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 8, 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.

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 Signal processing, Spearman's Rank Corelation Coefficient, and Principal Component Analysis 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.

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 hoped to discover time-related patterns, such as seasonal changes or performance trends over a specific period. Having completed the Fourier Transformation we used correlation analysis, we will be using Spearman's Rank Corelation Coefficient to explore the relationship between team indicators, points, goals, losses, etc.

PCA allows us to establish which were the most significant factors in achieving first place on the premier league table, by finding which component explains the majority of the variance between the teams. This is appropriate for our dataset because our dataset's multicollinearity is very high, so we are using PCA in an effort 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.

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.

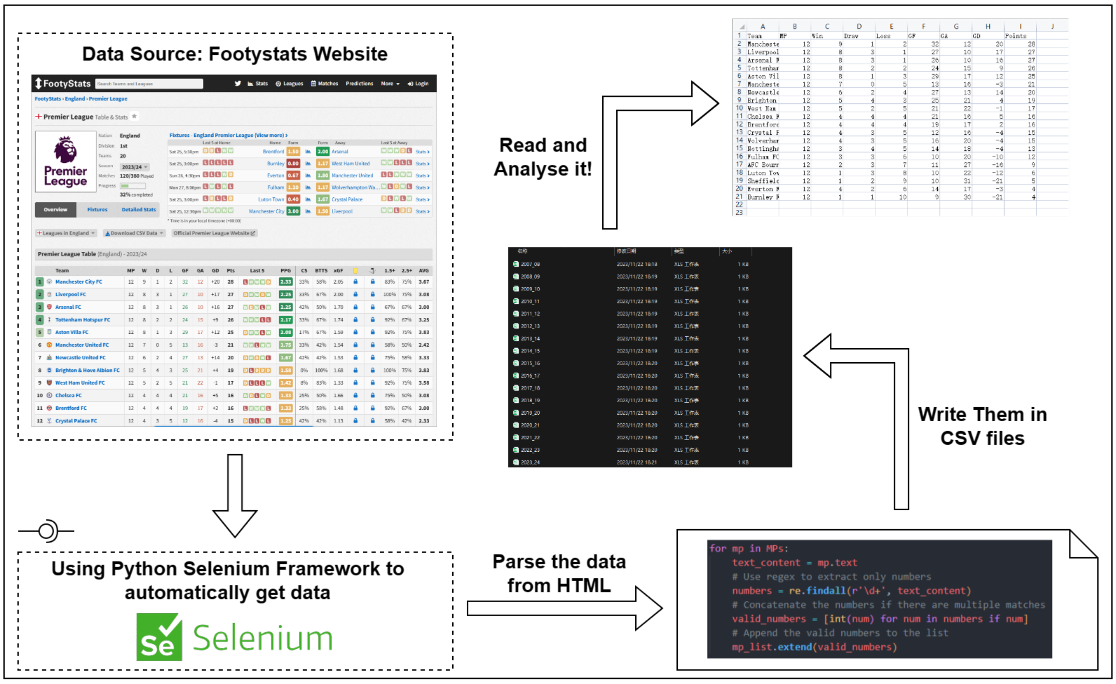

Figure 1: Flow Chart for Data Grabbing

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 (C).

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 σ 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.

let:

$$|\lambda E - A| = 0 \Rightarrow \lambda_1, \lambda_2, ..., \lambda_n$$

Bring $\lambda_1, \lambda_2, ..., \lambda_n$ back to matrix $(\lambda E - A)$ and get the Eigenvector:

4. Select Principal Components:

Based on the magnitude of eigenvalues, choose the number of principal components to retain. Typically, we 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.

In the appendix section of the report, we have included a detailed explanation and implementation of the PCA algorithm using the Scikit-Learn library. We provide corresponding Python code along with a comprehensive breakdown of each step. F

