



Spring 2020 INFO6205 Project Report

Topic: Predictive Ranking System of English Premiere League

(Underlining concepts utilized: Statistics, Probability Distribution Functions and Data Structures)

Instructor: Prof. Robin Hillyard

Submitted as partial fulfilment of the semester project by:

Deepak Lokwani (001316769)

Rupesh Alasundkar (001304532)

Teaching Assistants:

Aditi Jalkote

Ajay Goel

1. ABSTRACT INTRODUCTION

In the light of the development around the SARS-Covid-19 disease, the rest of the season of the English Premiere League has been postponed. And the chances seem very bleak of the league resuming anytime soon.

In such a scenario, a league that has been played proudly and followed by billions of fans for the last 28 years needs a predictive ranking system which can predict with the utmost accuracy what the final standing would have been if the tournament had been played as usual!



2. PROBLEM STATEMENT

The format of English Premiere League consists of 20 teams. In this league format of tournament, each team faces the remaining 19 teams, twice. Once at home ground and other at the opponent's home ground, i.e., away. So, in total there are be total 380 matches.

This season 288 out this 380 are already played and 92 are pending. Each team is awarded +3 points on winning and +1 is awarded to both the teams on a draw result.

Before jumping to any assumptions, we need to set our priorities and find out the deciding factors in any match played to be able to predict the result.

Few pivotal factors that could help us design the algorithm can be listed as follows:

- a. A team is mostly (well always) strong when playing at home.**
- b. Some teams are better at scoring goals**
- c. Some teams are better at defending the goalpost**
- d. A team is found to perform better if it is a do-or-die match for that team and not so crucial match for the opponent team**
- e. Performance of the two teams have been observed to be improved if they are arch-rivals**
- f. Club circumstances or any other injuries/accidents**

The latter three factors are pretty intangible and could require the use of a large data. So, we will focus only on the first three points, i.e., the location of the match i.e., home advantage, goal scoring ability, i.e., attack strength and the defense strength of a team, i.e., defending the goalpost

3. DESCRIPTION OF DATA AND CURRENT RANKING

The data that I used in this project are collected from <http://www.football-data.co.uk/englandm.php>. I collected the detailed match data of 2019-20 premier league season, which can be regard as my training set.





















The original data is a data frame with 289 rows and 49 columns, in which each row stands for full information of one game in 2019-20 season. The information contains team names, managers of teams, general information (date, location, name of referee, etc.) and detailed statistics (e.g. number of assists, number of goals, number of tackles, etc.)

However, the data is too messy for the further research—as I mentioned in part I, what I need is only the number of goals of home and away teams, so I simplify the original data and produced the read data that only contains 289 rows and 5 columns., Mainly, Home Team, Away Team, Full time Goals of home, Full time Goals of away team, Result.

This data is read through my reader file name *RankingSystems.java*

This file calculates all the required data values that would be required further in the *driver.java* class for the calculation of the PDF of a match result.

Looking at the current point table, it is not very difficult to predict that Liverpool must end the season at the top of the table (it is mathematically impossible for any other team to pass them), the next five positions are important for the summer, as are the last three positions, which teams will be relegated.

P	Team	GP	W	D	L	F	A	GD	Pts
1	 Liverpool	29	27	1	1	66	21	45	82
2	 Man City	28	18	3	7	68	31	37	57
3	 Leicester	29	16	5	8	58	28	30	53
4	 Chelsea	29	14	6	9	51	39	12	48
5	 Man Utd	29	12	9	8	44	30	14	45
6	 Wolves	29	10	13	6	41	34	7	43
7	 Sheff Utd	28	11	10	7	30	25	5	43
8	 Spurs	29	11	8	10	47	40	7	41
9	 Arsenal	28	9	13	6	40	36	4	40
10	 Burnley	29	11	6	12	34	40	-6	39
11	 C Palace	29	10	9	10	26	32	-6	39
12	 Everton	29	10	7	12	37	46	-9	37
13	 Newcastle	29	9	8	12	25	41	-16	35
14	 Southampton	29	10	4	15	35	52	-17	34
15	 Brighton	29	6	11	12	32	40	-8	29
16	 West Ham	29	7	6	16	35	50	-15	27
17	 Watford	29	6	9	14	27	44	-17	27
18	 AFC Bournemouth	29	7	6	16	29	47	-18	27
19	 Aston Villa	28	7	4	17	34	56	-22	25
20	 Norwich	29	5	6	18	25	52	-27	21

