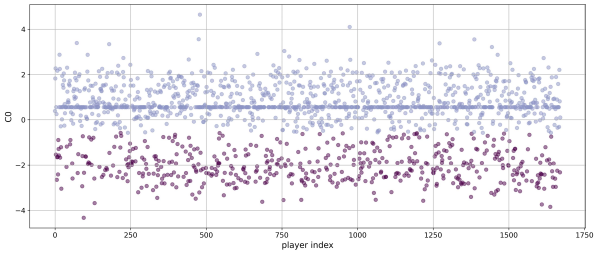


Figure 17: The two clusters of K-Means based on the first component.

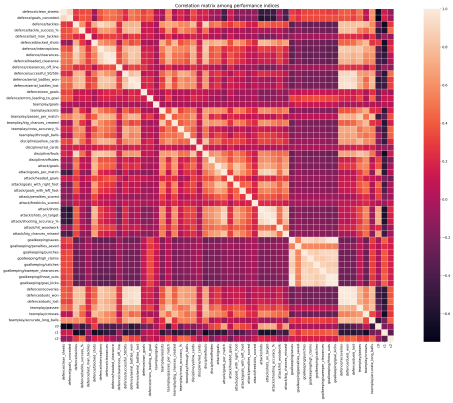

Figure 18: Correlation matrix between original features and discriminant components of LDA.

For the arrangement of forwards and midfielders in lineup, we assess the correlation matrix of original features and the discriminant components (figure 18). As expected, C0 highly correlates with defence indices, and interestingly with attack ones positively (e.g., *shots_on_targets*, *goals*, *shots*,...) as well. Thus, C0 can be used to estimate the performance of forwards; higher the value of C0, better performance the

forward has. Besides, C1 is also approximately linearly dependent on goalkeeping features and moreover negatively correlates with the statistics of goal opportunities (e.g., *tackles*, *crosses*, *big_chances_created*,...); hence this component could also assess the arrangement of midfielders.

## 4.2 Match Data

The match data singly contains solely the player id and match results across the three successive seasons, from 2019/20. Here, each season does not includes all 380 matches (figure 19) because of the cleaning performed in the previous section (section 2.2) that adheres to the data quality. Fortunately, this loss does not affect critically to the diversity of the data across across seasons since it includes just few matches.

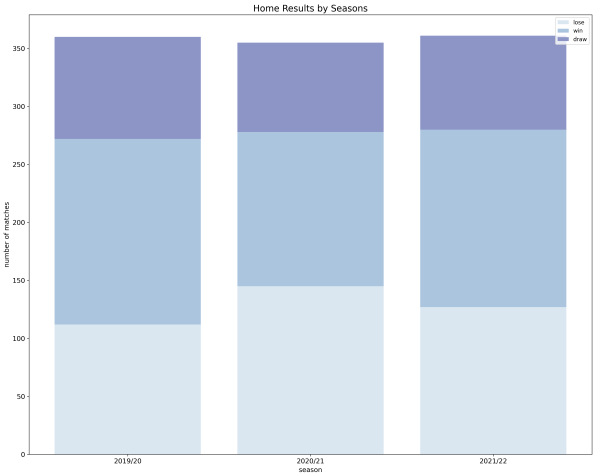


Figure 19: Number of matches by season.

For the home results, the percentage of win matches accounts for nearly 40% of the entire dataset, as shown in the figure 20. The number of lose and draw ones estimate 36% and 23%, respectively. In spite of the fact that there is a slight imbalance across the types of match result, the difference in their numbers of instances are just below 150 matches (15% of the dataset), which is acceptable.

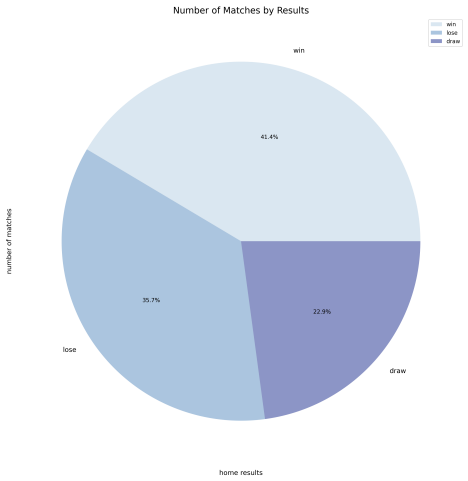


Figure 20: Percentage of matches by result.

## 4.3 Matches with Player Data

Firstly, we will discuss some observations from the data statistics, and its relation to real life cases in football.

Statistically, there are 470 players participated in EPL 2019/2020. The total number of players observes an upward trend in the latter seasons, reaching 472 players and 484 players in the next two seasons respectively. The dataset only works with players that appear at least 1 time in the season therefore the minimum and maximum number of matches that any player can attend in a season is 38.

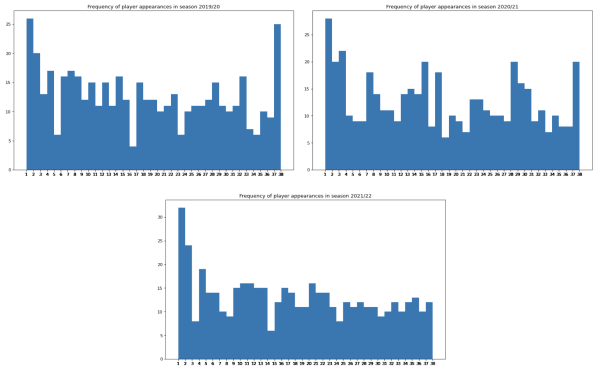


Figure 21: Distribution of number of matches played.

The distribution of the number of occurence of

players through the three seasons is illustrated in the Figure 21.

**Players with highest win/lose/ratio over the seasons**

Now we want to which player has the highest home win/lose/draw ratio. However, because the distribution of of the number of occurence is not uniform, we need statistical method to determine the minimum number of appearance for a player to have statistically significant comparison.

To determine the minimum number of appearances required for a statistically significant comparison, we can use the formula for minimum sample size:

$$n = \frac{Z^2 \times p \times (1-p)}{e^2} \quad (1)$$

where: $Z$ is the Z-score. We want a confidence level of 80% in this comparison, therefore the z-value confidence level is 1.645, $p$ is the estimated proportion of appearances, and $e$ is the desired margin of error (0.05).

