



INFO 6205

Ranking System

Project Report

BY:

PALAK RAJKUMAR MUNDRA (NUID: 001393220)

RISHABH SOOD (NUID: 001300579)

INDEX

1) Introduction

2) Aim of the Project

3) Datasets used in the project

4) Project Description

5) Implementation

6) Output

7) Mathematical Analysis

8) Conclusion and Future Implementation

9) References

INTRODUCTION

"Programming isn't about what you know; it's about what you can figure out."

This very quote drove us as a team to explore not just the concepts of algorithms but also nuances of how mathematics is the basis of it.

Ranking System, a vast topic in itself has multidimensional applications associated to it like:

- ◆ **Google's page ranking system:** Used to give each page a relative score of importance by evaluating the quality & quantity of its links.
- ◆ **TrueSkill ranking system:** Used to identify and track the skills of gamers in a game (mode) in order to be able to match them into competitive matches.
- ◆ **Elo ranking system:** Used for calculating the relative skill levels of players in zero-sum games such as Chess.

The English Premier League (EPL), one of the most watched and loved Football leagues is a tournament that brings about 20 teams and millions of fans together every season. Each team here plays 38 matches (19 at home and 19 away) approximately strategizing and streaming 722 matches across one season running from August to May.

The EPL also has its own Ranking system that is used for ranking teams, its players and predicting a probable winning team by the end of every season. The ranking system solely depends on 3 factors:

- ◆ Teams are ranked by total points (Teams receive three points for a win and one point for a draw. No points are awarded for a loss)
- ◆ Followed by the goal difference
- ◆ Finally, the goals scored. If still equal, teams are deemed to occupy the same position.