4. ACQUAINTANCE WITH BASIC CONCEPTS

a. What is Attack Strength and how to calculate it?

The first step in calculating Attack Strength based upon last season's results is to determine the average number of goals scored per team, per home game, and per away game.

Season total goals scored at home / number of games (in season)

Season total goals scored away / number of games (in season)

Take for e.g.,

In 2015/16 English Premier League season, there were 567/380 at home and 459/380 away, equaling an average of 1.492 goals per game at home and 1.207 away.

Average number of goals scored at home: 1.492

Average number of goals scored away: 1.207

The ratio of a team's average and the league average is what constitutes "Attack Strength".

b. What is Defense Strength and how to calculate it?

This is simply the inverse of the above numbers (as the number of goals a home team scores will equal the same number that an away team concedes):

Average number of goals conceded at home: 1.207

Average number of goals conceded away from home: 1.492

The ratio of a team's average and the league average is what constitutes "Defense Strength".

c. How to predict scores based on it?

Take for e.g.,

Match: Tottenham Hotspur and Everton (as of 1st March 2017).

Calculate Tottenham's Attack Strength:

Step - 1: Take the number of goals scored at home last season by the home team (Tottenham: 35) and divide by the number of home games (35/19): 1.842.

Step - 2: Divide this value by the season's average home goals scored per game (1.842/1.492) to get an ***"Attack Strength" of 1.235.***

Calculate Everton's Defense Strength:

Step - 1: Take the number of goals conceded away from home last season by the away team (Everton: 25) and divide by the number of away games (25/19): 1.315.

Step - 2: Divide this by the season's average goals conceded by an away team per game (1.315/1.492) to get a ***"Defense Strength" of 0.881.***

Predicting Tottenham's Goals:

We can now use the following formula to calculate the likely number of goals Tottenham might score (this is done by multiplying Tottenham's Attack Strength by Everton's Defense Strength and the average number of home goals in the Premier League):

$$1.235 \times 0.881 \times 1.492 = 1.623 \quad \text{-(a)}$$

Predicting Everton's Goals:

To calculate the number of goals Everton might score, simply use the above formulas **but replace the average number of home goals with the average number of away goals.**

Everton's Attack Strength:

$$(24/19) / (459/380) = 1.046$$

Tottenham's Defence Strength:

$$(15/19) / (459/380) = 0.653$$

In the same way we predicted the number of goals Tottenham will score, we can calculate the likely number of goals Everton might ***score (done by multiplying Everton's Attack Strength by Tottenham's Defence Strength and the average number of away goals in the Premier League):***

$$1.046 \times 0.653 \times 1.207 = \mathbf{0.824} \quad \text{-(b)}$$

d. What is Poisson distribution and how is it applicable here?

Poisson Distribution is a mathematical concept for translating mean averages into a probability for variable outcomes across a distribution.

Of course, no game ends 1.623 vs. 0.824 – this is ***simply the average***. Poisson Distribution, a formula created by French mathematician Simeon Denis Poisson, allows us to use these figures to distribute 100% of probability across a range of goal outcomes for each side.

For example, if we know Manchester City average 1.7 goals per game, so by putting the Poisson Distribution formula tells us that this average equates to Manchester City scoring 0 goals 18.3% of the time, 1 goal 31% of the time, 2 goals 26.4% of the time and 3 goals 15% of the time.