5 Conclusions

5.1 Results Analysis

Unveiling Team Prowess Through Fourier Transform Analysis

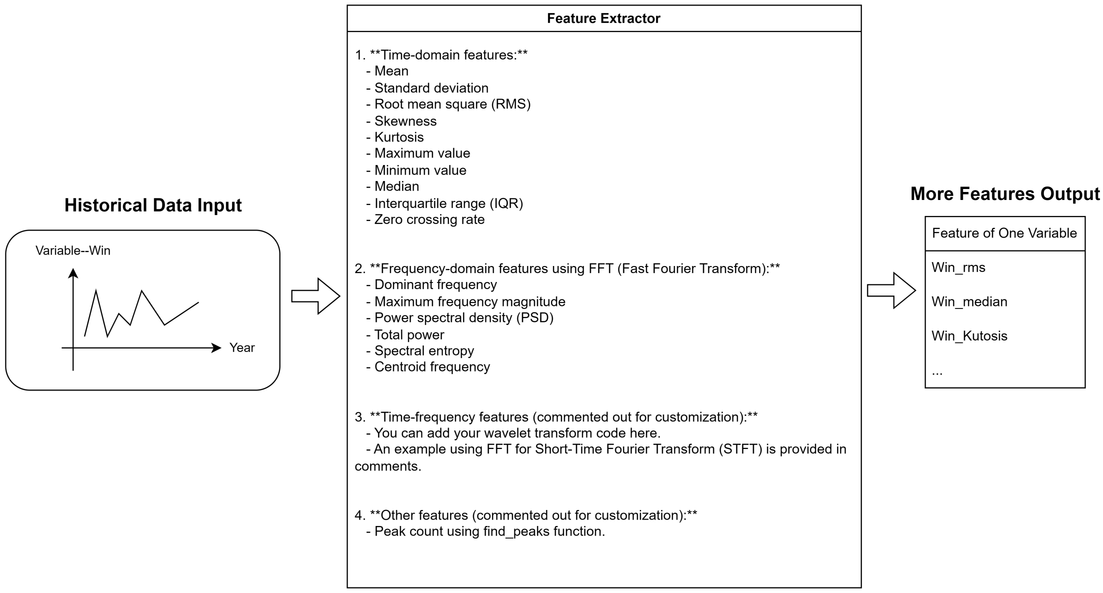

Figure 2: Feature Extraction

Historical data serves as a manifestation of a team's capabilities. Therefore, we consider each set of data from the past 15 years of the English Premier League as a unique signal. By applying Fourier transforms in both the time and frequency domains, we aim to extract hidden information from these historical datasets, treating each team's performance as a distinctive signal. We employ feature extraction formulas to analyze these signals, attempting to unveil latent information that reflects the teams' strengths. This approach helps delve into patterns and trends in a team's past performances, providing a more comprehensive understanding of their athletic prowess.

Our initial dataset comprises only 9 columns from the 2023-24 season. Through feature extraction on historical season data, we have expanded the dataset to include 121 columns. This provides us with greater flexibility for the upcoming steps of correlation analysis and PCA.

Analysis of Correlation between other variables and Points

Through the feature extraction of historical Premier League season data in the previous section, we incorporated more variables into the Spearman correlation analysis function and discovered many intriguing conclusions. Here are some data points with significant correlations. We extracted variables with correlations greater than 0.65:

Variable	Spearman Correlation
Win	0.9533104557457376
GD	0.8948732427304006
GF	0.8586225905364607
GF_max_value	0.8215219412798919
GD_max_value	0.7745621530420769
Points_rms	0.7518113287871696
GF_mean	0.7394017882844927
Points_mean	0.7362994031588235
GF_rms	0.7300946329074851
Win_rms	0.7269922477818159
GF_median	0.7232290137967814
Points_max_value	0.7230852735691926
GD_mean	0.7197533491552544
Win_max_value	0.7174050173932447
GF_power_spectral_density	0.697002524900347
GF_total_power	0.697002524900347
Win_mean	0.6876953695233393
GF_max_frequency_magnitude	0.681490599272001
Points_power_spectral_density	0.6649445452684319
Points_max_frequency_magnitude	0.6649445452684319
Points_total_power	0.6649445452684319
Win_total_power	0.6649445452684319
Win_power_spectral_density	0.6649445452684319
Points_median	0.6599275122106363
GF_min_value	0.6542250749488004
GF_std_dev	0.651500876390532
Draw_centroid_frequency	0.6504667480153089

Variable	Spearman Correlation
Loss	-0.9180829639568062
GA	-0.7707937688076131
Loss_min_value	-0.7688337623041291
Loss_mean	-0.7387248271014586
Loss_rms	-0.7145827072791391
Loss_median	-0.712066382380961
Loss_max_value	-0.6820301277059824

Through correlation analysis, we have uncovered some surprising findings. Firstly, there is a positive correlation between the number of wins (Win) and the final score (Points), while the number of losses (Loss) shows a negative correlation with the final score. This aligns well with our intuitive expectations.