For the season 2019/20, the average number of appearances is 17 matches thus the equation 1 yields the minimum sample size of 18, so players need to play at least 18 matches to be taken into significant comparison. Following the similar calculations, comparisons of players in the season 2020/21 and 2021/22 only consider those who played at least 19 matches.

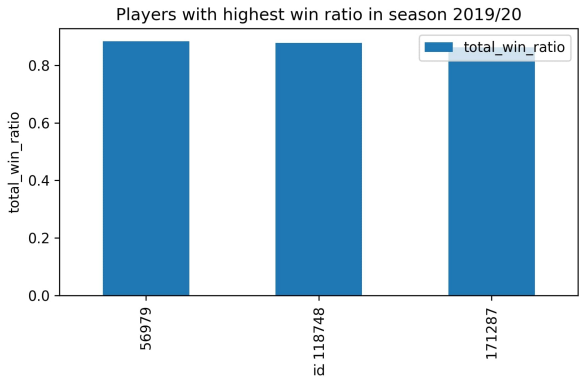


Figure 22: Players with highest win ratio in 2019/20 season.

***Season 2019/20*** With the minimum number of matches played acquired, we can now conduct further analysis on the players with highest win, lose, or draw ratio statistically significant. For the 2019/20 season, all three players with highest win ratio (Figure 22 all belong to one club: Liverpool. The history can clearly explains this observation, as Liverpool achieve the victory league in the EPL 2019/20 season [1]. The three players corresponding are: Jordan Henderson (26 matches played, 23 wins, 88.4%), Mohamed Salah (29 wins out of 33 matches played, 87.8%), and Joe Gomez (19 victories out of 22 played matches, 86.3%). Following up the analysis above, we can also have an insight on the players with highest win ratio when playing as home or away team, shown in Figure 23.

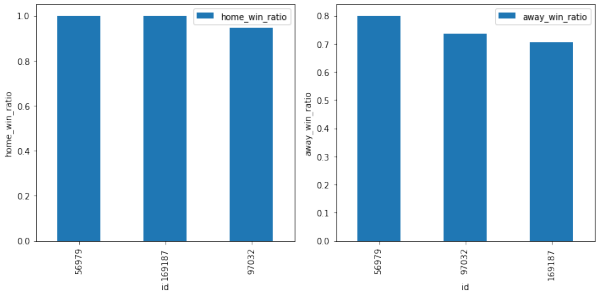


Figure 23: Players with highest home and away win ratio in 2019/20 season.

The three players with highest win ratio when playing as home team in the 2019/20 season is Jordan Henderson, followed by Trent Alexander-Arnold and Virgil van Dijk - who are also Liverpool players.

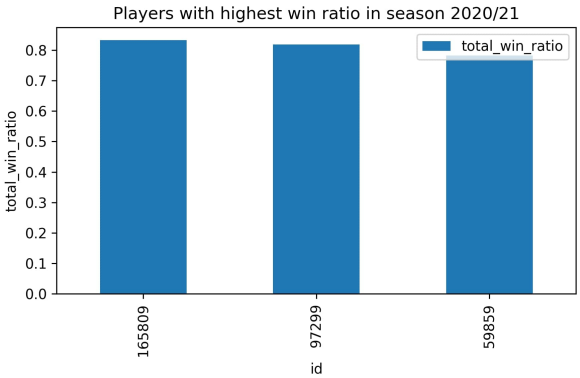


Figure 24: Players with highest win ratio in 2020/21 season.

***Season 2020/21*** Moving on to the analysis on the next season, 2020/21, Manchester City won the league so it is understandable that their players achieved the highest win ratio, either on home or away (Figure 24). The three players are Bernardo Silva, John Stones and Ilkay Gundogan, ranked in decreasing order. The three players also achieved highest home and away win ratio after playing 24 matches throughout the season (Figure 25).

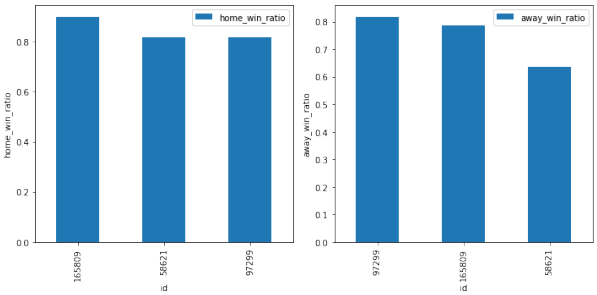


Figure 25: Players with highest home and away win ratio in 2020/21 season.

***Season 2021/21*** The season 2021/22 observed the same trend. As shown in the Figure 26, Ilkay Gundogan, Aymeric Laporte and Rodri (all Manchester City players) have highest win ratio in total of over 80% after winning the trophy in the 2021/22 season.

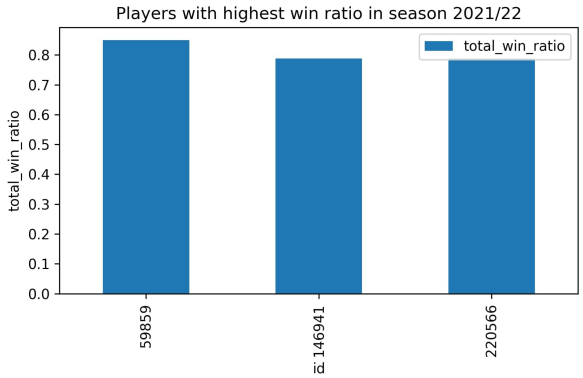


Figure 26: Players with highest win ratio in 2021/22 season.

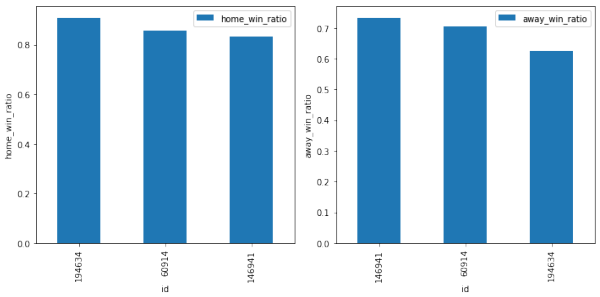


Figure 27: Players with highest home and away win ratio in 2021/22 season.