To calculate the most likely score-line using Poisson Distribution:

Poisson Distribution formula:

$$P(x; \mu) = (e^{-\mu}) (\mu^x) / x!$$

e = Euler's constant = 2.718

$!$ = Factorial

x = number of successes of the event

μ = mean distribution of the event

All we need to do is enter the different event occurrences - in our case goals outcomes from 0-5 - and the expected occurrences which are the likelihood of each team scoring - in our example Tottenham at 1.623 is their average rate of success, and Everton 0.824; the calculator will output the probability of the score for the given outcome.

Tottenham vs. Everton Poisson Distribution



Goals	0	1	2	3	4	5
Tottenham	19.73%	32.02%	25.99%	14.06%	5.07%	1.85%
Everton	43.86%	36.14%	14.89%	4.09%	0.84%	0.14%

This example shows that there is a 19.73% chance that Tottenham will fail to score, but a 32.02% chance they will score a single goal and a 25.99% chance they'll score two. Everton, on the other hand, is at 43.86% not to score, 36.14% to score one and 14.89% to score two. Hoping for a side to score five? The probability is 1.85% for Tottenham or 0.14% for Everton - or 2% for either team to score 5.

As both scores are independent (mathematically-speaking), you can see that the expected score is 1-0 - pairing the most probable outcomes for each team. If you multiply those two probabilities together, you'll get the probability of the 1-0 outcome - $(0.3202 * 0.4386) = 0.1404$ or 14.04%.

5. PROPOSAL

The project intends to get data from the matches that have already been played. The data that we intend to consider are the average number of goals a team has been scoring at the home and away and the average number of goals a team has been conceding at home and away.

Based on the data of the season played until now the algorithm design computes the probability of the final score and will also check remaining 98 matches.

In summary factors considered are:

- a. Location (Home/Away)
- b. Attack Strength
- c. Defense Strength

Based on the above factors, a face-off between two teams can be evaluated in two ways by considering:

- a. Cumulative Probability of a team winning***
- b. Discrete Probability of a goal score***

Difference between both the methods is marked out in the observations section.

And we are using both the methods in our algorithm. It is left on the user to choose a method of their choice.

We are using *Poisson Distribution* to determine how goals can be scored by the team and probabilities of scoring goals.

The ***design is run over the matches that are yet to played and a point table is created to rank the teams accordingly.***

Later, a ***user-interface is further extended where in a user inputs the home team and the away team and the results of that face off in the form of a probability distribution function is represented.***

6. DESIGN IMPLEMENTATION

My Training Data is stored in the directory in a .csv file format. This data is read through my reader file name ***RankingSystems.java***

This file calculates the home team and the away team parameters like, matches played, the goals scored/conceded and their average. It also calculates the total league parameters for the normalization of the data required further in the calculation. The ***Data Structures used here is an array*** of floats. ***An array is used for the ease of access and faster manipulation as well as fetching.***

There's one more class called ***MakePairs.java*** file.

This class makes the pairs of teams to face a match between them. And each match to be played or already played are segregated is given a combination ID and is stored in a ***HashMap***. All the pairs are also grouped in played or not played ***HashMap***. ***A HashMap is used here as the team pairs should not be repeated and we need unique keys for each combination of teams.***

It is worth noting that a match between teams A & B are not same as Team B & A, as we consider the team precedence order to determine the home/away location. The first string/team is considered the home team and the latter one the away team.

Our main class, the ***Driver.java*** file, initiates all the pre-requisite functions that are

- Required to tabulate the final point table.
- The driver constructor initially runs and makes assigns the combination id to each of the match that is played or is supposed to played.
- This class contains the algorithm design for predicting the ranking of all the clubs playing in the EPL based on the previous records of the current season.

