# Introduction to Math for DS Group Mini-project

Analysis of factors affecting Premier League match results


**IMDS Group 24**

Zehao Qian, Chloe Mendez, Mohammad Jamshaid Iqbal


```
                    __
               .-'||'-.
             ,'   ||   `.
            .     ||     .
           /    __||__    \
           | /`-      -`\ |
           | | 6      6 | |
           \/\____7____/\/
    .--------:\:I:II:I:/;--------.
   /          \`:I::I:`/          \
  |            `------'            |
  |             \____/             |
  |      ,    ____  _  _    ,      |
  |======|   |___ \| || |   |======|
  |======|     __) | || |_  |======|
  |~~~~~|     / __/|__   _|  |~~~~~|
  |     |\   |_____|  |_|   /|     |
   \    \|                  |/    /
    `\    \  _ _.-=""=-._ _ /    /'
      `\   ',`_)\\-++++-//(_`'  /'
        ;   (__||        ||__)  ;
         ;   ___\        /___  ;
          '.  ---/-=..=-\---  .'
            `""`          `""`
```

# Introduction to Math for DS Group Mini-project

IMDS Group 24
Zehao Qian, Chloe Mendez, Mohammad Jamshaid Iqbal

December 7, 2023

## Contents

## 1 Introduction

The English Premier League is a ranking from one to twenty of the teams who won the most matches over the season. We are investigating the variables which factor into the ranking of the premier league; however, our dataset is fairly limited. We are examining historical data from previous seasons consisting of: the games won, lost and drawn by each team as well as the total number of goals for and against their team, and the goal difference. Ideally, we would analyse a dataset which featured variables not directly related to the ranking. In an attempt to enrich our dataset we performed the Fourier Transformation on the historical data from time domain to frequency domain. Time Domain feature extraction will allow us to examine the Time Domain evolution of the data. By analysing the time variability of metrics such as points, goals scored, goals conceded, etc., we hope to discover time-related patterns, such as seasonal changes or performance trends over a specific period. Of course, the most direct relationship is between the number of games won and the teams' placement on the league table, because this is how the table is curated. However, we did find some less obvious correlation; the teams' consistency in performance, as measured by a balanced distribution of wins, draws, and losses, is associated with a higher league position. To test this hypothesis, we have made use of Principal Component Analysis (PCA), to find the most significant components, and the Entropy method (Entropy Weighting), with the intention of finding the overall weighting of the team's performance. With the goal of establishing that teams with a consistent performance history over the seasons is likely to allow them to maintain their competitive positions. PCA allows us to establish which were the most significant factors in achieving first place on the premier league table. This is appropriate for our dataset because our dataset's multicollinearity is very high, so we are using PCA to eliminate this issue.

# 2 Model Assumptions

## 2.1 Performance Assumptions (Correlation Analysis)

- The number of wins positively correlates with the final league standing.
- Teams with a higher goal difference (GF - GA) tend to achieve higher league positions.
- Drawn matches have a minimal impact on final league standings.
- Teams with a higher number of goals scored (GF) are more likely to finish in the top positions.
- The defensive performance, measured by goals against (GA), influences the team's final standing.
- The number of points earned directly correlates with the team's final position in the league.

## 2.2 Consistency Hypothesis (Principal Component Analysis)

- Consistency in performance, as measured by a balanced distribution of wins, draws, and losses, is associated with a higher league position.

PCA can help identify patterns and relationships among these variables, which can contribute to understanding the consistency in team performance.

## 2.3 Historical Performance Hypothesis (Entropy Weighting)

- Teams with a consistent performance history over the years are likely to maintain their competitive positions.

# 3 Data

## 3.1 Introduction to Match Data Set

Our dataset includes full-season data from 2007 to 2022 and partial-season English Premier League (EPL) match data up to December 7, 2023, for the 2023-2024 season. The data is organized into CSV files, each titled with the respective season "20xx-20xy". Each file contains match records for various teams.

1. **Matches Played (MP):** Represents the number of matches in which the team participated in the current season.