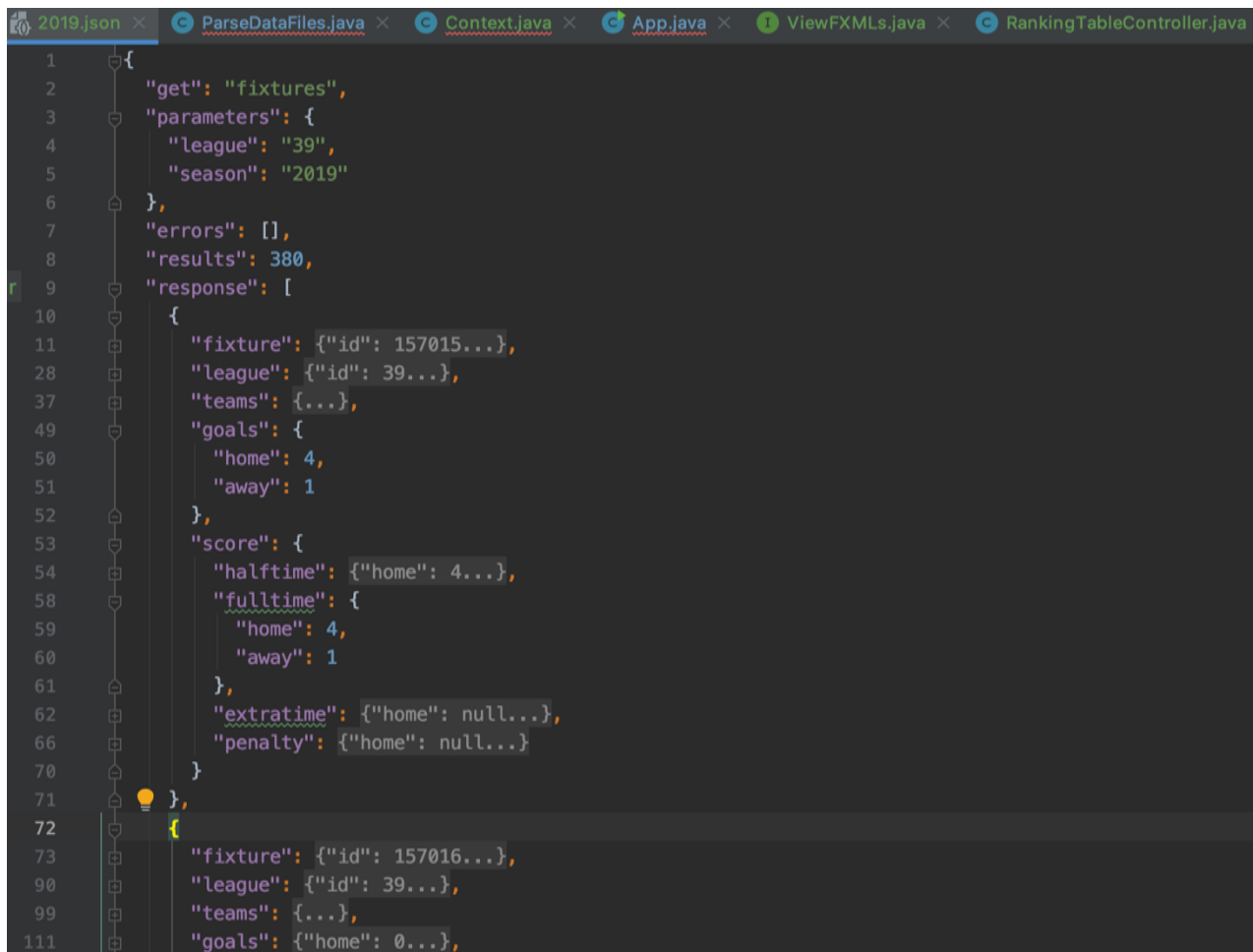
AIM OF THE PROJECT

- ◆ To create a ranking system for the teams in the EPL
- ◆ A predictive analysis of the winning team in EPL 2019 which has been kept on-hold due to Global pandemic, COVID-19
- ◆ A model which can be mingled with many datasets with less processing.
- ◆ Create a comparative matrix between two teams and predict the winning probability when they play match against each other.
- ◆ A user-friendly interface to reflect output generated by the predictive analysis.

DATASETS USED IN THE PROJECT

The dataset used in this project is **API-FOOTBALL (3.4.9)**. The API is used to access all API endpoints, which can get information about Football Leagues & Cups.

The tool uses API keys to allow access to the API. In our project we have called the dataset (manually) and stored it in JSON format, you can refer to the below example:



```
1  {
2    "get": "fixtures",
3    "parameters": {
4      "league": "39",
5      "season": "2019"
6    },
7    "errors": [],
8    "results": 380,
9    "response": [
10     {
11       "fixture": {"id": 157015...},
12       "league": {"id": 39...},
13       "teams": {...},
14       "goals": {
15         "home": 4,
16         "away": 1
17       },
18       "score": {
19         "halftime": {"home": 4...},
20         "fulltime": {
21           "home": 4,
22           "away": 1
23         },
24         "extratime": {"home": null...},
25         "penalty": {"home": null...}
26       }
27     },
28     {
29       "fixture": {"id": 157016...},
30       "league": {"id": 39...},
31       "teams": {...},
32       "goals": {"home": 0...},
```

PROJECT DESCRIPTION

- The project aims at creating user interface predicting the ranking of the teams participating in the EPL.
- The project structure holds 2 main directories:
 - ◆ Java
 - ◆ Resources
- Where the Java directory implements the object-oriented design concepts, definition of data structures, functionalities of the user interface and implementation of the Probability Density function; the Resource directory holds the datasets as well the XML structure definition of the user interface.
- We use Jackson library to convert our data into map structure which is then parsed for predictive analysis. We retrieve the team information participating in the EPL and the match results/fixtures through the ParseData class.
- The Context class, a singleton class is used as a bridge-storage class by the other classes where data retrieval and loading occurs.
- We call our main function in the App class which also initializes the Java FX for the Frontend of the UI.
- The Analyzer class hold the business logic of the entire program. Here the Probability Density function is built which is used for ranking teams in our project.
- The **Probability Density Function (PDF)** uses Goal Difference as a parameter for calculation. Here we consider the summation of the mean of the goal difference –X which is calculated as;

$$\text{Mean} = (X * P(X))$$

X: Goal difference

P(X): Probability of the goal difference

- Note that while finding the probability of the goal difference we **consider the years and provide them weightage according to their precedence.** For example;

Let's say the goal difference between team A and Team B is 3 for the years 2010, 2015 and 2019. While calculating the probability, 2019 would be given a higher weightage, followed by the 2015 and then 2010. (We do this to make sure the frequency of the older years is lesser than the newer ones).

Note: To keep the code and mathematical understanding easier, we have used mean of the probability distribution to predict the goal difference in a match between two teams instead of using probability – weighted randomness.