Furthermore, we observed a significant correlation between the derived features from our historical data processing and the final score. For instance, variables obtained through time-domain analysis such as GF_max_value, GD_max_value, Loss_mean, and Loss_median, as well as those obtained through frequency-domain analysis like GF_power_spectral_density and GF_max_frequency_magnitude, exhibit a strong correlation with the final score.

This discovery suggests that our feature engineering methods go beyond simple processing of raw data collected from websites. These derived features may offer valuable insights when predicting the performance of football teams.

5.2 Model Application

In terms of model application, the insights derived from feature extraction and in-depth analysis of variable correlations in historical data can be applied to football team management decisions. Specifically, these analytical results can assist team managers in better understanding the team's strengths and weaknesses, enabling them to formulate more effective game strategies. For example, a deeper understanding of key variables such as wins (Win) and losses (Loss) can provide a basis for making more informed decisions during matches.

Simultaneously, the model can also provide assistance in football betting analysis. By understanding the relationship between various features and the final score, fans and bettors can more accurately assess a team's performance in a match, making more informed choices in football betting.

Overall, through in-depth analysis of football match data, we not only offer strategic recommendations for teams but also provide more precise information for fans and bettors, enhancing their confidence and enjoyment in following and participating in football matches.

5.3 Limitation

Our analysis focused on assessing the importance of various variables. While it can be a valuable tool to aid in the analysis of the English Premier League, it is not a predictive model. Using PCA and correlation analysis to predict which teams will win in the future is a challenging task. Our work should be seen as part of a holistic analysis, as a reference rather than an explicit forecasting tool.

5.4 Future Work

Unfortunately, due to the lack of independent variables in our dataset, we cannot derive much from the models/analysis that we did not already know. In future, researchers might consider amassing datasets featuring variables which are not part of the calculation of the ranking, such as red cards, yellow cards, penalties and net worth of the players on the team in that season. Then performing this analysis could find which of the variables factored most significantly into the ranking of the team.

At the same time, since our team now only has two members and has only completed the model and code parts of the PCA part, the results of the PCA analysis have not appeared in this report. If possible, we hope to complete this part.

A A breif proof for Fourier Transform

The continuous Fourier Transform of a function $f(t)$ is given by:

$$F(\omega) = \int_{-\infty}^{\infty} f(t) \cdot e^{-i\omega t} \, dt$$

To prove this, we start with the Fourier series representation of a periodic function. Let T be the period of $f(t)$. The Fourier series expansion is given by:

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{i\omega_n t}$$

Here, c_n are the Fourier coefficients, and $\omega_n = \frac{2\pi n}{T}$ are the angular frequencies.

Now, consider the limit as T approaches infinity, turning the sum into an integral:

$$f(t) = \lim_{T\to\infty} \sum_{n=-\infty}^{\infty} c_n e^{i\omega_n t}$$

This leads to the continuous Fourier Transform integral:

$$F(\omega) = \lim_{T\to\infty} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) e^{-i\omega t} \, dt$$

As T approaches infinity, the limits of integration become $-\infty$ to ∞:

$$F(\omega) = \lim_{T\to\infty} \int_{-\infty}^{\infty} f(t) e^{-i\omega t} \, dt$$

This is essentially the definition of the continuous Fourier Transform. The limit is introduced to handle functions that are not strictly periodic.

The rigorous proof involves showing that this limit exists for a wide class of functions and exploring the convergence properties. It requires knowledge of mathematical analysis and complex analysis.

B Data Set

Due to the large number of datasets and the number of columns, we uploaded the data to GitHub, and below are links to the various datasets.

B.1 Data Original Source

The data set we got from FootyStats.org and we saved them with csv format.

Click the Hyperlink here and Review the orginal data set.

B.2 The processed dataset

A data set for combining the 2023-24 season data and historical features.