2. **Wins (W):** Indicates the number of victories the team achieved in the current season. In a match, winning refers to having a higher goal count than the opponent at the end of the game.

3. **Draws (D):** Represents the number of matches in which the team ended with a tied score. In a match, a draw occurs when both teams have an equal number of goals at the end.

4. **Losses (L):** Indicates the number of matches in which the team was defeated in the current season. In a match, losing refers to having fewer goals than the opponent at the end.

5. **Goals For (GF):** Represents the total number of goals scored by the team in the current season. This is an indicator reflecting the team's offensive strength.

6. **Goals Against (GA):** Represents the total number of goals conceded by the team in the current season. This is an indicator reflecting the team's defensive capabilities.

7. **Goals Difference (GD):** Represents the difference between the number of goals scored and the number of goals conceded by the team in the current season. It is calculated as Goals For - Goals Against. A positive value indicates that the team's offensive strength is greater than its defensive strength, while a negative value indicates the opposite.

8. **Points (Pts):** Represents the cumulative points earned by the team in the current season. Typically, a win awards the team 3 points, a draw awards 1 point, and a loss awards no points. Points are a crucial metric for assessing the overall competitive level of the team and are commonly used to determine team rankings in leagues.

These variables provide quantitative measures of various aspects of the team's performance in the current season, including match count, win-loss-draw record, offensive and defensive performance, and overall competitiveness assessed through point accumulation. These indicators are common and important metrics in football analysis.

Table 1: Example Dataset: 2014_15.csv

| Team | MP | Win | Draw | Loss | GF | GA | GD | Points |
|---|---|---|---|---|---|---|---|---|
| Chelsea FC | 38 | 26 | 9 | 3 | 73 | 32 | 41 | 87 |
| Manchester City FC | 38 | 24 | 7 | 7 | 83 | 38 | 45 | 79 |
| Arsenal FC | 38 | 22 | 9 | 7 | 71 | 36 | 35 | 75 |
| Manchester United FC | 38 | 20 | 10 | 8 | 62 | 37 | 25 | 70 |
| Tottenham Hotspur FC | 38 | 19 | 7 | 12 | 58 | 53 | 5 | 64 |
| Liverpool FC | 38 | 18 | 8 | 12 | 52 | 48 | 4 | 62 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |

## 3.2 Data Acquisition Framework

Instead of using the open source data set, in order to obtain up-to-date and customized data, we wrote Python scripts to grab data from the web.

