Football Match Forecasting

October 29, 2023

1 Football Match Forecasting: League Analysis across Various Countries

1.0.1 Yunus Kocabey

1.1 Introduction

Football is not just a sport; it embodies the spirit of competition, unpredictability, and the thrill of every goal scored. In this data-driven exploration of top football leagues across the globe, we delve into the exhilarating realm of **match predictions using the powerful Poisson distribution**. Leveraging a blend of web scraping techniques and statistical computations, we harness data from premier leagues like the English Premier League, La Liga, Serie A, and others. *** By dissecting team performance metrics, including goal averages, attacking and defensive strengths, and league-wide goal tendencies, we uncover valuable insights that fuel our dynamic match outcome forecasts. Join us on this enthralling journey through the statistical landscape of football, where every calculation and prediction serves as a testament to the endless excitement encapsulated within the beautiful game.

1.1.1 Gathering League Statistics from FBREF

The variables `Premier_table` and `Premier_fixture` are specifically tailored to reference the statistical table and fixture data for the English Premier League. These variables act as the primary conduits for extracting crucial performance metrics and match information from the popular sports data platform, fbref.com. The comprehensive statistical insights derived from this data serve as the fundamental building blocks for subsequent data processing and intricate statistical computations within the project. By leveraging the rich repository of team statistics and match results provided by fbref.com, the project can provide nuanced analyses and accurate predictions, facilitating a deeper understanding of the intricacies within the English Premier League and its teams' performances.

> The data for the project is sourced from fbref.com, a comprehensive sports statistics website providing detailed information on various football leagues worldwide. FBREF website

[1]:
```python
# Importing necessary libraries
import pandas as pd
import numpy as np
```

```
from itertools import product
from scipy.stats import poisson
```

[2]:
```
# URLs for data sources of various football leagues
# Each league's statistical data and fixture information is retrieved from␣
 ↪fbref.com
Sweden_table = 'https://fbref.com/en/comps/29/
 ↪Allsvenskan-Stats#results100861_home_away::none'
Sweden_fixture = 'https://fbref.com/en/comps/29/schedule/
 ↪Allsvenskan-Scores-and-Fixtures'

Norway_table = 'https://fbref.com/en/comps/28/
 ↪Eliteserien-Stats#results100821_home_away::none'
Norway_fixture = 'https://fbref.com/en/comps/28/schedule/
 ↪Eliteserien-Scores-and-Fixtures'

Finland_table = 'https://fbref.com/en/comps/43/
 ↪Veikkausliiga-Stats#results100780_home_away::none'
Finland_fixture = 'https://fbref.com/en/comps/43/schedule/
 ↪Veikkausliiga-Scores-and-Fixtures'

Mexico_table = 'https://fbref.com/en/comps/31/
 ↪Liga-MX-Stats#results107430_home_away::none'
Mexico_fixture = 'https://fbref.com/en/comps/31/schedule/
 ↪Liga-MX-Scores-and-Fixtures'

Turkey_table = 'https://fbref.com/en/comps/26/
 ↪Super-Lig-Stats#results107401_home_away::none'
Turkey_fixture = 'https://fbref.com/en/comps/26/schedule/
 ↪Super-Lig-Scores-and-Fixtures'

Premier_table = 'https://fbref.com/en/comps/9/
 ↪Premier-League-Stats#results107281_home_away::none'
Premier_fixture = 'https://fbref.com/en/comps/9/schedule/
 ↪Premier-League-Scores-and-Fixtures'

France_table = 'https://fbref.com/en/comps/13/
 ↪Ligue-1-Stats#results107321_home_away::none'
France_fixture = 'https://fbref.com/en/comps/13/schedule/
 ↪Ligue-1-Scores-and-Fixtures'

Germany_table = 'https://fbref.com/en/comps/20/
 ↪Bundesliga-Stats#results107371_home_away::none'
Germany_fixture = 'https://fbref.com/en/comps/20/schedule/
 ↪Bundesliga-Scores-and-Fixtures'
```

```
Italy_table = 'https://fbref.com/en/comps/11/
 ↪Serie-A-Stats#results107301_home_away::none'
Italy_fixture = 'https://fbref.com/en/comps/11/schedule/
 ↪Serie-A-Scores-and-Fixtures'

Spain_table = 'https://fbref.com/en/comps/12/
 ↪La-Liga-Stats#results107311_home_away::none'
Spain_fixture = 'https://fbref.com/en/comps/12/schedule/
 ↪La-Liga-Scores-and-Fixtures'

Brazil_table = 'https://fbref.com/en/comps/24/
 ↪Serie-A-Stats#results100721_home_away::none'
Brazil_fixture = 'https://fbref.com/en/comps/24/schedule/
 ↪Serie-A-Scores-and-Fixtures'

Columbia_table = 'https://fbref.com/en/comps/41/
 ↪Primera-A-Stats#results100760_home_away::none'
Columbia_fixture = 'https://fbref.com/en/comps/41/schedule/
 ↪Primera-A-Scores-and-Fixtures'

Japan_table = 'https://fbref.com/en/comps/25/
 ↪J1-League-Stats#results100791_home_away::none'
Japan_fixture = 'https://fbref.com/en/comps/25/schedule/
 ↪J1-League-Scores-and-Fixtures'

Netherlands_table = "https://fbref.com/en/comps/23/
 ↪Dutch-Eredivisie-Stats#results107391_home_away::none"
Netherlands_fixture = "https://fbref.com/en/comps/23/schedule/
 ↪Dutch-Eredivisie-Scores-and-Fixtures"
```