In brief, players that belong to trophy-winning teams tend to have the best win ratio, even playing as home or away team.

**Most common lineups**

Team managers tend to use more players in the midfield position to gain better possession in the match. This is shown in the Figure 28.

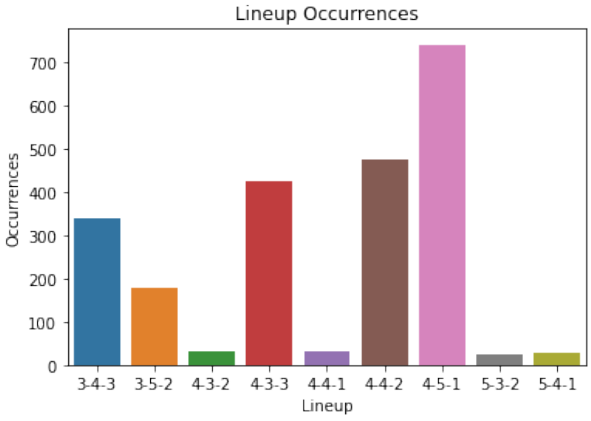


Figure 28: Lineup used by teams across all three seasons.

The 4-5-1 lineup is used in over 700 matches, much more than the second most common lineup (4-4-2, which is used around 500 times). Moreover, the Figure 33 also shows high correlation between the features of midfield players with the possibility of winning the match. The 4-5-1 lineup also accounts for the highest portion of victories, with over 32% in 700 matches. The pie charts in Figure 29 demonstrates the efficiency of different lineups.

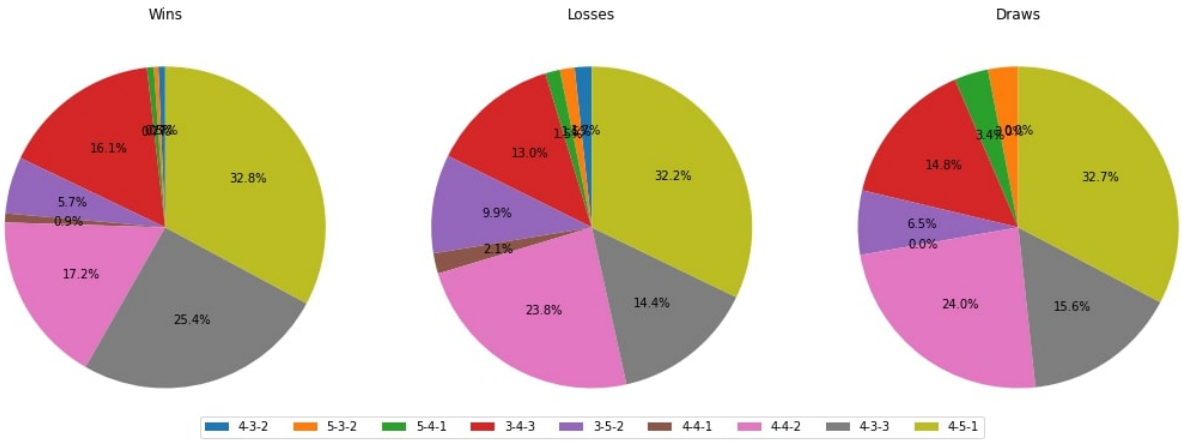


Figure 29: How much each lineup account for the total number of win/lose/draw matches in the seasons.

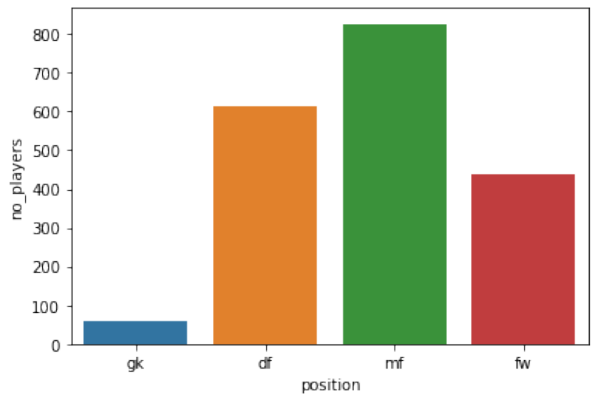


Figure 30: Number of players by each position.

In real life, teams invest in having more midfielders and attackers (Figure 30), managers also tend to rotate many choices for the midfield and attacking positions, as these positions require intense activities. Therefore, players in midfield and attacking positions tend to have more rest time in less important matches. From the Figure 31, it is easily seen that goalkeepers tend to play most of the matches of the season, which means they have less rest time compared to other attacking positions.

**Position and player features importance**

Next, we will analyse the importance of players features and yield the most decisive attributes to the victory of a team after conducting experimental modelling in the Section 5. Upon implementing the Random Forest model, we can extract the features importance and yield the top 10 most decisive attributes as shown in Figure 32.

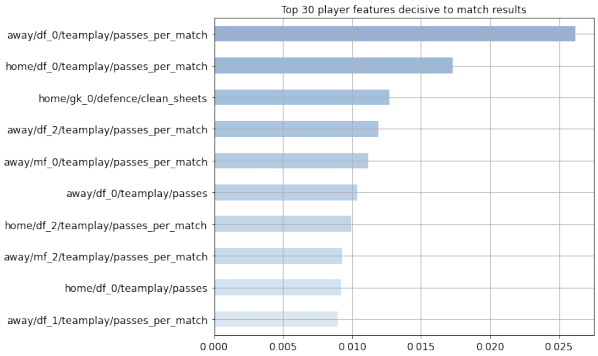


Figure 32: Top 10 most important features.

We can observe the playstyle of EPL teams here: building from the bottom (from the defending line and midfield line), focus on passes and possession to control the match. This also shows that forwarders are not decisive in matches results as in the past anymore.

To affirm the outlook above, we can further analyse how different positions contribute to the overall victory of the team. Figure 33 shows that the defenders are among the most important players on the field, followed by the players in the midfield lines. Goalkeepers and forwarders' work do not seem to directly affect the win opportunity of the team.