- Using the above mathematical analysis, design model and data parsing we have achieved:

1) Comparative analysis between two selected teams:

- ◆ This is used to calculate win, loss and draw probability between selected two teams.
- ◆ The output is represented as a probability distribution graph.

2) Generating current ranking of teams based on available data:

- ◆ Here we rank the teams as per current data taken from the external API which shows the total points scored by each team along with the number of matches won, lost and were draw.
- ◆ This table only considers the number of matches that a team has played.

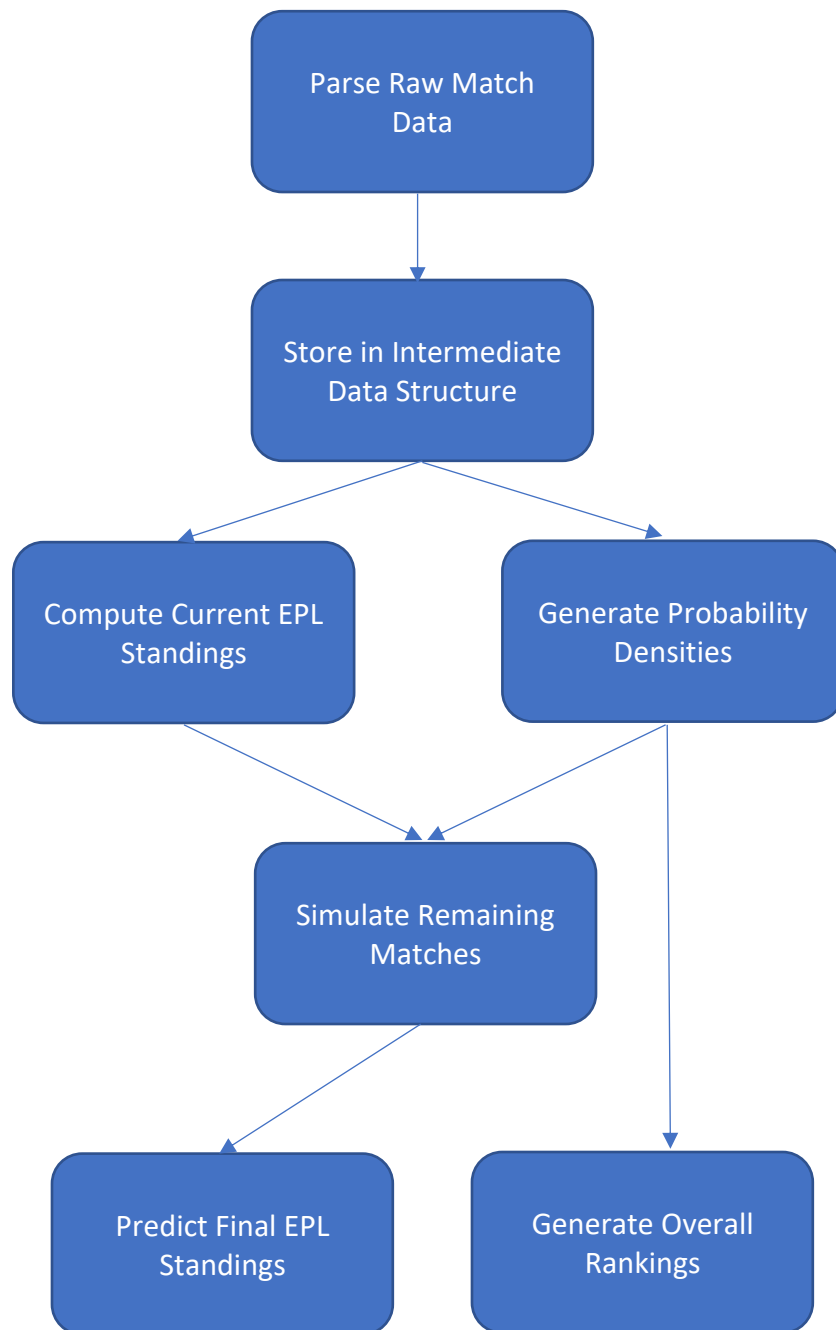
3) Predictive ranking of teams in EPL 2019:

- ◆ Here our main objective is to predict the winning team for the year 2019.
- ◆ This table not only considers the matches that have been played earlier but also considers matches that need to be played and predict if a team is going to win or lose based on the probability distribution function.
- ◆ Here we also highlight the probability score for the prediction simulated by our code.