1.1.2 Configuration

In this section of the code, the script is preparing to analyze a specific football league, indicated as the Premier League, by setting the variables `Table`, `Fixture`, and `Week` to values that correspond to the Premier League.

The variable `Table` represents the statistical data table sourced from the Premier League, `Fixture` points to the fixture data for the same league, and `Week` is set to a specific week within the season for the subsequent analysis. By customizing these variables, the code is set to focus on the statistical analysis and predictions for the specified league and week.

[3]:
```
# Configuration for League Analysis

# Mapping country names to their respective URLs
country_urls = {
    "Sweden": [Sweden_table, Sweden_fixture],
    "Norway": [Norway_table, Norway_fixture],
    "Finland": [Finland_table, Finland_fixture],
    "Mexico": [Mexico_table, Mexico_fixture],
    "Turkey": [Turkey_table, Turkey_fixture],
    "England": [Premier_table, Premier_fixture],
    "France": [France_table, France_fixture],
    "Germany": [Germany_table, Germany_fixture],
    "Italy": [Italy_table, Italy_fixture],
    "Spain": [Spain_table, Spain_fixture],
    "Brazil": [Brazil_table, Brazil_fixture],
    "Columbia": [Columbia_table, Columbia_fixture],
    "Japan": [Japan_table, Japan_fixture],
    "Netherlands": [Netherlands_table, Netherlands_fixture]
}

# Input the desired country for analysis
selected_country = input("Enter the country name for the analysis: ")

# Automatically select the corresponding URLs based on the input country
if selected_country in country_urls:
    Table, Fixture = country_urls[selected_country]
    print(f"Selected {selected_country} league and fixture for analysis.")
else:
    print("Country not found. Please input a valid country name.")

    # Input the specific week for analysis
selected_week = int(input("Enter the week for analysis: "))

# Selected week for analysis
Week = selected_week
print(f"Selected {selected_week} week for analysis.")
```

```
Enter the country name for the analysis: England
Selected England league and fixture for analysis.
Enter the week for analysis: 11
Selected 11 week for analysis.
```

1.1.3 Data Processing and Metric Computation

Metric Calculation This segment of code is instrumental in processing the data extracted from the selected statistical table, enabling the computation of essential performance metrics for both home and away teams in the context of the football league being analyzed. By calculating key parameters such as average goals scored and conceded, along with league-wide average goals, the code establishes a comprehensive framework for evaluating the attacking and defensive strengths of each team. *** These calculated metrics, including **Home Attacking Strength**, **Home Defensive Strength**, **Away Attacking Strength**, and **Away Defensive Strength**, play a pivotal role in assessing the relative capabilities of teams within the league, forming the basis for subsequent match outcome predictions and in-depth performance analyses.

[4]:
```python
# Extracting data from the selected statistical table
df_site = pd.read_html(Table)
df = df_site[1]

# Calculating various metrics based on the extracted data

# HAGF = Home Average Goals For
df['HAGF'] = df['Home']['GF'] / df['Home']['MP']
# HAGA = Home Average Goals Against
df['HAGA'] = df['Home']['GA'] / df['Home']['MP']
# AAGF = Away Average Goals For
df['AAGF'] = df['Away']['GF'] / df['Away']['MP']
# AAGA = Away Average Goals Against
df['AAGA'] = df['Away']['GA'] / df['Away']['MP']

index = df.index

# LAGFH = League Average Goals For the Home teams
LAGFH = df['Home']['GF'].sum() / len(index)
# LAGFA = League Average Goals For the Away teams
LAGFA = df['Away']['GF'].sum() / len(index)
# LAGFEH = League Average Goals For Each game for Home teams
df['LAGFEH'] = LAGFH / df['Home']['MP']
# LAGFEA = League Average Goals For Each game for Away teams
df['LAGFEA'] = LAGFA / df['Away']['MP']

# Computing various team strength metrics

# HomeAttackingStrength
df['HomeAttackingStrength'] = df['HAGF'] / df['LAGFEH']
# HomeDefensiveStrength
df['HomeDefensiveStrength'] = df['HAGA'] / df['LAGFEA']
# AwayAttackingStrenth
df['AwayAttackingStrength'] = df['AAGF'] / df['LAGFEA']
# AwayDefensiveStrength
df['AwayDefensiveStrength'] = df['AAGA'] / df['LAGFEH']
```