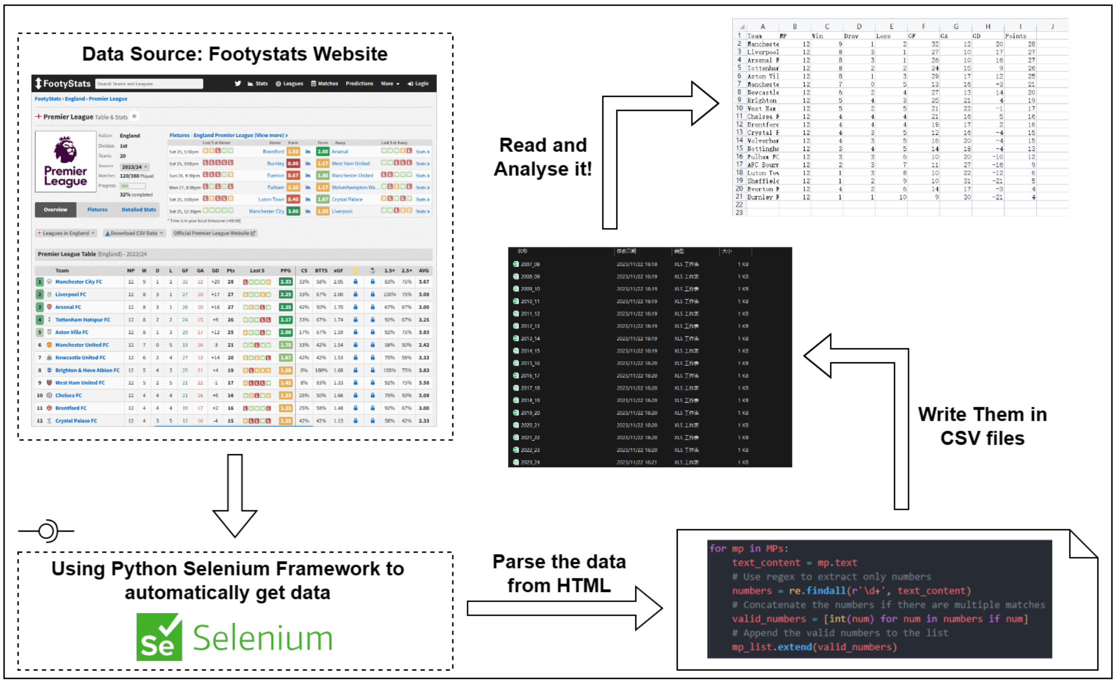


Figure 1: Flow Chart for Data Grabbing

Due to the modern structure of websites relying on user interaction for dynamic data retrieval, traditional Python web scraping libraries like requests face challenges in obtaining specific data. Therefore, we employ Selenium, a web automation testing framework. Selenium allows us to simulate human interaction by opening a browser on a computer, navigating to a website, and interacting with elements such as buttons. This assists us in navigating to the desired season page on the website. Subsequently, we use Python to parse the HTML content, extract the data mentioned earlier, and finally, employ the Pandas library to save the acquired data to a CSV file.

We put the data acquisition part code in the Appnedix Section (B).

# 4 Methods

## 4.1 Data Feature Extraction with Fourier Transform

Given that the Premier League data we crawled from the web is relatively limited, we are eager to enrich our feature set through feature extraction. To this end, we decided to treat historical match data as an information-rich signal and adopt time-domain and frequency-domain feature extraction methods to extract more key features from it, laying the foundation for a comprehensive analysis of the team's performance.

Therefore, in our research, we introduced Fourier transform, a powerful mathematical tool, to transform historical game data from time domain to frequency domain. Fourier transform is a method of converting time domain signals into frequency domain representation. Through this conversion, we are expected to reveal the underlying periodicity and frequency information in the data, providing strong support for deeper analysis and understanding.

We will give the proof of the Fourier transform in the appendix section (A). In its continuous form:

$$X(f) = \int_{-\infty}^{\infty} x(t) \cdot e^{-j2\pi ft} dt$$

Among them, X represents the frequency domain signal, and x represents the time domain signal. In the discrete form (because our data is discrete), we will use discrete Fourier transform methods such as fast Fourier transform (FFT) to calculate:

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j2\pi kn/N}$$

Time Domain feature extraction will allow us to delve into the Time Domain evolution of the data, revealing overall trends in the team over the past few years. By analyzing the time variability of metrics such as points, goals scored, goals conceded, etc., we can hopefully discover time-related patterns, such as seasonal changes or performance trends over a specific period.

At the same time, frequency domain feature extraction will help us understand the periodicity and frequency present in the data. This approach will help identify recurring seasonal patterns, allowing us to gain a more complete understanding of how a team's performance changes throughout the season. We will also explore whether specific events have a frequency-domain impact on team performance.

By converting historical match data into time and frequency domain features, we expect to be able to mine deeper information and provide a richer and more accurate feature set for our data-driven models to better interpret and predict Premier League matches. The team's performance. Here are the Time and Frequency Domain Features we will use:

Table 2: Time Domain Features and Formulas

| Time Domain Feature | Formula for Time Domain Feature |
|---|---|
| Mean | $\text{mean} = \frac{1}{N}\sum_{i=1}^{N} x_i$ |
| Standard Deviation | $\text{std\_dev} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \text{mean})^2}$ |
| Root Mean Square (RMS) | $\text{rms} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$ |
| Skewness | $\text{skewness} = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - \text{mean})^3}{\left(\frac{1}{N}\sum_{i=1}^{N}(x_i - \text{mean})^2\right)^{3/2}}$ |
| Kurtosis | $\text{kurtosis} = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - \text{mean})^4}{\left(\frac{1}{N}\sum_{i=1}^{N}(x_i - \text{mean})^2\right)^{2}} - 3$ |
| Max Value | $\text{max\_value} = \max(\text{signal\_array})$ |
| Min Value | $\text{min\_value} = \min(\text{signal\_array})$ |
| Median | $\text{median} = \text{np.median}(\text{signal\_array})$ |
| Zero Crossing Rate | $\text{zero\_crossing\_rate} = \frac{\sum_{i=1}^{N-1}(\text{sign}(x_{i+1}) - \text{sign}(x_i)) \neq 0}{N}$ |