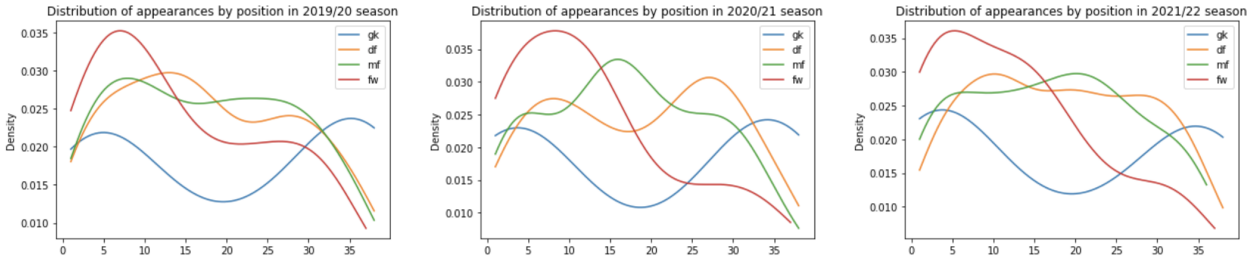


Figure 31: Number of appearances by player position.

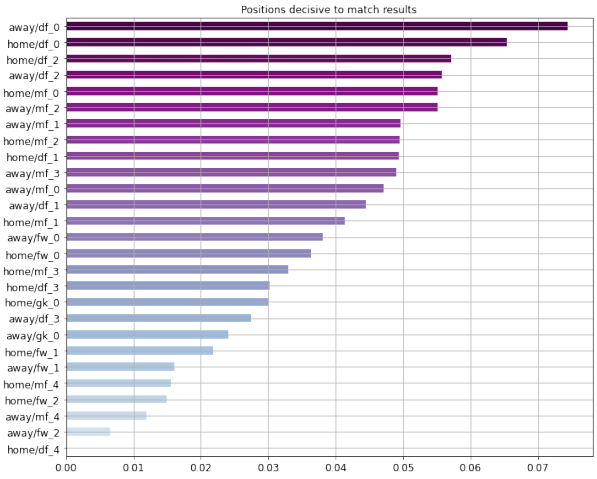


Figure 33: Top most important positions.

Nevertheless, we cannot deny that forwarders are still the main scorer of the teams. The evidence is that performance indices like: shooting accuracy, the number of shots on target, etc. are still among the most important indices to decide the match. Defenders - whose primary goal is to stop opposition attacks during the match and prevent opposing players from scoring a goal - in EPL teams still possess the most important indices towards the vicotry of the team, as shown in Figure 34.

To conclude, successful EPL teams tend to play with more midfielders to control the match as much as they can, and backed up mainly by excellent defenders to build up the possession from the bottom. Forwarders should excel in the shooting accuracy to decide the result of the match. Players that played most during the season are also mainly defenders and midfielders, specially goalkeepers tend to attend most of the matches of the teams.

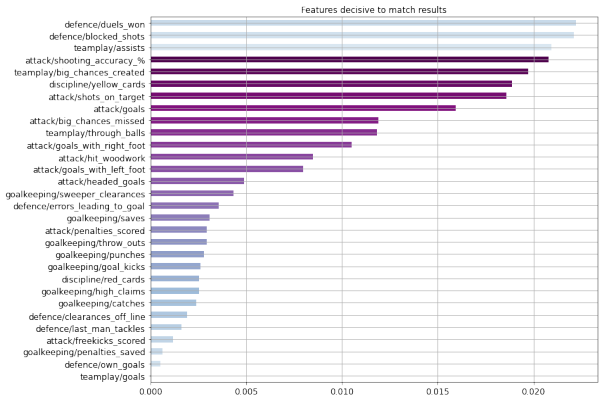


Figure 34: Top most important performance indices.

# 5 Modelling

The modelling aims to design and seek for the models and settings of machine learning that are capable of learning the general pattern of match results based on starting lineups. As decided during the exploratory data analysis, we will experiment on both two integrated datasets deduced by using LDA and PCA, and the match data with full player statistics. The settings that hold the insights about the former data are then utilized to re-explore the relationship between matches and players as well as optimizing the modelling. To start up, the models and common measures to search the best setting used during experiment-

ing on both types of dataset are introduced.

**Algorithm Proposals**
We start experimenting from simple machine learning models (e.g., logistic regression) to more complex algorithms (e.g., ensemble learning,...), which grants a wide perspective on the model fitting, helping us select the well-fitted models which remains as less complex as possible.

***Logistic Regression*** Logistic regression is a statistical method used to model the relationship between a binary outcome variable and one or more predictor variables. It is a type of generalized linear model that uses a logistic function to model the probability of the outcome variable being in one of two possible categories.

The logistic function, also known as the sigmoid function, takes the form of a curve that ranges from 0 to 1, and it is used to transform the linear predictor values into probabilities. The logistic regression model estimates the parameters of the logistic function to maximize the likelihood of the observed data, which allows for the prediction of the probability of the binary outcome variable based on the predictor variables. The probability of the binary outcome variable is modeled using the logistic function, which is defined as follows:

$$P(Y = 1|X) = \frac{e^{\beta_0+\beta_1 X_1+\beta_2 X_2+...+\beta_p X_p}}{(1 + e^{\beta_0+\beta_1 X_1+\beta_2 X_2+...+\beta_p X_p})}$$

where:

- $P(Y = 1|X)$: the probability of the binary outcome variable being equal to 1 given the values of the predictor variables X.
- $\beta_0$ : the intercept term.
- $\beta_1, \beta_2, ..., \beta_p$ : the coefficients of the variates.
- $X_1, X_2, ..., X_p$ : the values of the variates.

The logistic regression model estimates the values of the coefficients $\beta_0, \beta_1, \beta_2, ..., \beta_p$ using maximum likelihood estimation. The goal is to find the values of the coefficients that maximize the likelihood of the observed data. The likelihood function is defined as the product of the probabilities of the observed data, given the values of the predictor variables and the model parameters.