IMPLEMENTATION

FLOW OF THE PROJECT

Parsing data for generation of current and predictive ranking table



Function to consolidate match statistics and build team wise PDFs:

```
public void buildPdf() {
    List<MatchResult> matchResults = app.getMatchResults();
    Map<Team, List<MatchResult>> matchResultMap = app.getMatchResultMap();
    List<Team> teamList = app.getTeamList();

    //Building a simple map of teams vs all match results for that team
    for (MatchResult matchResult : matchResults) {
        if (matchResultMap.containsKey(matchResult.getHomeTeam())) {
            matchResultMap.get(matchResult.getHomeTeam()).add(matchResult);
        } else {
            List<MatchResult> list = new ArrayList<>();
            list.add(matchResult);
            matchResultMap.put(matchResult.getHomeTeam(), list);
        }
        if (matchResultMap.containsKey(matchResult.getAwayTeam())) {
            matchResultMap.get(matchResult.getAwayTeam()).add(matchResult);
        } else {
            List<MatchResult> list = new ArrayList<>();
            list.add(matchResult);
            matchResultMap.put(matchResult.getAwayTeam(), list);
        }
    }
}
```

Function to build final standings table on the basis of prediction of the rest of the matches which are yet to take place:

```
public void generateOverallRankings() {
    Set<RankingTableModel> rankingTable = app.getRankingTable();
    List<TeamStats> teamStatsList = app.getTeamStats();
    teamStatsList.forEach(teamStats -> {
        RankingTableModel rankingTableModel = new RankingTableModel();
        rankingTableModel.setTeam(teamStats.getTeam());

        double totalMean = teamStats.getStatsMapVsX().values().stream().map(stats -> stats.getProbDensity().getMean()).mapToDouble(Double::doubleValue).sum();
        double totalVariance = teamStats.getStatsMapVsX().values().stream().map(stats -> stats.getProbDensity().getVariance()).mapToDouble(Double::doubleValue).sum();
        double totalStandardDeviation = Math.sqrt(totalVariance);
        rankingTableModel.setWinProbability(getAreaUnderTheCurve(totalMean, totalVariance, totalStandardDeviation, start: 0.1, end: 200));
        rankingTable.add(rankingTableModel);
    });
}
```


Consolidating all PDFs to generate cumulative ranking:

```
public void build2019Standings() {  
    // ...  
    public List<StandingTableModel> predict2019Standings() {  
        Random random = new Random();  
        List<StandingTableModel> standingsTable = new ArrayList<>();  
        app.getCurrent2019standings().forEach(currentStanding -> standingsTable.add(new StandingTableModel(currentStanding)));  
        List<MatchResult> simulated2019Results = app.getSimulated2019Results();  
        for (MatchResult matchResult : app.getMatchResults()) {  
            if (matchResult.getSeason() == 2019 && matchResult.getStatus().equalsIgnoreCase("ns")) {  
                ProbDensity probDensity = p(matchResult.getHomeTeam(), matchResult.getAwayTeam());  
                int predictedGoalDifference = ((Long) Math.round(probDensity.getMean())).intValue();  
                int predictedHomeScore;  
                int predictedAwayScore;  
                StandingTableModel homeModel = standingsTable.stream().filter(model -> model.getTeam().equals(matchResult.getHomeTeam())).find  
                StandingTableModel awayModel = standingsTable.stream().filter(model -> model.getTeam().equals(matchResult.getAwayTeam())).find  
                if (predictedGoalDifference > 0) {  
                    predictedGoalDifference = Math.min(predictedGoalDifference, 5);  
                    predictedAwayScore = random.nextInt( bound: 3);  
                    predictedHomeScore = predictedAwayScore + predictedGoalDifference;  
                    homeModel.setWins(homeModel.getWins() + 1);  
                    awayModel.setLosses(awayModel.getLosses() + 1);  
                } else if (predictedGoalDifference < 0) {  
                    predictedGoalDifference = Math.max(predictedGoalDifference, -5);  
                    predictedHomeScore = random.nextInt( bound: 3);  
                    predictedAwayScore = predictedHomeScore + Math.abs(predictedGoalDifference);  
                    awayModel.setWins(awayModel.getWins() + 1);  
                    homeModel.setLosses(homeModel.getLosses() + 1);  
                } else {  
                    predictedAwayScore = random.nextInt( bound: 3);  
                    predictedHomeScore = predictedAwayScore;  
                    homeModel.setDraws(homeModel.getDraws() + 1);  
                    awayModel.setDraws(awayModel.getDraws() + 1);  
                }  
            }  
        }  
    }  
}
```