Table 3: Frequency Domain Features and Formulas

| Frequency Domain Feature | Formula for Frequency Domain Feature |
|---|---|
| Dominant Frequency | $\text{dominant\_frequency} = \arg\max(\text{magnitude\_spectrum})$ |
| Max Frequency Magnitude | $\text{max\_frequency\_magnitude} = \max(\text{magnitude\_spectrum})$ |
| Power Spectral Density | $\text{power\_spectral\_density} = \frac{1}{N}\sum_{i=1}^{N} \text{magnitude\_spectrum}^2$ |
| Spectral Entropy | $\text{spectral\_entropy} = \text{entropy}(\text{magnitude\_spectrum})$ |
| Total Power | $\text{total\_power} = \sum_{i=1}^{N} x_i^2$ |
| Centroid Frequency | $\text{centroid\_frequency} = \frac{\sum_{i=1}^{N} i \cdot \text{magnitude\_spectrum}[i]}{\sum_{i=1}^{N} \text{magnitude\_spectrum}[i]}$ |

We apply these formulas to our data (historical matches). 'Magnitude_spectrum' is the magnitude of the spectrum calculated by Fourier transform.

## 4.2 Correlation Analysis

After feature extraction, we are preparing to conduct a thorough analysis of the relationship between various team indicators (MP, Win, Draw, Loss, GF, GA, GD and other features generated by last part) and the final score (Points) through correlation analysis. To choose an appropriate correlation analysis method, we will compare the characteristics of the Pearson correlation coefficient and the Spearman rank correlation coefficient, ultimately selecting the Spearman rank correlation coefficient for correlation analysis in English Premier League football.

After comparing the above characteristics, we have decided to choose the Spearman rank correlation coefficient as our correlation analysis method. This is because in the context of English Premier League football matches, our data may not follow a normal distribution,

Table 4: Characteristics of Pearson and Spearman Correlation Coefficients

| Characteristics | Pearson Correlation Coefficient | Spearman Rank Correlation Coefficient |
|---|---|---|
| Calculation | $r_{xy} = \frac{\sum(X_i-\bar{X})(Y_i-\bar{Y})}{\sqrt{\sum(X_i-\bar{X})^2 \cdot \sum(Y_i-\bar{Y})^2}}$ | $\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$ |
| Data Type | Continuous variables | Ordinal variables and non-linear relationships |
| Linear Assumption | Assumes a linear relationship between variables | Does not make a specific linear assumption about the relationship |
| Applicability | Works well when data is approximately normally distributed | Applicable to ordinal variables, non-linear relationships, or when data distribution does not follow a normal distribution |
| Outlier Sensitivity | Sensitive to outliers | Relatively insensitive to outliers as it is based on ranks |
| Interpretation | Emphasizes the strength and direction of linear relationships | Focuses more on the ordinal relationship between variables |

and the Spearman rank correlation coefficient is more robust to non-linear relationships and ordinal variables, while being relatively insensitive to outliers. This makes it more suitable for our research purposes.

In the report, we will use the Spearman rank correlation coefficient to explore the relationship between various team indicators and the final score, providing a more comprehensive understanding of the factors influencing different aspects of English Premier League football matches.