Feature Extraction The subsequent batch of code facilitates the processing of fixture data extracted from the designated source, enabling the creation of comprehensive data frames for both home and away teams. By leveraging the calculated metrics of Home Attacking Strength, Home Defensive Strength, Away Attacking Strength, and Away Defensive Strength, the code constructs an amalgamated data frame that consolidates these vital parameters. The amalgamated data frame forms the basis for computing the goal expectancies for the home and away teams, incorporating intricate calculations based on the **interplay between team strengths and league-wide average goals**. This calculated data provides critical insights into the anticipated **goal-scoring potential**

of each team, crucial for informing the subsequent match outcome predictions and facilitating a deeper understanding of the dynamics at play within the football league.

```
[5]: # Extracting fixture data from the specified source
fixtures_site = pd.read_html(Fixture)
fixtures = fixtures_site[0]
fixtures21 = fixtures[fixtures['Wk'] == Week]

# Creating data frames for home teams
last_df = pd.DataFrame()
for hometeam in fixtures21['Home']:
    # Extracting relevant metrics for the home team
    HTAS = df[df['Unnamed: 1_level_0']['Squad'] ==␣
 ↪hometeam]['HomeAttackingStrength'].values[0]
    HTDS = df[df['Unnamed: 1_level_0']['Squad'] ==␣
 ↪hometeam]['HomeDefensiveStrength'].values[0]
    LAGFEH = df[df['Unnamed: 1_level_0']['Squad'] == hometeam]['LAGFEH'].
 ↪values[0]
    # Creating a dictionary and initializing a data frame for the home team
    home_dict = {'Home':[hometeam],
                 'Home Team Attacking Strength':[HTAS],
                 'Home Team Defensive Strength':[HTDS],
                 'LAGFEH':[LAGFEH]}
    initial_df = pd.DataFrame.from_dict(home_dict)
    last_df = pd.concat([last_df, initial_df], ignore_index=True)

# Creating data frames for away teams
last2_df = pd.DataFrame()
for awayteam in fixtures21['Away']:
    # Extracting relevant metrics for the away team
    ATAS = df[df['Unnamed: 1_level_0']['Squad'] ==␣
 ↪awayteam]['AwayAttackingStrength'].values[0]
    ATDS = df[df['Unnamed: 1_level_0']['Squad'] ==␣
 ↪awayteam]['AwayDefensiveStrength'].values[0]
    LAGFEA = df[df['Unnamed: 1_level_0']['Squad'] == awayteam]['LAGFEA'].
 ↪values[0]
    # Creating a dictionary and initializing a data frame for the away team
    away_dict = {'Away':[awayteam],
                 'Away Team Attacking Strength':[ATAS],
                 'Away Team Defensive Strength':[ATDS],
                 'LAGFEA':[LAGFEA]}
    initial2_df = pd.DataFrame.from_dict(away_dict)
    last2_df = pd.concat([last2_df, initial2_df], ignore_index=True)

# Concatenating the data frames for home and away teams
df_concat = pd.concat([last_df, last2_df], axis=1)
df_concat.reset_index(drop=True, inplace=True)
```

```
# Computing our special goal expectancy metric for home and away teams
df_concat['Home Team Goal Expectancy'] = df_concat['Home Team Attacking␣
 ↪Strength'] * df_concat['Away Team Defensive Strength'] * df_concat['LAGFEH']
df_concat['Away Team Goal Expectancy'] = df_concat['Away Team Attacking␣
 ↪Strength'] * df_concat['Home Team Defensive Strength'] * df_concat['LAGFEA']
```