Data set for caculating the Correlation between variables and Points

Data set for caculating PCA

B.3 Dataset for Saving Results

Results about Correlation between variables and Points

Results about PCA

C 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)
```

```python
    # 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()
```

D Feature Extract Function

D.1 extractFeatures.py

```python
import pandas as pd
import numpy as np
from scipy.stats import skew, kurtosis
from scipy.fftpack import fft
from scipy.signal import find_peaks
from scipy.stats import entropy


def extract_features(signal):
    features = {}

```

```python
    # Convert the column to numeric, handling non-numeric values
    signal_numeric = pd.to_numeric(signal, errors='coerce')

    # Replace or remove specific non-numeric values
    signal_numeric.replace('ALIGNED', np.nan, inplace=True)

    # Remove NaN values or use interpolation as needed
    signal_numeric.dropna(inplace=True)

    # Convert to NumPy array
    signal_array = signal_numeric.to_numpy()

    # Check if the array is not empty
    if len(signal_array) > 0:
        # Time-domain features
        features['mean'] = np.mean(signal_array)
        features['std_dev'] = np.std(signal_array)
        features['rms'] = np.sqrt(np.mean(np.square(signal_array)))
        features['skewness'] = skew(signal_array)
        features['kurtosis'] = kurtosis(signal_array)
        features['max_value'] = np.max(signal_array)
        features['min_value'] = np.min(signal_array)
        features['median'] = np.median(signal_array)
        features['iqr'] = np.percentile(
            signal_array, 75) - np.percentile(signal_array, 25)
        features['zero_crossing_rate'] = np.sum(
            np.diff(np.sign(signal_array)) != 0) / len(signal_array)

        # Frequency-domain features using FFT
        try:
            fft_result = fft(signal_array)
            magnitude_spectrum = np.abs(fft_result)

            features['dominant_frequency'] = np.argmax(magnitude_spectrum)
            features['max_frequency_magnitude'] = np.max(magnitude_spectrum)

            # Statistical measures
            # features['auto_correlation'] = np.correlate(
```

```
            #     signal_array, signal_array, mode='full')

            # Frequency-domain features
            features['power_spectral_density'] = np.mean(
                np.square(magnitude_spectrum))
            features['total_power'] = np.sum(np.square(signal_array))
            features['spectral_entropy'] = entropy(magnitude_spectrum)
            features['centroid_frequency'] = np.sum(np.arange(
                len(magnitude_spectrum)) * magnitude_spectrum) / np.sum(magnitude_spectrum)

            # Time-frequency features
            # Add your wavelet transform code here
            # Example using FFT for STFT
            # features['wavelet_coefficients'] = []
            # features['stft'] = np.abs(np.fft.fftshift(
            # np.fft.fft(signal_array)))

            # Other features
            # features['peaks_count'], _ = find_peaks(signal_array)

        except ValueError as e:
            print(f"Error in FFT calculation: {e}")

    return features
```

D.2 Using Data Extraction Function

```

import os
import pandas as pd
from extractFeatures import extract_features

# Folder path containing all CSV files
folder_path = "./Data"

# Target team
target_team = "Everton FC"
target_teams = ['Manchester City FC', 'Liverpool FC', 'Arsenal FC',
```

```
                    'Tottenham Hotspur FC', 'Aston Villa FC',
                    'Manchester United FC', 'Newcastle United FC',
                    'Brighton & Hove Albion FC', 'West Ham United FC',
                    'Chelsea FC', 'Brentford FC', 'Crystal Palace FC',
                    'Wolverhampton Wanderers FC', 'Nottingham Forest FC',
                    'Fulham FC', 'AFC Bournemouth', 'Luton Town FC',
                    'Sheffield United FC', 'Everton FC', 'Burnley FC']
# List of all CSV file names
csv_files = [
    '2007_08.csv', '2008_09.csv', '2009_10.csv', '2010_11.csv',
    '2011_12.csv', '2012_13.csv', '2013_14.csv', '2014_15.csv',
    '2015_16.csv', '2016_17.csv', '2017_18.csv', '2018_19.csv',
    '2019_20.csv', '2020_21.csv', '2021_22.csv', '2022_23.csv'
]



def get_result_df(target_team):
    # Create an empty DataFrame to store the extracted data
    result_df = pd.DataFrame(
        columns=["File", "Team", "MP", "Win",
                 "Draw", "Loss", "GF", "GA",
                 "GD", "Points"]
    )
    # Loop through each CSV file
    for csv_file in csv_files:
        # Build the full path of the CSV file
        file_path = os.path.join(folder_path, csv_file)

        # Read the CSV file
        df = pd.read_csv(file_path)

        # Extract data for the target team
        team_data = df[df["Team"] == target_team]

        # If data for the target team is found, add it to the result DataFrame
        if not team_data.empty:
```

```
            # Use pd.concat to add the target team's data from the current file to the result DataFrame,
            # and add the file name column
            result_df = pd.concat(
                [result_df, team_data.assign(File=csv_file)], ignore_index=True)

    return result_df

# print(result_df)
# result_df.to_csv("output.csv", index=False)

selected_columns = ['Win','Draw','Loss','GF','GA','GD','Points']

dfs = []

for target_team in target_teams:
    result_df = get_result_df(target_team)


    features_dict = {}
    for column in selected_columns:
        selected_data = result_df[column]
        features_column = extract_features(selected_data)
        features_dict[column] = features_column