***Support Vector Machine*** Support Vector Machines (SVMs) are based on the mathematical theory of optimization and the concept of separating hyperplanes in high-dimensional spaces. The algorithm was first introduced by Vapnik and colleagues in the 1990s and has since become one of the most popular and widely used machine learning techniques.

SVMs are based on the concept of structural risk minimization, which involves finding a model that has the lowest expected risk. The expected risk is the sum of the training error and a penalty for model complexity. The SVM algorithm involves finding a hyperplane in a high-dimensional space that maximizes the margin between the two classes of data. The margin is defined as the distance between the hyperplane and the closest data points of each class. The optimal hyperplane is chosen in such a way that it maximizes the margin while still correctly classifying the training data.

To handle non-linearly separable data, SVMs use the kernel trick, which involves mapping the input data into a higher-dimensional space, where it is more likely to be linearly separable. The choice of kernel function has a significant impact on the performance of the SVM algorithm, and different kernels can be selected based on the nature of the data and the problem at hand.

***Decision Tree*** A decision tree is a popular tool in data mining and machine learning that is used to model and visualize decision-making processes. It is a tree-like graph that uses a set of decision rules and their corresponding outcomes to help individuals or organizations make informed decisions.

The decision tree is constructed by breaking down a complex decision-making process into a series of simpler decisions or questions. Each decision node in the tree represents a question or decision that must be made, and each branch represents a possible outcome of that decision. The leaves of the tree represent the final decisions or outcomes.

The construction of a decision tree involves two

main steps: tree induction and tree pruning. Tree induction involves creating the initial tree structure by selecting the best feature to split the data at each node. This process is typically guided by a heuristic such as the gini index or information gain. Once the tree is constructed, tree pruning is used to remove unnecessary branches and nodes, which helps to prevent overfitting.

***Random Forest*** Random Forest is a supervised machine learning algorithm that uses an ensemble of decision trees to enhance prediction accuracy and stability. It creates multiple decision trees on random subsets of input data and combines their predictions to produce a final result. Each tree is trained independently using a subset of features, and the final output is determined by majority vote. Random Forest can handle high-dimensional data and noisy environments and can estimate feature importance, which helps in understanding data relationships. It is widely used in classification and regression tasks.

Random forest also uses the concept of bootstrap aggregating (bagging), which is a statistical technique that involves creating multiple samples of a data set by randomly selecting data points with replacement. The multiple samples are then used to train multiple decision trees, each with a different set of training data, and their outputs are combined to produce the final prediction.

Additionally, random forest uses a technique called feature bagging or random subspace method, which involves randomly selecting a subset of features at each node of a decision tree. This helps to reduce the correlation between trees and prevent overfitting.

***Adaptive Boosting*** AdaBoost, also known as Adaptive Boosting, is a popular ensemble learning method that combines multiple weak classifiers to create a strong classifier. The basic idea behind AdaBoost is to iteratively train a series of weak classifiers on different subsets of the training data and combine them in a way that results in a more accurate and robust final classifier.

AdaBoost is an ensemble learning method that trains weak classifiers in multiple iterations. In each iteration, a weak classifier is trained on a subset of the training data with adjusted sample weights. Misclassified samples are given higher weights, and correctly classified samples are given lower weights. During final classification, each weak classifier contributes its prediction, which is weighted by its accuracy. The final prediction is made by combining the weighted predictions of all the weak classifiers.

One of the key strengths of AdaBoost is its ability to handle high-dimensional data and avoid overfitting. This is achieved by selecting only the most informative features during the training process, which reduces the dimensionality of the problem and improves generalization performance. Another strength of AdaBoost is its ability to handle class imbalance, where one class has significantly more samples than the other. By assigning higher weights to the misclassified samples, AdaBoost can effectively balance the training process and improve the classification accuracy for the minority class.

The theoretical foundation of AdaBoost lies in the concept of exponential loss minimization. This involves finding a set of weights for the weak classifiers that minimizes the exponential loss function. The exponential loss function is defined as the negative logarithm of the probability of the correct classification, and its minimization ensures that the final classifier is highly accurate.

***Gradient Boosting*** Gradient boosting, proposed by Friedman in 1999, is a machine learning technique used for both regression and classification tasks. It is a type of ensemble method that combines multiple weak predictors to form a strong predictor. The idea behind gradient boosting is to iteratively add new models to the ensemble that correct the errors made by the previous models.

The core concept of gradient boosting is to minimize a loss function by adding new models to the ensemble that correct the errors of the previous models. At each iteration, the model fits a new weak learner to the negative gradient of the loss function with respect to the previous model's predictions. This new learner is then added to the ensemble, and the process is repeated until the error on the training set is minimized or a stopping criterion is met.

The gradient boosting algorithm is based on a functional gradient descent approach, where the goal is to find the function that minimizes the loss function. In this approach, the gradient of the loss function is computed with respect to the current function, and a new function is fitted to the negative gradient. The new function is added to the current function to produce a new function, which is then used in the next iteration.

**Methodology**

To train the models and compare them in their good sets of hyperparameters found, we design a default scheme with Area Under the Receiver Operating Characteristic Curve (ROC AUC) for evaluation and a searching algorithm, Bayesian optimization, for model tuning. Finally, each of the best models is evaluated on the test set to observe the final result, comparing them with each other and examining the fitting quality (e.g., overfitting,...). The best of best will be benchmarked with more intuitive classification metrics (e.g., accuracy, precision,...) and through error analysis.

***Bayesian optimization*** Grid search is a basic hyperparameter tuning method, set up with a grid of hyperparameter values. This tuning algorithm exhaustively searches this space in a sequential manner and trains a model for every possible combination of hyperparameter values, which is not efficient in both computing power and time. Random search differs from grid search in that we no longer provide an explicit set of possible values for each hyperparameter; rather, users provide a statistical distribution for each hyperparameter from which values are sampled. Nevertheless, both of these techniques have their steps being independent of each other, causing blind defining of search spaces. In Bayesian approaches [3], in contrast to random or grid search, keep track of past evaluation results which they use to form a probabilistic model mapping hyperparameters to a probability of a score on the objective function: $P(score|hyperparameter)$. This optimizatio methods are efficient because they select hyperparameters in an informed manner. By prioritizing hyperparameters that appear more promising from past results, Bayesian methods can find the best hyperparameters in lesser time (in fewer iterations) than both grid search and random search. Therefore, it will be introduced for hyperparameter tuning.