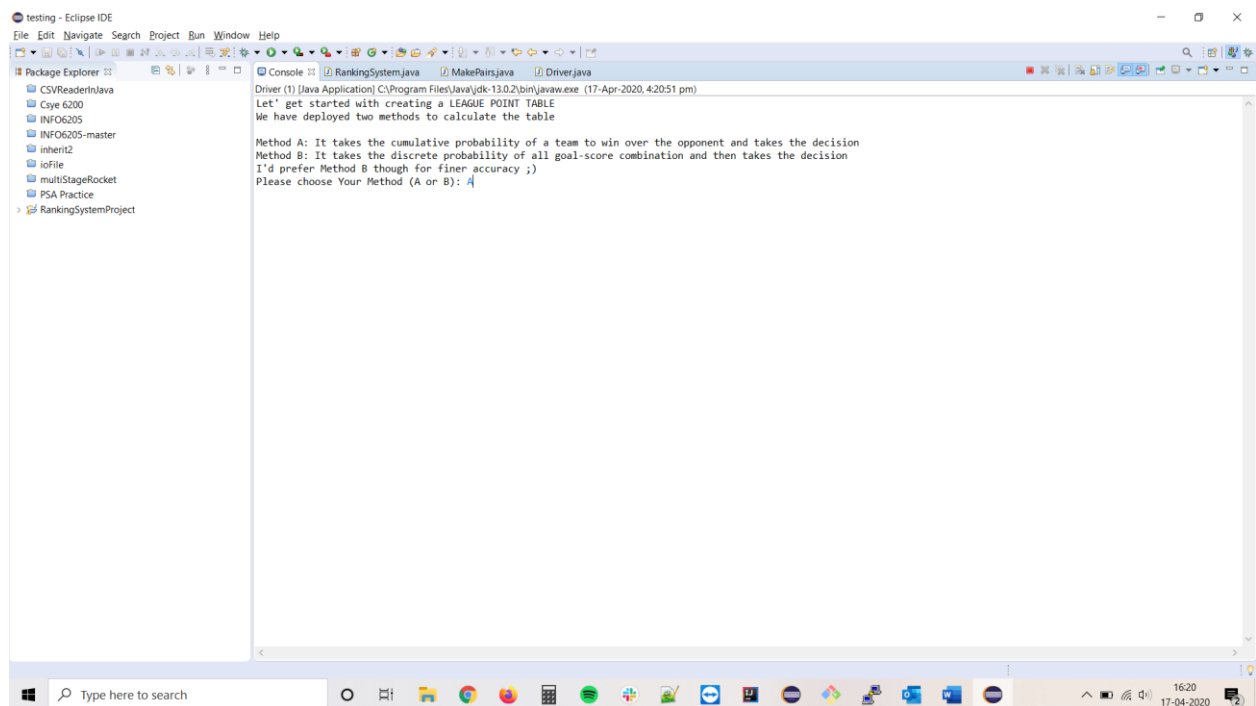
- The driver class considers the user input for the method selection of for the point tables (A or B)
- Then the class takes the user input for the Team selection (home and away) and computes the PDF between them and represents it in a matrix form. The cumulative as well as discrete probability methods are run both the results are shown to the user.

7. OBSERVATIONS AND TESTING

PART I: LEAGUE POINT TABLE

Firstly, when the main class (Driver.java) is run, a console output is thrown asking user to choose the method to rank the teams and create a point table.

A user can input either method A or B. Below are the screenshot of the following steps:



This screenshot shows us the results based on the method chosen A

```

Driver (1) [Java Application] C:\Program Files\Java\jdk-13.0.2\bin\javaw.exe (17-Apr-2020, 4:20:51 pm)
Let's get started with creating a LEAGUE POINT TABLE
We have deployed two methods to calculate the table

Method A: It takes the cumulative probability of a team to win over the opponent and takes the decision
Method B: It takes the discrete probability of all goal-score combination and then takes the decision
I'd prefer Method B though for finer accuracy ;)
Please choose Your Method (A or B): A
Please wait while we compute the point table
The updated point table with selected method is as below

Rank      Club Name      FinalPoints
1          Liverpool      109
2          Man City        84
3          Leicester       80
4          Wolves          70
5          Man United      66
6          Chelsea        66
7          Tottenham       62
8          Sheffield United 61
9          Arsenal         55
10         Burnley         51
11         Southampton     46
12         Crystal Palace  45
13         Everton         43
14         West Ham       39
15         Brighton       35
16         Newcastle      35
17         Watford        33
18         Norwich        30
19         Bournemouth    30
20         Aston Villa    28

{Liverpool=109, Man City=84, Leicester=80, Wolves=70, Man United=66, Chelsea=66, Tottenham=62, Sheffield United=61, Arsenal=55, Burnley=51, Southampton=46, Crystal Pal=
Please Enter Club Names to find one-on-one Probability Distribution Function
Team names are case-sensitive
Choose Home Team:
  
```

This screenshot shows us the results based on the method chosen B

```

Driver (1) [Java Application] C:\Program Files\Java\jdk-13.0.2\bin\javaw.exe (17-Apr-2020, 4:21:40 pm)
Let's get started with creating a LEAGUE POINT TABLE
We have deployed two methods to calculate the table

Method A: It takes the cumulative probability of a team to win over the opponent and takes the decision
Method B: It takes the discrete probability of all goal-score combination and then takes the decision
I'd prefer Method B though for finer accuracy ;)
Please choose Your Method (A or B): B
Please wait while we compute the point table
The updated point table with selected method is as below

Rank      Club Name      FinalPoints
1          Liverpool      107
2          Man City        85
3          Leicester       80
4          Wolves          62
5          Chelsea        62
6          Sheffield United 61
7          Man United      59
8          Tottenham       57
9          Arsenal         55
10         Burnley         49
11         Southampton     44
12         Crystal Palace  41
13         Everton         41
14         West Ham       39
15         Newcastle      39
16         Brighton       34
17         Bournemouth    32
18         Watford        32
19         Aston Villa    30
20         Norwich        26

{Liverpool=107, Man City=85, Leicester=80, Wolves=62, Chelsea=62, Sheffield United=61, Man United=59, Tottenham=57, Arsenal=55, Burnley=49, Southampton=44, Crystal Pal=
Please Enter Club Names to find one-on-one Probability Distribution Function
Team names are case-sensitive
Choose Home Team:
  
```

There are minor changes in the final ranking table with different methods as can be seen.

PART II: Face-off Probability Distribution Function

Here the console asks the user to input two teams, home and away team. The output shows the following results in order:

- Goals prediction of home team***
- Goals prediction of away team***
- Probability distribution function over different combinations of the final score***
- Final score with the maximum probability(discrete)***
- Cumulative probability of a team winning, losing or match drawing based on the above discrete probability functions***

```

testing - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Package Explorer
  CSVReaderInJava
  Cye 6200
  INFO6205
  INFO6205-master
  inheri2
  ioFile
  multiStageRocket
  PSA Practice
  > RankingSystemProject
  Console
    <terminated> Driver (1) [Java Application] C:\Program Files\Java\jdk-13.0.2\bin\javaw.exe (17-Apr-2020, 4:24:11 pm - 4:24:36 pm)
    4 Wolves 62
    5 Chelsea 62
    6 Sheffield United 61
    7 Man United 59
    8 Tottenham 57
    9 Arsenal 55
    10 Burnley 49
    11 Southampton 44
    12 Crystal Palace 41
    13 Everton 41
    14 West Ham 39
    15 Newcastle 39
    16 Brighton 34
    17 Bournemouth 32
    18 Watford 32
    19 Aston Villa 30
    20 Norwich 26

    {Liverpool=107, Man City=85, Leicester=80, Wolves=62, Chelsea=62, Sheffield United=61, Man United=59, Tottenham=57, Arsenal=55, Burnley=49, Southampton=44, Crystal Palace=41, Everton=41, West Ham=39, Newcastle=39, Brighton=34, Bournemouth=32, Watford=32, Aston Villa=30, Norwich=26}

    Please Enter Club Names to find one-on-one Probability Distribution Function
    Team names are case-sensitive
    Choose Home Team: Liverpool
    Choose Away Team: Arsenal

    Home Team Liverpool's final score probability: 2.18 goals
    Away Team Arsenal's final score probability: 0.57 goals

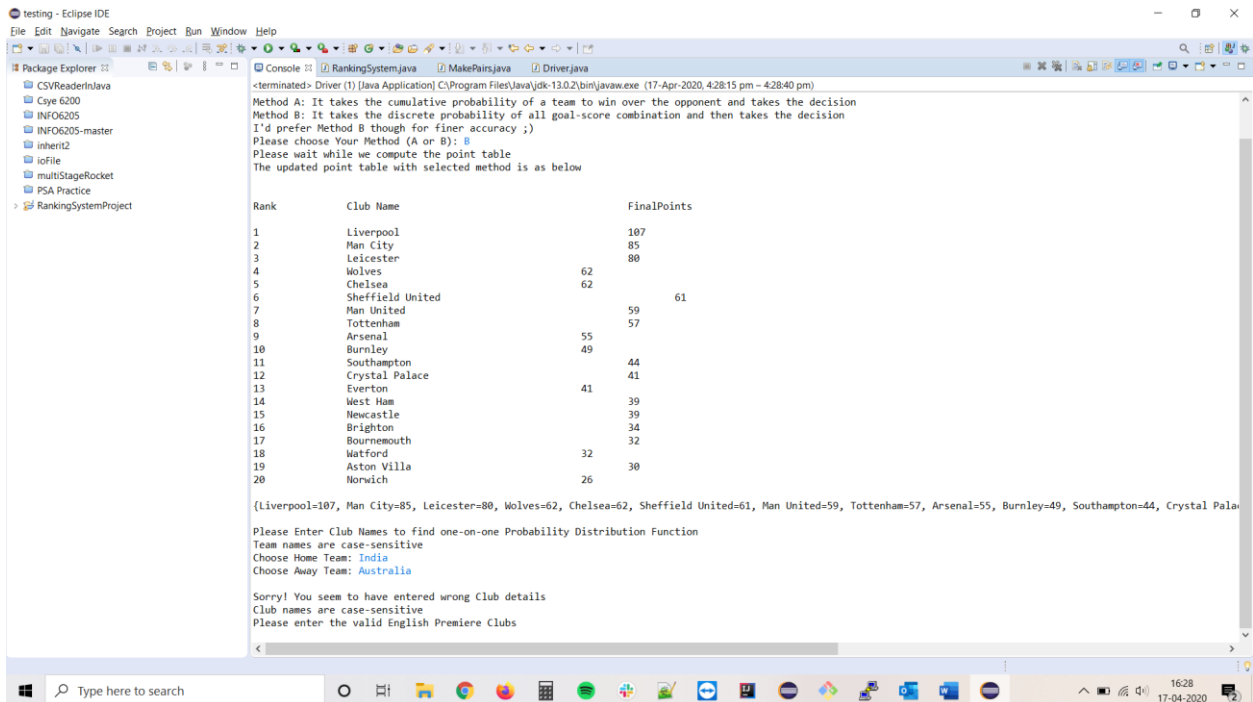
    Liverpool's and Arsenal's different score combinations and their corresponding probabilities are as follows:
    0 : 0 = 6.41 % 0 : 1 = 3.65 % 0 : 2 = 1.04 % 0 : 3 = 0.2 % 0 : 4 = 0.03 % 0 : 5 = 0.0 %
    1 : 0 = 13.96 % 1 : 1 = 7.95 % 1 : 2 = 2.26 % 1 : 3 = 0.43 % 1 : 4 = 0.06 % 1 : 5 = 0.01 %
    2 : 0 = 15.2 % 2 : 1 = 8.66 % 2 : 2 = 2.47 % 2 : 3 = 0.47 % 2 : 4 = 0.07 % 2 : 5 = 0.01 %
    3 : 0 = 11.03 % 3 : 1 = 6.29 % 3 : 2 = 1.79 % 3 : 3 = 0.34 % 3 : 4 = 0.05 % 3 : 5 = 0.01 %
    4 : 0 = 6.01 % 4 : 1 = 3.42 % 4 : 2 = 0.97 % 4 : 3 = 0.19 % 4 : 4 = 0.03 % 4 : 5 = 0.0 %
    5 : 0 = 2.62 % 5 : 1 = 1.49 % 5 : 2 = 0.42 % 5 : 3 = 0.08 % 5 : 4 = 0.01 % 5 : 5 = 0.0 %

    We predict that the final score for Liverpool : Arsenal 2 : 0 has the maximum probability around 15.2 %

    Chances of Liverpool winning are 72.14 %
    Chances of a Draw are 17.19 %
    Chances of Arsenal winning are 8.28 %
  