## 4.3 Principal Component Analysis

Principal Component Analysis (PCA) is a technique for data dimensionality reduction and feature extraction. It achieves this by identifying the principal directions (principal components) in the data to reduce its dimensionality. Below are the detailed calculation steps for PCA, introducing an example dataset:

Assume we have the following dataset:

Table 5: Example Dataset

| MP | Win | Draw | Loss | GF | GA | GD | Others |
|---|---|---|---|---|---|---|---|
| 38 | 27 | 6 | 5 | 80 | 22 | 58 | ... |
| 38 | 25 | 10 | 3 | 65 | 26 | 39 | ... |
| 38 | 24 | 11 | 3 | 74 | 31 | 43 | ... |
| 38 | 21 | 13 | 4 | 67 | 28 | 39 | ... |
| 38 | 19 | 8 | 11 | 55 | 33 | 22 | ... |

Steps:

1. Standardize Data:

Standardize each feature, making its mean 0 and standard deviation 1. The standardization formula is:

$$Z = \frac{(X - \bar{X})}{\sigma}$$

where $X$ is the original data, $\bar{X}$ is the mean, and $\sigma$ is the standard deviation. Applying this to the dataset yields the standardized data.

2. Compute Covariance Matrix:

The covariance matrix is the covariance matrix of the standardized data. The covariance matrix's formula is:

$$\mathrm{Cov}(X, Y) = \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{n - 1}$$

where $X$ and $Y$ are two features, $\bar{X}$ and $\bar{Y}$ are their means. After calculating the covariance matrix, we get:

3. Compute Eigenvalues and Eigenvectors:

Perform eigenvalue decomposition on the covariance matrix to obtain eigenvalues and their corresponding eigenvectors. Eigenvalues represent variance in the data, and eigenvectors are the directions of principal components.

4. Select Principal Components:

Based on the magnitude of eigenvalues, choose the number of principal components to retain. Typically, you might select components that capture a certain percentage of variance, such as 90%.

5. Build Projection Matrix:

Compose a matrix with the selected eigenvectors as columns. This matrix serves as the projection matrix, mapping the original data into the new principal component space.

6. Project to New Principal Component Space:

Multiply the standardized data by the projection matrix to obtain the reduced-dimensional data.