***Area under the Receiver Operating Characteristic Curve (ROC AUC)*** The area under a receiver operating characteristic (ROC) curve, abbreviated as AUC, is a single scalar value that measures the overall performance of a binary classifier [4]. The AUC value is within the range [0.5–1.0], where the minimum value represents the performance of a random classifier and the maximum value would correspond to a perfect classifier (e.g., with a classification error rate equivalent to zero). The AUC is a robust overall measure to evaluate the performance of score classifiers because its calculation relies on the complete ROC curve and thus involves all possible classification thresholds, which are not the case for other intuitive metrics such as accuracy.

Our dataset poses a mere imbalance in match results, as concerned in the section 4.2. Therefore, we calculate AUC of all possible pairwise combinations of classes and find their unweighted mean, which is insensitive to the class imbalance. In other words, we use macro one-vs-one ROC AUC to evaluate the generalized performance of the models through cross-validation.

***Create a test set*** We split the dataset, including 80% for training and 20% for final testing, in a random manner since the data is just insignificantly imbalance. The training set is used to train and search for the best model of each kind using cross-validation and bayes optimization. The test set is left out till the last minute when we recall it to benchmark those best models with never-before-seen instances in order to attain best of the best which is then evaluated using intuitive metrics (e.g., accuracy) and error analysis for obvious interpretation of model performance.

## 5.1 Matches with Full Player Data

### 5.1.1 Preprocessing

**Manage empty positions** Empty position is the position on the field that belongs to no one based on the line-up and indicated in raw data as None value - missing data. The integrated data left the default value for statistic data of empty position was -100. However, this number was too large compared to the existing statistic of on-field players that make the distribution of attribute become skewed and create many outlier which led to incorrect analysis and future prediction of model. Therefore, we decide to set empty position values as zeros which are natural a "player" which does actually not play in match.

**Normalization** After dealing with the empty positions, we perform standard normalization for the integrated data to scale them to the form of normal distribution (or close), making it easier for machine learning algorithms (e.g., SVM) to learn the data thoroughly.

### 5.1.2 Experiment

We intend to pick around four best models for the final testing. The pure fitting, with 3-fold cross-validation for a quick test, observes the issue of overfitting for all algorithms listed. Therefore, in creating search spaces, we focus on regularized terms (e.g., $C$ of SVM, $n\_estimators$ of random forest,...) and search for the regularization setup that could generalize acceptably.

The validated result of the best random forest is highest (0.652 AUC), following by SVC (0.644), gradient boosting (0.643), ada boost (0.62) and decision tree (0.60), respectively. The logistic regression is left far away under as it, on the contrary, seems underfitting. Since our current best model is a random forest, which can provide the insight into the features' importance to right decisions of the model, we perform feature selection and exclude all non-contributory statistics which accounts for over $\frac{1}{3}$ number of features. The results obtained do not change much but all the models slightly increase their AUC with random forest staying the best (0.659).

```
Loading model forest_bayes_search
              precision    recall  f1-score   support

        draw       0.50      0.04      0.07        56
        lose       0.51      0.70      0.59        71
         win       0.58      0.73      0.64        89

    accuracy                           0.54       216
   macro avg       0.53      0.49      0.43       216
weighted avg       0.53      0.54      0.48       216
```

Figure 35: The classification report for the best model.

The final testing is then carried out for three algorithms including gradient boosting, random forest and SVC, comparing their best models trained on reduced and full data. The crown then belongs to the random forest trained on the former (.651 vs. .659 AUC for test and train). The other models follow closely but overfit stronger (e.g., the random forest learning the reduced data achieve 0.65 AUC on train set but just 0.63 on test set). All the testing results are summarized in the table 1.

| Dataset | **RF** | GB | SVC |
|---|---|---|---|
| full train | **0.659** | 0.657 | 0.647 |
| reduced train | **0.651** | 0.6516 | 0.658 |
| full test | **0.651** | 0.645 | 0.608 |
| reduced test | **0.628** | 0.6414 | 0.609 |

Table 1: The results of testing. RF stands for random forest, GB gradient boosting and SVC support vector classifier.

Last but not least, we bring the intuitive report for our best model, shown in the figure 35. One can observe that the model can recognize well the lose and win matches but not draw ones. The model's prediction can be 54% true, which is better than a baseline model that only predict a match in which the home team wins (approximately 41%). Also, since predicting a match result are itself not easy as pie, this performance is acceptable.

The figure 36 shows the errors that model usually commits. We can see that the model is confused between draw and win the most. We also can see that the model misclassifies matches as win frequently. Therefore, when the model says

a match is draw, or even lose, the observes should stay careful.

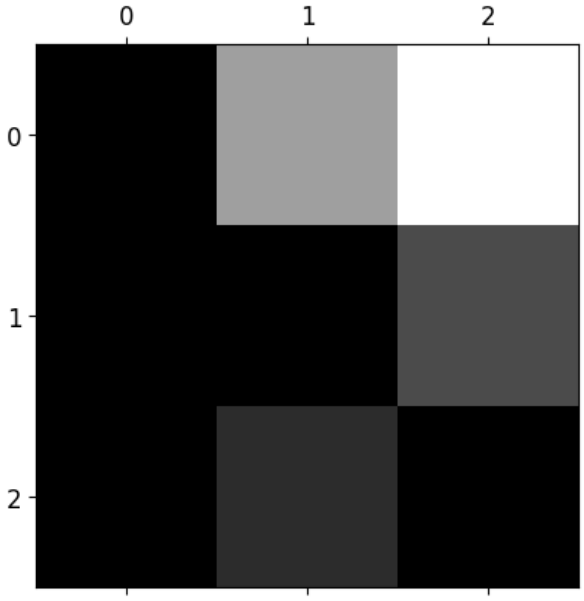


Figure 36: Error matrix of the best model. Brighter the slot, more errors; the row stands for actual classes and vice versa (0 - draw, 1 - lose, 2 - win).