[6]:
```
df_concat
```

[6]:
```
             Home  Home Team Attacking Strength  \
0          Fulham                      0.487805
1       Brentford                      1.097561
2         Everton                      0.487805
3         Burnley                      0.487805
4 Manchester City                      1.219512
5   Sheffield Utd                      0.487805
6   Newcastle Utd                      1.585366
7 Nott'ham Forest                      0.731707
8      Luton Town                      0.365854
9       Tottenham                      0.975610

   Home Team Defensive Strength  LAGFEH            Away  \
0                      1.294964    2.05  Manchester Utd
1                      1.151079    1.64        West Ham
2                      0.719424    1.64        Brighton
3                      2.302158    1.64  Crystal Palace
4                      0.431655    2.05     Bournemouth
5                      2.158273    1.64          Wolves
6                      0.431655    1.64         Arsenal
7                      1.079137    2.05     Aston Villa
8                      1.294964    2.05       Liverpool
9                      0.431655    2.05         Chelsea

   Away Team Attacking Strength  Away Team Defensive Strength  LAGFEA  \
0                      0.575540                      1.097561  1.7375
1                      1.151079                      1.219512  1.3900
2                      1.294964                      1.829268  1.7375
3                      0.575540                      0.975610  1.3900
4                      0.575540                      2.012195  1.7375
5                      0.863309                      0.731707  1.3900
6                      1.151079                      0.365854  1.7375
7                      0.863309                      1.219512  1.3900
8                      1.294964                      0.853659  1.3900
9                      1.007194                      0.731707  1.7375

   Home Team Goal Expectancy  Away Team Goal Expectancy
0                   1.097561                   1.294964
1                   2.195122                   1.841727
```

```
2                  1.463415                1.618705
3                  0.780488                1.841727
4                  5.030488                0.431655
5                  0.585366                2.589928
6                  0.951220                0.863309
7                  1.829268                1.294964
8                  0.640244                2.330935
9                  1.463415                0.755396
```

1.1.4 Match Predictions Analysis

> Now it's time to unravel the mystical predictions imbued within the essence of the code. Witness the spellbinding insights conjured from the depths of statistical sorcery and the otherworldly powers of the **Poisson distribution**.

This section details the comprehensive analysis of match predictions based on calculated goal expectancies. The code iterates through each match specified in the dataset and computes various prediction outcomes. First, it constructs a probability table based on Poisson distribution, representing the probabilities of different goal combinations for each team. Then, it calculates the likelihood of over, under, and exact goal predictions, as well as the likelihood of win, draw, or loss, and the likelihood of both teams scoring (BTS). The results are stored in the `predictions_df` DataFrame, providing insights into potential match outcomes based on statistical computations and goal expectancies.

[7]:
```python
predictions_df = pd.DataFrame()
# Iterating through each match for predictions analysis
for match in df_concat['Home']:
    # Calculating various prediction outcomes based on goal expectancies
    Htge = df_concat[df_concat['Home'] == match]['Home Team Goal Expectancy'].
 ↪values[0]
    Atge = df_concat[df_concat['Home'] == match]['Away Team Goal Expectancy'].
 ↪values[0]
    away = df_concat[df_concat['Home'] == match]['Away'].values[0]
    prob_a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    prob_b = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

    prod_table = np.array([(i*j) for i, j in product(poisson.pmf(prob_b, Htge),␣
 ↪poisson.pmf(prob_a, Atge))])
    prod_table.shape = (11, 11)

    prob_df = pd.DataFrame(prod_table, index=prob_b, columns=prob_a)
    prob_df = prob_df.select_dtypes(exclude=['object', 'datetime']) * 100

    # Predictions
    list_over_prediction = []
    for a in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:
        home_goals = prob_df[prob_df[a] >= 3].index.values
        for i in home_goals:
            score = str(i)+"-"+str(a)
            total_goals = i + a
            if total_goals == 0:
                list_over_prediction.append(0)
            elif total_goals >= 0.5 and total_goals <= 1.5:
                list_over_prediction.append(0.5)
            elif total_goals >= 1.5 and total_goals <= 2.5:
                list_over_prediction.append(1.5)
            elif total_goals >= 2.5 and total_goals <=3.5:
                list_over_prediction.append(2.5)
            elif total_goals >= 3.5 and total_goals <= 4.5:
                list_over_prediction.append(3.5)
            else:
                list_over_prediction.append(4.5)
    Over_prediction = "Over " + str(min(list_over_prediction))

    list_under_prediction = []
    for a in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:
        home_goals = prob_df[prob_df[a] >= 3].index.values
        for i in home_goals:
            score = str(i)+"-"+str(a)
            total_goals = i + a
            if total_goals == 0:
                list_under_prediction.append(0.5)
            elif total_goals >= 0.5 and total_goals <= 1.5:
                list_under_prediction.append(1.5)
            elif total_goals >= 1.5 and total_goals <= 2.5:
                list_under_prediction.append(2.5)
            elif total_goals >= 2.5 and total_goals <=3.5:
                list_under_prediction.append(3.5)
            elif total_goals >= 3.5 and total_goals <= 4.5:
                list_under_prediction.append(4.5)
            else:
                list_under_prediction.append(5.5)
    Under_prediction = "Under " + str(max(list_under_prediction))

    list_win_prediction = []
    for a in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:
        home_goals = prob_df[prob_df[a] >= 3].index.values
        for i in home_goals:
            score = str(i)+"-"+str(a)
            if i > a:
                list_win_prediction.append(1)
            if i == a:
                list_win_prediction.append(0)
            if i < a:
                list_win_prediction.append(2)

    if 2 and 0 not in list_win_prediction:
        Win_prediction = "Home wins"
    elif 2 not in list_win_prediction:
        Win_prediction = "Home does not lose"
    elif 1 and 0 not in list_win_prediction:
        Win_prediction = "Away wins"
    elif 1 not in list_win_prediction:
        Win_prediction = "Away does not lose"
    elif 1 and 2 not in list_win_prediction:
        Win_prediction = "Draw"
    else:
        Win_prediction = "No result"

    list_BTS_prediction = []
    for a in [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:
        home_goals = prob_df[prob_df[a] >= 3].index.values
        for i in home_goals:
            score = str(i)+"-"+str(a)
            if i and a > 0:
                list_BTS_prediction.append(1)
            else:
                list_BTS_prediction.append(0)

    if 0 not in list_BTS_prediction:
        BTS_prediction = "BTS"
    else:
        BTS_prediction = "No result"

    dict_predictions = {"Home":[match],
                         "Away":[away],
                         "Over Prediction":[Over_prediction],
                        "Under Prediction":[Under_prediction],
                        "Result Prediction":[Win_prediction],
                        "BTS Prediction":[BTS_prediction]}
    inner_dataframe = pd.DataFrame.from_dict(dict_predictions)

    predictions_df = pd.concat([predictions_df, inner_dataframe],␣
 ↪ignore_index=True)
predictions_df.reset_index(drop=True, inplace=True)
```