OUTPUT



PROBABILITY DISTRIBUTION BETWEEN TWO TEAMS

Overall Rankings



















Rank	Logo	Team	Probability Score
1		Liverpool	0.9970023775551949
2		Chelsea	0.9935981202481663
3		Manchester United	0.9860614889853038
4		Tottenham	0.9818312439892565
5		Arsenal	0.9799775021262686
6		Manchester City	0.9406664119586099
7		Leicester	0.8532353055419241
8		Everton	0.7899969779358976
9		Sheffield Utd	0.7808471882822947
10		Wolves	0.6225690493741702
11		Southampton	0.4910818759031876
12		West Ham	0.3865180736025466
13		Crystal Palace	0.3611872644906455
14		Blackpool	0.25647282763194906
15		Stoke City	0.25355587513932154
16		Birmingham	0.2493604618150521
17		Burnley	0.24225551891813427
18		Bolton	0.19217065198392136
19		Bournemouth	0.16495300794789977

OVERALL RANKING OF THE PARTICIPATING TEAMS

EPL Ranking System

View Overall Rankings



















Current Standings

Logo	Club	Played	W	L	D	GF	GA	Diff	Points
	Liverpool	29	27	1	1	66	21	45	82
	Manchester City	28	18	7	3	68	31	37	57
	Leicester	29	16	8	5	58	28	30	53
	Chelsea	29	14	9	6	51	39	12	48
	Manchester United	29	12	8	9	44	30	14	45
	Wolves	29	10	6	13	41	34	7	43
	Sheffield Utd	28	11	7	10	30	25	5	43
	Tottenham	29	11	10	8	47	40	7	41
	Arsenal	28	9	6	13	40	36	4	40
	Burnley	29	11	12	6	34	40	-6	39
	Crystal Palace	29	10	10	9	26	32	-6	39
	Everton	29	10	12	7	37	46	-9	37
	Newcastle	29	9	12	8	25	41	-16	35
	Southampton	29	10	15	4	35	52	-17	34
	Brighton	29	6	12	11	32	40	-8	29
	West Ham	29	7	16	6	35	50	-15	27
	Watford	29	6	14	9	27	44	-17	27
	Bournemouth	29	7	16	6	29	47	-18	27

View Probability Distributions

Predicted Standings

View Simulated Results

Logo	Club	Played	W	L	D	GF	GA	Diff	Points
	Liverpool	34	32	1	1	87	29	58	97
	Manchester City	32	22	7	3	89	33	56	69
	Leicester	34	18	11	5	69	46	23	59
	Chelsea	34	17	10	7	61	42	19	58
	Manchester United	34	16	9	9	59	38	21	57
	Tottenham	34	15	11	8	69	47	22	53
	Burnley	34	15	13	6	47	48	-1	51
	Arsenal	33	12	8	13	55	43	12	49
	Sheffield Utd	33	12	9	12	35	30	5	48
	Wolves	34	11	9	14	44	40	4	47
	Everton	34	12	14	8	48	55	-7	44
	Crystal Palace	34	11	13	10	28	43	-15	43
	Southampton	34	12	16	6	46	60	-14	42
	West Ham	34	11	17	6	50	62	-12	39
	Newcastle	33	9	15	9	31	54	-23	36
	Brighton	34	7	16	11	36	55	-19	32
	Bournemouth	34	8	19	7	37	66	-29	31
	Watford	34	6	17	11	32	62	-30	29