## 5.2 Matches with Reduced Player Data

### 5.2.1 Preprocessing

As above, we also manage empty positions and normalize the data as the very first steps. One to note that instead of filling zeros, we add the minimum value to the empty's statistics since the fields of dimensionality reduction is not just naturally positive as the original.

**Outlier removal** After dimensionality reduction, the information was summarized in fewer components. Although the outlier can tell the truth of the real data, they still can skew the results of an analysis or model. By removing outliers, we can obtain a more representative sample of the data and reduce the impact of noise to better insights and more accurate predictions. We choose 2 method to do outlier removal: Isolation Forest and DBSCAN clustering.(All the parameter was manual tuning)

- *Isolation Forest*

  + The isolation forest algorithm uses a tree-based method to identify anomalies in data by randomly partitioning it. The algorithm randomly selects a feature and value to create a split on the data, and repeats this process recursively to create a new branch in the tree until the data is completely isolated or a predefined depth is reached. The isolation score of a data point is calculated as the average path length from the root node to the terminal node in the tree, with a higher isolation score indicating a higher likelihood of being an anomaly.

  + The isolation forest algorithm has multiple benefits for anomaly detection, including its computational efficiency for large datasets with high-dimensional features, its insensitivity to the data distribution and its ability to handle both linear and non-linear relationships between the features, and its flexibility and robustness due to not requiring assumptions about the data distribution. The algorithm only needs to create a few trees to identify anomalies, and its random selection of features and values reduces computational complexity. This makes it suitable for a wide range of applications with complex data patterns and structures.

  + In this paper, we use function IsolationForest of sklearn library with parameter( n_estimators=50, max_samples='auto', contamination= 0.1, max_features=1.0)

- *Density-Based Spatial Clustering of Applications with Noise*

  + The DBSCAN algorithm is a density-based method for clustering data points that identifies clusters based on their density. It defines two parameters, epsilon and minimum points, to construct a cluster. The algorithm starts with a random data point and checks if its $\epsilon$-neighborhood contains at least MinPts data points. If this condition is met, a new cluster is formed and the algorithm continues to add neighboring data

points until no more points can be added. If a data point does not belong to any cluster and has fewer than MinPts neighbors, it is considered an outlier.

+ The DBSCAN algorithm has multiple benefits for anomaly detection, including its ability to handle arbitrary cluster shapes and its robustness to noise and outliers, its computational efficiency for large datasets, and its scalability and parallelizability, making it useful for real-time applications and big data analysis. The algorithm does not require predefined assumptions about the data distribution or cluster shapes and only needs to examine the $\epsilon$-neighborhood of each data point, without requiring a distance matrix or pairwise comparisons.

+ In this paper, we use function DBSCAN of sklearn library with parameter ( min_samples=2, eps = 8.5 with LDA and 30 with PCA)

#### 5.2.2 Experiment

Due to the decision of reducing the data's dimension with 2 approach: LDA and PCA. The experiment was conducted with both kind of data to evaluate exactly the effectioness of the model. All model was testing with Baysian optimization to tune the importance hyperparameter related to regularization, and evaluation with 3-fold cross validation for the most corrected genalization. Overall, AUC score reflexed the underfitting issues of all models that best result was 0.65 in training and 0.63 on testing( Random Forest), which might be an inevitable consequence of predicting an hard-predictable problem like match result as well as line-up and player information do not represent and reflex enough the result of a match.

**LDA** Table 2 show the ordered result of all model with LDA dimension reduction data. The result show that the best model is Random Forest with 0.61 AUC score of training set and 0.55 AUC score of testing set and Gradient Boosting with 0.60 AUC score of training set and 0.53 AUC score of testing set. The other model result, however, have closed result with the best, which respectively are support vector machine (0.59 and 0.51 on train and test set),decision tree(0.57 and 0.51), adaptive boost(0.54 and 0.51) and logistic regression( 0.58 and 0.51).

| Model | train | test |
|---|---|---|
| RF | 0.594 | 0.549 |
| GB | 0.574 | 0.526 |
| DT | 0.539 | 0.509 |
| SVC | 0.579 | 0.506 |
| LR | 0.600 | 0.505 |
| AB | 0.554 | 0.505 |

Table 2: The results of testing of LDA. RF stands for random forest, GB gradient boosting LR Logistic Regression AB Adaptive Boost DT Decision Tree and SVC support vector classifier.

Based on the table 2, we decide to use Random forest as the estimator to do feature selection as we take top 60 feature over total 81 feature such performing the more significant impact or larger feature importance score than other. However the result do not have change much, best model still Random Forest with slightly decrease to 0.60 in training set and unchange in testset.We can conclude that lda reduce the dimension of each player from 51 to 3 so that:

1. Every feature have been summarize to 3 dimension so every feature all have large impact on prediction of the model.
2. There are data having been lost during the reduction of dimension as the result of full data is much better than lda prediction ( 0.65 on testset of full data » 0.55 on test set of lda)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| draw | 0.00 | 0.00 | 0.00 | 49 |
| lose | 0.49 | 0.45 | 0.47 | 91 |
| win | 0.45 | 0.69 | 0.54 | 88 |
| accuracy | | | 0.45 | 228 |
| macro avg | 0.31 | 0.38 | 0.34 | 228 |
| weighted avg | 0.37 | 0.45 | 0.40 | 228 |

Figure 37: The classification report for the best model with lda data.