## 5 Conclusions

## A Appendix I: A breif proof for Fourier Transform

Equation

## B Appendix II: Premier League Data Fetch Scripts

```python
from selenium import webdriver
from selenium.webdriver.common.by import By
import re
from selenium.webdriver.support.ui import WebDriverWait
# from selenium.webdriver.support.select import Select
from selenium.webdriver.support import expected_conditions as
    EC
import time
import csv

options = webdriver.EdgeOptions()
options.add_experimental_option("detach", True)
driver = webdriver.Edge()
driver.maximize_window()
driver.get('https://footystats.org/england/premier-league')

start_year = 2007
end_year = 2023
Years = [
    f'{year}/{str(year+1)[-2:]}' for year in range(start_year,
        end_year + 1)]
# year = '2022/23'


for year in Years:
    select = driver.find_element(By.CLASS_NAME, "drop-down-
        parent.fl.boldFont")
    select.click()
    time.sleep(2)
    # Replace 'your_data_hash_value' with the specific value
        you want to select
    # chooseSeason = driver.find_element(By.)
    # get element
    element = WebDriverWait(driver, 10).until(
        EC.element_to_be_clickable((By.LINK_TEXT, year))
    )
    element.click()
```

```
    time.sleep(2)
    # mp, win, draw, loss, gf, ga, gd
    # Find an element by its class (replace 'element_class'
       with the actual class of the element on the webpage)
    TEAMs = driver.find_elements(
        By.CLASS_NAME, 'bold.hover-modal-parent.hover-modal-
           ajax-team')
    MPs = driver.find_elements(By.CLASS_NAME, 'mp')
    WINs = driver.find_elements(By.CLASS_NAME, 'win')
    DRAWs = driver.find_elements(By.CLASS_NAME, 'draw')
    LOSSs = driver.find_elements(By.CLASS_NAME, 'loss')
    GFs = driver.find_elements(By.CLASS_NAME, 'gf')
    GAs = driver.find_elements(By.CLASS_NAME, 'ga')
    GDs = driver.find_elements(By.CLASS_NAME, 'gd')
    POINTs = driver.find_elements(By.CLASS_NAME, 'points.bold')
    # Get text content from the element
    team_list = []
    mp_list = []
    win_list = []
    draw_list = []
    loss_list = []
    gf_list = []
    ga_list = []
    gd_list = []
    point_list = []

    for team in TEAMs:
        text_content = team.text
        team_list.append(text_content)

    for mp in MPs:
        text_content = mp.text
        # Use regex to extract only numbers
        numbers = re.findall(r'\d+', text_content)
        # Concatenate the numbers if there are multiple matches
        valid_numbers = [int(num) for num in numbers if num]
        # Append the valid numbers to the list
        mp_list.extend(valid_numbers)

    for win in WINs:
        text_content = win.text
        # Use regex to extract only numbers
        numbers = re.findall(r'\d+', text_content)
```

```
            # Concatenate the numbers if there are multiple matches
            valid_numbers = [int(num) for num in numbers if num]
            # Append the valid numbers to the list
            win_list.extend(valid_numbers)

        for draw in DRAWs:
            text_content = draw.text
            # Use regex to extract only numbers
            numbers = re.findall(r'\d+', text_content)
            # Concatenate the numbers if there are multiple matches
            valid_numbers = [int(num) for num in numbers if num]
            # Append the valid numbers to the list
            draw_list.extend(valid_numbers)

        for loss in LOSSs:
            text_content = loss.text
            # Use regex to extract only numbers
            numbers = re.findall(r'\d+', text_content)
            # Concatenate the numbers if there are multiple matches
            valid_numbers = [int(num) for num in numbers if num]
            # Append the valid numbers to the list
            loss_list.extend(valid_numbers)

        for gf in GFs:
            text_content = gf.text
            # Use regex to extract only numbers
            numbers = re.findall(r'\d+', text_content)
            # Concatenate the numbers if there are multiple matches
            valid_numbers = [int(num) for num in numbers if num]
            # Append the valid numbers to the list
            gf_list.extend(valid_numbers)

        for ga in GAs:
            text_content = ga.text
            # Use regex to extract only numbers
            numbers = re.findall(r'\d+', text_content)
            # Concatenate the numbers if there are multiple matches
            valid_numbers = [int(num) for num in numbers if num]
            # Append the valid numbers to the list
            ga_list.extend(valid_numbers)

        for gd in GDs:
            text_content = gd.text
```

```
            # Use regex to extract only numbers
            numbers = re.findall(r'-?\d+', text_content)
            # Concatenate the numbers if there are multiple matches
            valid_numbers = [int(num) for num in numbers if num]
            # Append the valid numbers to the list
            gd_list.extend(valid_numbers)

        for point in POINTs:
            text_content = point.text
            # Use regex to extract only numbers
            numbers = re.findall(r'\d+', text_content)
            # Concatenate the numbers if there are multiple matches
            valid_numbers = [int(num) for num in numbers if num]
            # Append the valid numbers to the list
            point_list.extend(valid_numbers)

        # Print the list of extracted numbers
        print('TEAM␣List␣is:', team_list)
        print('MP␣List␣is:', mp_list)
        print('WIN␣List␣is:', win_list)
        print('DRAW␣List␣is:', draw_list)
        print('Loss␣List␣is:', loss_list)
        print('GF␣List␣is:', gf_list)
        print('GA␣List␣is:', ga_list)
        print('GD␣List␣is:', gd_list)
        print('POINT␣List␣is:', point_list)

        # CSV file
        year_file = year.replace('/', '_')
        csv_file_path = './' + 'Data/' + year_file + '.csv'

        # write data to csv
        with open(csv_file_path, mode='w', newline='', encoding='
           utf-8') as file:
            writer = csv.writer(file)

            # edit header
            header = ['Team', 'MP', 'Win', 'Draw',
                        'Loss', 'GF', 'GA', 'GD', 'Points']
            writer.writerow(header)

            # write data
            for i in range(len(team_list)):
```

```
            row = [team_list[i], mp_list[i], win_list[i],
               draw_list[i],
                    loss_list[i], gf_list[i], ga_list[i],
                       gd_list[i], point_list[i]]
            writer.writerow(row)

    print(f'Data has been written to {csv_file_path}')

driver.quit()
```