CURRENT STANDINGS vs PREDICTED STANDINGS

Match Date	Home	Away	Full Time Home Goals	Full Time Away Goals
04-18-2020 11...	Crystal Palace	Chelsea	0	4
04-18-2020 1...	Bournemouth	Tottenham	2	7
04-18-2020 1...	Everton	Southampton	1	1
04-18-2020 1...	Watford	Norwich	1	1
04-18-2020 1...	West Ham	Burnley	1	4
04-18-2020 1...	Arsenal	Leicester	2	3
04-19-2020 1...	Sheffield Utd	Wolves	0	0
04-19-2020 1...	Aston Villa	Manchester United	0	3
04-20-2020 1...	Brighton	Liverpool	1	2
04-25-2020 1...	Liverpool	Burnley	6	2
04-25-2020 1...	Bournemouth	Leicester	2	1
04-25-2020 1...	Manchester United	Southampton	3	2
04-25-2020 1...	Norwich	West Ham	2	3
04-25-2020 1...	Watford	Newcastle	1	1
04-25-2020 1...	Wolves	Everton	3	2
04-25-2020 1...	Brighton	Manchester City	0	5
04-26-2020 1...	Sheffield Utd	Chelsea	0	0
04-26-2020 1...	Tottenham	Arsenal	0	1
04-27-2020 1...	Aston Villa	Crystal Palace	0	0
05-02-2020 1...	Arsenal	Liverpool	2	4
05-02-2020 1...	Burnley	Wolves	1	0

SIMULATED RESULTS

MATHEMATICAL ANALYSIS

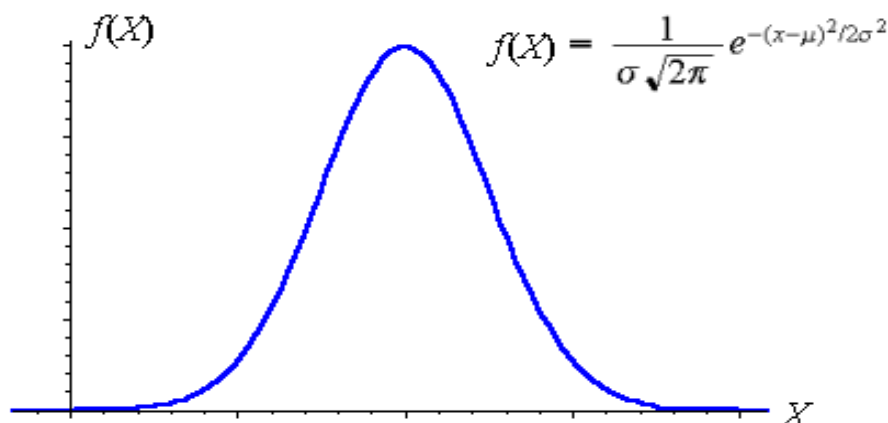
Probability of a specific goal difference between two teams:

$$\sum \frac{(1 - ((2019 - year) \times 0.1)) \times \text{Frequency of Goal Difference}}{\text{Total Matches}}$$

The Normal Distribution:

A random variable X whose distribution has the shape of a normal curve is called a normal random variable

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$$



The Normal Distribution Curve

Probability of Win from Probability Distribution:

Area under the curve on the right of goal difference of 0, i.e.:

$$\int_{0.1}^{\infty} f(x)$$

Consolidating Probability Distribution:

We have considered all matches to be independent of each other. Therefore, we can consider:

Consolidated Mean (μ_s): $\sum \mu$

Consolidated Variance (σ_s^2): $\sum \sigma^2$

Consolidated Standard Deviation: σ_s

Consolidated Probability Distribution Function: $f(x) = \frac{1}{\sigma_s \sqrt{2\pi}} e^{-(x-\mu_s)^2 / 2\sigma_s^2}$

CONCLUSION AND FUTURE IMPLEMENTATION

- Through this project we have learnt about statistical data computation using Probability Density Function and implementing them through a structured code.
- The accuracy of the prediction is close to 67% using mean, variance and standard deviation.
- The area under the curve is represented as follows: Area on the right of 0 (under the curve) represents the win probability while area on the left of the code depicts the loss probability.
- In future, we can optimize our ranking system by adding other deciding factors/constraints like Referee's decision during a game, Home vs Away (location of the match), Probability-Weighted Randomness, etc.

REFERENCES

- <https://www.api-football.com/documentation-beta#section/Authentication>
- <https://mathworld.wolfram.com/Legendre-GaussQuadrature.html>
- <https://www.khanacademy.org/math/ap-statistics/random-variables-ap/combining-random-variables/v/analyzing-distribution-of-sum-of-two-normally-distributed-random-variables>
- <https://www.youtube.com/watch?v=m0o-585xwW0>
- https://www.stat.berkeley.edu/~aldous/157/Papers/aldous_sports_draft.pdf
- <https://www.intmath.com/counting-probability/14-normal-probability-distribution.php>