1.2 Results

The printed `predictions_df` DataFrame provides a comprehensive set of match predictions for the given fixtures. The table includes columns such as ”Home” and ”Away,” listing the participating teams for each match. The ”Over Prediction” and ”Under Prediction” columns provide anticipated goal totals for the corresponding match, with the ”Result Prediction” column delivering a forecast of the likely match outcome, including potential scenarios such as a home win, away win, or a draw. The ”BTS Prediction” column offers insights into whether both teams are expected to score. These predictions are based on intricate calculations leveraging Poisson statistics and the teams' respective attacking and defensive strengths. The DataFrame serves as a valuable reference for forecasting match results and goal expectations in the context of the analyzed football league.

[8]:
```
predictions_df
```

[8]:
```
              Home            Away Over Prediction Under Prediction  \
0           Fulham  Manchester Utd          Over 0        Under 4.5
1        Brentford        West Ham        Over 0.5        Under 5.5
2          Everton        Brighton          Over 0        Under 5.5
3          Burnley  Crystal Palace          Over 0        Under 4.5
4  Manchester City     Bournemouth        Over 1.5        Under 5.5
5    Sheffield Utd          Wolves          Over 0        Under 5.5
6    Newcastle Utd         Arsenal          Over 0        Under 3.5
7  Nott'ham Forest     Aston Villa          Over 0        Under 5.5
8       Luton Town       Liverpool          Over 0        Under 5.5
9        Tottenham         Chelsea          Over 0        Under 4.5

  Result Prediction BTS Prediction
0         No result      No result
1         No result      No result
2         No result      No result
3         No result      No result
4         Home wins      No result
5  Away does not lose    No result
6         No result      No result
7         No result      No result
8         No result      No result
9         No result      No result
```

1.3 Conclusion

In this project, we embarked on a mystical journey into the realm of football analytics, leveraging data extracted from the esteemed website fbref.com. We began by extracting statistical tables and fixture data for various renowned football leagues across the globe, focusing specifically on the English Premier League. By diving into the code, we comprehensively computed a range of crucial metrics, including team attacking and defensive strengths, league average goals, and team goal expectancies. Employing advanced statistical methodologies such as the Poisson distribution, we conjured forth a myriad of intriguing predictions, including over/under predictions, match results, and both teams to score (BTS) forecasts. > Through this magical journey, we unearthed valuable insights that illuminated the mystical depths of football analytics, unraveling the enigmatic tapestry of the beautiful game.

> However, the unpredictability of football results is the essence of the sport, defying any attempts to fully forecast the outcomes, adding to the allure and magic of the game.