**PCA** Table 3 illustrate the AUC result of all model in ascending order with PCA strategy of reducing dimension. As seen in table 3, Random Forest still the best model with 0.63 AUC score of train set and 0.61 AUC score of test set, the second rank is Gradient Boosting with 0.63 and 0.60 respectively. SVM is still in top 3 with training AUC 0.63 and testing AUC 0.60 better than rank 4 Adaptive Boost (0.69 on training and 0.55 on testing).

| Model | Train | Test |
|---|---|---|
| RF | 0.627 | 0.614 |
| GB | 0.631 | 0.608 |
| SVC | 0.620 | 0.606 |
| AB | 0.609 | 0.596 |
| LR | 0.606 | 0.576 |
| DT | 0.527 | 0.519 |

Table 3: The results of testing of PCA. RF stands for random forest, GB gradient boosting LR Logistic Regression AB Adaptive Boost DT Decision Tree and SVC support vector classifier

Feature selection is applied with the best model Random Forest as we get 60 best feature out of total 648 feature which have highest feature importance. Beside that we do tuning manually on number of feature selected (30, 40, 50, 100, 150, 200, 400, 500), but the result is worse. The final result with feature selection witness the rise in both training AUC and testing AUC by 1% for the best model Random Forest (0.64 and 0.62) and Gradient Boosting is still the second highest AUC( 0.63 and 0.60) together with SVM( 0.62 and 0.60)

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| draw | 0.37 | 0.14 | 0.21 | 49 |
| lose | 0.58 | 0.49 | 0.53 | 91 |
| win | 0.52 | 0.77 | 0.62 | 88 |
| accuracy | | | 0.53 | 228 |
| macro avg | 0.49 | 0.47 | 0.45 | 228 |
| weighted avg | 0.51 | 0.53 | 0.50 | 228 |

Figure 38: The classification report for the best model with pca data.

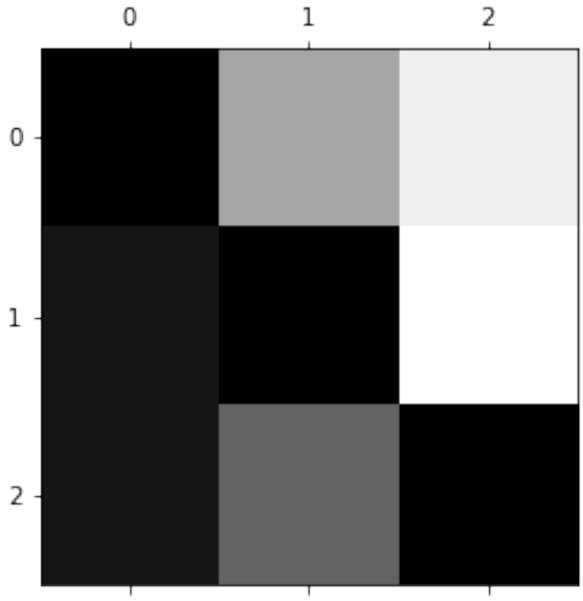


Figure 39: Error matrix of the best model (LDA). Brighter the slot, more errors; the row stands for actual classes and vice versa (0 - draw, 1 - lose, 2 - win).

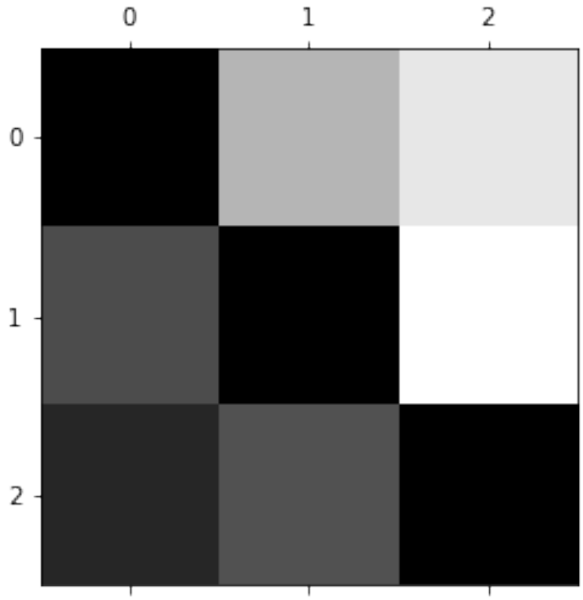


Figure 40: Error matrix of the best model (PCA). Brighter the slot, more errors; the row stands for actual classes and vice versa (labeled as above).

In the end, we do initiative report as with LDA(figure 37) and PCA(figure 38). The model perform acceptable on lose and win but not with draw ones. The final accuracy is 0.52 on LDA and 0.52 on PCA are a acceptable result if compared with the baseline model 0.41. Besides, we illustrate the confusion matrix of both dimension reduction strategy (figure 39 and figure 40) that we can conclude about the two model as in both LDA and PCA, misclassification class is draw.

# 6 Conclusion

Starting from data of teams lineups and attributes of players, we performed several data cleaning, preprocessing and dimensionality reduction techniques before integrating into a consistent, comprehensive dataset for exploratory data analysis and modelling with the goal of predicting the outcome of matches. Statistical analysis shows that EPL teams in the 2019/20, 2020/21 and 2021/22 season focus on building a strong midfield section to better possess the match, with the main playstyle of building the game from the back, where the balls mainly start from the defending position. Moreover, analysis shows that defenders and midfielders also play the key role in deciding the result of the match. Winning team in the season also generally dominates the highest winning ratio, either playing as home or away teams, which implies that the winner also outperforms the runner-ups.

To predict the outcomes of matches, we divide into two sets of data: one full data containing all players' features, and one data deduced from LDA or PCA technique, experimented using simple machine learning algorithms. Random Forest was selected to be the best classifier for our problem and can handle the overfitting well. For the LDA data, Random Forest achieved an accuracy of 45% on the test set. The resultant data from PCA technique observes better result, also with the Random Forest classifier, of 53% accuracy. The dataset containing full players' features achieved the best result of 54% accuracy owing to the fact that it does not suffer from information loss caused by dimensionality reduction. However, on any of the dataset, the classifier seems to perform worse in predicting the draw result, which is intuitive.

Possible extensions in the future is expected to involve analysing the playstyle of players in each positions, and we expect to gather more data in the past to observe the shift in how teams approach the game.

# References


[1] *Wikipedia, The Free Encyclopedia.* [Online; accessed 10-Feb-2023]. URL: `https://en.wikipedia.org/wiki/List_of_English_football_champions#References`.

[2] Richard Bellman. *Dynamic Programming.* Reprint of the 1957 edition, With a new introduction by Stuart Dreyfus. Princeton University Press, 2010.

[3] Jonas Mockus. *Bayesian approach to global optimization: Theory and applications.* Kluwer Academic, 1989.

[4] James A Hanley and Barbara J McNeil. "The meaning and use of the area under a receiver operating characteristic (ROC) curve." In: *Radiology* 143.1 (1982), pp. 29–36.