    # Flatten the nested structure and convert to a DataFrame
    flattened_features = {}
    for column, feature_dict in features_dict.items():
        for key, value in feature_dict.items():
            flattened_features[f'{column}_{key}'] = value

    flattened_df = pd.DataFrame([flattened_features])

    # Append the DataFrame to the list
    dfs.append(flattened_df)

# Concatenate all DataFrames into a single DataFrame

result_df = pd.concat(dfs, ignore_index=True)

df_23_24 = pd.read_csv('./Data/2023_24.csv')
```

```python
result = pd.concat([df_23_24, result_df], axis=1)
# Export the result to a CSV file
result.to_csv("2023_24_General.csv", index=False)
```

E Correlation Calculating

```python
# Import the pandas library
import pandas as pd

# Read the CSV file
# Replace '2023_24_Processed.csv' with the actual file name and path
df = pd.read_csv('./Results/2023_24_Processed.csv')

# Define the target column for Spearman correlation
target_column = 'Points'

# Calculate Spearman correlation between the target column and all other columns
correlation_result = df.corrwith(df[target_column], method='spearman')

# Create a DataFrame containing the correlation results
result_df = pd.DataFrame({
    'Positive Correlation': correlation_result[correlation_result > 0].sort_values(ascending=False),
    'Negative Correlation': correlation_result[correlation_result < 0].sort_values(ascending=True)
})

# Print the results
print(result_df)

# Save the results to a CSV file
result_df.to_csv('./Results/Cor_result.csv', index=True)

# Extract the variables with absolute correlation greater than 0.6
positive_correlation_list = correlation_result[(correlation_result > 0) & (correlation_result.abs() > 0.65)].sort_values(ascending=False)
```

```python
negative_correlation_list = correlation_result[(correlation_result < 0) & (correlation_result.abs() > 0.65)].sort_values(ascending=True)

# Print the lists of variables with absolute correlation greater than 0.6
print(positive_correlation_list)
print(negative_correlation_list)

positive_correlation_list.to_csv('./Results/Positive_Cor_result.csv', index=True)
negative_correlation_list.to_csv('./Results/Negative_Cor_result.csv', index=True)


positive_correlation_variables = list(positive_correlation_list.index)
negative_correlation_variables = list(negative_correlation_list.index)


print(positive_correlation_variables)
print(negative_correlation_variables)
```

F PCA Analysis Implementation

```python
# Import the pandas library
import pandas as pd
from sklearn.decomposition import PCA


# Read the CSV file
# Replace '2023_24_Processed.csv' with the actual file name and path
df = pd.read_csv('./Results/2023_24_Processed.csv')


select_col = ['Win', 'GD', 'GF', 'GF_max_value', 'GD_max_value', 'Points_rms',
                 'GF_mean', 'Points_mean', 'GF_rms', 'Win_rms', 'GF_median', 'Points_max_value',
```

```
                'GD_mean', 'Win_max_value', 'GF_power_spectral_density', 'GF_total_power',
                'Win_mean', 'GF_max_frequency_magnitude', 'Points_power_spectral_density',
                'Points_max_frequency_magnitude', 'Points_total_power', 'Win_total_power',
                'Win_power_spectral_density', 'Points_median', 'GF_min_value', 'GF_std_dev',
                'Draw_centroid_frequency', 'Loss', 'GA', 'Loss_min_value', 'Loss_mean',
                'Loss_rms', 'Loss_median', 'Loss_max_value']


selected_columns = df[select_col]
selected_columns.to_csv('./Results/2023_24_Processed_PCA.csv', index=False)

pca = PCA()
pca_result = pca.fit_transform(selected_columns)

pca_df = pd.DataFrame(data=pca_result, columns=[
                        f'PC{i+1}' for i in range(pca_result.shape[1])])
# result_df = pd.concat([selected_columns.reset_index(drop=True), pca_df], axis=1)

# result_df.to_csv('./pca_result.csv', index=False)
pca_df.to_csv('./Results/pca_result.csv', index=False)
```