```

Note: The percentage probability almost always adds up to 100% giving us a hint of the correctness of the calculations

Exception Handling



```
<terminated> Driver (1) [Java Application] C:\Program Files\Java\jdk-13.0.2\bin\javaw.exe (17-Apr-2020, 4:28:15 pm - 4:28:40 pm)
Method A: It takes the cumulative probability of a team to win over the opponent and takes the decision
Method B: It takes the discrete probability of all goal-score combination and then takes the decision
I'd prefer Method B though for finer accuracy ;)
Please choose Your Method (A or B): B
Please wait while we compute the point table
The updated point table with selected method is as below

Rank      Club Name      FinalPoints
1          Liverpool      107
2          Man City       85
3          Leicester      80
4          Wolves         62
5          Chelsea        62
6          Sheffield United 61
7          Man United     59
8          Tottenham      57
9          Arsenal        55
10         Burnley       49
11         Southampton   44
12         Crystal Palace 41
13         Everton       41
14         West Ham      39
15         Newcastle     39
16         Brighton      34
17         Bournemouth   32
18         Watford       32
19         Aston Villa   30
20         Norwich       26

{Liverpool=107, Man City=85, Leicester=80, Wolves=62, Chelsea=62, Sheffield United=61, Man United=59, Tottenham=57, Arsenal=55, Burnley=49, Southampton=44, Crystal Palace=41, Everton=41, West Ham=39, Newcastle=39, Brighton=34, Bournemouth=32, Watford=32, Aston Villa=30, Norwich=26}

Please Enter Club Names to find one-on-one Probability Distribution Function
Team names are case-sensitive
Choose Home Team: India
Choose Away Team: Australia

Sorry! You seem to have entered wrong Club details
Club names are case-sensitive
Please enter the valid English Premier Clubs
```

The exceptions are gracefully handled whenever user fails to give the valid inputs

8. DISCRETE V/S CUMULATIVE PROBABILITY CONUNDRUM AND ITS EXAMPLE

Take the below screenshot and look at the teams chosen. For a pair of teams like Everton (Home team) and Newcastle (Away team). Their individual final score probability comes out to be (E : N) 1.74 : 1.47. *Pretty close, eh!*

Running a Poisson distribution over these two inputs we get a Matrix of different combinations of score and their corresponding probabilities. ***To-not-so-our surprise, the PDF declares the maximum probability of 10.3% to the final score ending up to 1:1 and resulting the match in a draw!***

If all the probabilities are added where the scores are equal on both sides. i.e., if all the numbers on an imaginary diagonal line (0:0, 1:1, 2:2, 3:3, 4:4, 5:5) are added, **the probability of match resulting in a draw comes out to be 23.15 %**

Similarly, if you add up all the numbers above that diagonal, we get the cumulative probability suggesting that the **home team Everton might win are added up, the result comes out to be 43.22%**

Similarly, if you add up all the numbers below that diagonal, we get the cumulative probability suggesting that the **away team Newcastle might win are added up, the result comes out to be 32.33%**

This way, the highest cumulative probability suggests us that Everton might win, but the discrete probabilities of each score combination suggest that the match end up draw with score 1:1.

This may be mainly because the cumulative draw probability has only 6 cells on the diagonal. While win or the lose probabilities have 15 cells on each side of the diagonal. So, this might be causing the results to be a little skewed and not giving enough chances for the result to come out to be a draw.

```

<terminated> Driver (1) [Java Application] C:\Program Files\Java\jdk-13.0.2\bin\javaw.exe (17-Apr-2020, 4:29:06 pm - 4:29:27 pm)
4 Wolves 62
5 Chelsea 62
6 Sheffield United 61
7 Man United 59
8 Tottenham 57
9 Arsenal 55
10 Burnley 49
11 Southampton 44
12 Crystal Palace 41
13 Everton 41
14 West Ham 39
15 Newcastle 39
16 Brighton 34
17 Bournemouth 32
18 Watford 32
19 Aston Villa 30
20 Norwich 26

{Liverpool=107, Man City=85, Leicester=80, Wolves=62, Chelsea=62, Sheffield United=61, Man United=59, Tottenham=57, Arsenal=55, Burnley=49, Southampton=44, Crystal Palace=41, Everton=41, West Ham=39, Newcastle=39, Brighton=34, Bournemouth=32, Watford=32, Aston Villa=30, Norwich=26}

Please Enter Club Names to find one-on-one Probability Distribution Function
Team names are case-sensitive
Choose Home Team: Everton
Choose Away Team: Newcastle

Home Team Everton's final score probability: 1.74 goals
Away Team Newcastle's final score probability: 1.47 goals

Everton's and Newcastle's different score combinations and their corresponding probabilities are as follows:
0 : 0 = 4.02 % 0 : 1 = 5.92 % 0 : 2 = 4.36 % 0 : 3 = 2.14 % 0 : 4 = 0.79 % 0 : 5 = 0.23 %
1 : 0 = 6.99 % 1 : 1 = 10.3 % 1 : 2 = 7.59 % 1 : 3 = 3.73 % 1 : 4 = 1.37 % 1 : 5 = 0.41 %
2 : 0 = 6.09 % 2 : 1 = 8.97 % 2 : 2 = 6.61 % 2 : 3 = 3.25 % 2 : 4 = 1.2 % 2 : 5 = 0.35 %
3 : 0 = 3.53 % 3 : 1 = 5.21 % 3 : 2 = 3.84 % 3 : 3 = 1.88 % 3 : 4 = 0.69 % 3 : 5 = 0.2 %
4 : 0 = 1.54 % 4 : 1 = 2.27 % 4 : 2 = 1.67 % 4 : 3 = 0.82 % 4 : 4 = 0.3 % 4 : 5 = 0.09 %
5 : 0 = 0.54 % 5 : 1 = 0.79 % 5 : 2 = 0.58 % 5 : 3 = 0.29 % 5 : 4 = 0.11 % 5 : 5 = 0.03 %

We predict that the final score for Everton : Newcastle 1 : 1 has the maximum probability around 10.3 %

Chances of Everton winning are 43.22 %
Chances of a Draw are 23.15 %
Chances of Newcastle winning are 32.33 %

```

It is a tricky choice to make and it can be avoided if the data to be considered by enormous amount. Personally, I feel the method B of considering individual scores is more accurate method giving a fair chance to match to end up being a draw!

9. CONCLUSION AND FUTURE IMPROVEMENTS

- This method can further be expanded to consider taking into account the importance of a match from the point of view of survival in league for a team.
- Factors discussed earlier like arch-rivals facing each other may also be taken into account
- The amount data to train the design can also be increased

10. REFERENCES

- <http://www.football-data.co.uk/englandm.php>
- <https://www.hackerearth.com/blog/developers/football-betting-odds-work-using-poisson-distribution/>