**INFO6205 Ranking System Project Report**

Group: NotAtAll

Qiushi Zhang 001057253

Yu Chen 001385622

**Introduction**

* **Background**

Due to the spread of COVID19, many industries are impacted and suffering great losses, including football. England Premier League (EPL) has announced that all the matches will be suspended. Therefore, it will be interesting to forecast the final result of EPL in this season. Although there will be a little suspense of champion because Liverpool has been leading the scoreboard too much, the suspense of the Champions League place and relegation. Therefore, we will manage to forecast the rest EPL matches and find some relationship between two teams and their match result.

* **Ranking System**

There are many superior performance ranking systems which utilized in sports. For example, ELO ranking system is one of the most commonly used in sports field such as chess, FIFA national team points and celestial system of League of Legends (LOL). As a matter of fact, ELO Ranking System is our first idea to solve the problem. However, after doing research on ELO and EPL, we found that the existing ranking system of EPL (point +3 if won, +1 if draw and +0 if lost) does not fit the ELO ranking system, that means we have to build up another ranking system to calculate the points of each teams. Furthermore, as far as I am concerned, the essence of ELO ranking system is two points. Firstly, using logistic distribution to estimate the expectation win rate point. Secondly, using a mutable K value to make the point change more reasonable, for example, a better team will not win a lot from a very week team. It might promote us building up a new ranking system to connect with the existing scoreboard of EPL, which will spend too much extra time. Therefore, we will utilize the match data of EPL from Season 00-01 to Season 19-20 to record the relationship between two sides (Home team and Away Team) and the probability of every results (Home Win, Draw and Away win) to estimate the probability of a match result.

**Project Summarize**

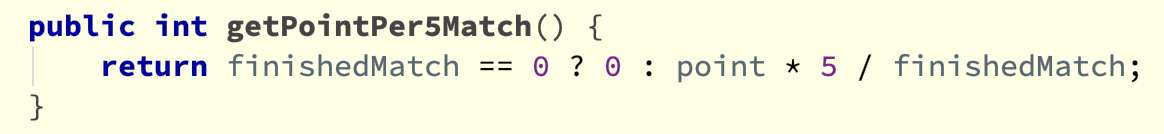
* **Target**

Our target can be summarized in such points. First of all, we will find out a probability of a result of an upcoming EPL match. For example, Everton will play with Liverpool at home in round 30, we will get the probability of the wining rate of home team (Everton), the draw rate and the wining rate of Away team (Liverpool). Furthermore, we will get the expectational point of both teams. We will ‘Kick off’ every match and get the expectational point of home and away teams, which will be added in their total points in scoreboard. Therefore, we will get a final expectational scoreboard. Finally, we will simulate every next match randomly, using the probability of each match calculated before. For example, if the match Everton vs Liverpool is calculated that the wining rate of Everton is 22%, draw rate is 16% and wining rate of Liverpool is 62%, we will use java.util.Random to simulate the match with such probability and get the result randomly.

* **Basic Frame**

The package League is used to simulate the EPL system, which Java classes Team, Match can create every team and match objects. Package Rank is used to build up some basic algorithms and package Main is used to simulate the season and calculate the final point expectation.

To calculate the ability gap between, we cannot just use the current point gap because it may occur that two teams have finished different count of rounds. For example, Team A has finished 10 matches and its point is 25. Team B has finished 15 matches while its point is also 25. We cannot say that team A and B are at the same level because they have different match count although they have the same point. Therefore, we have to calculate point per match to judge the ability gap between two teams. As a matter of fact, we will use floor of 5 \* point / finished match to estimate the team point, because we want a beautiful probability distribution of match results of two teams.



Actually, we can distribute the point gap between two against teams from -15 to 15 according to the equation . To simplify, we will define as .

The next step of the system is read the data from past 19 seasons. We will ignore the irrelevant information such as index of the gambling and keep only home team, away team and the match result (H, D or A, which represent home win, draw and away win). Because of lack of scoreboard in this data, we have to calculate the point of each team before match began. Afterwards, we will put the point gaps between two against teams as the key and an integer array as the value in a HashMap. The array contains three numbers, which respectively correspond to home won counts, draw counts and away won counts when the point gaps between two against teams is the key integer. The result will be written in relationship.csv file, in which the corresponding Home Win / Draw / Away Win rate will be calculated.

A close up of text on a white background

Description automatically generatedA close up of a piece of paper

Description automatically generated

However, there is some “unreasonable” data. For example, we can notice that when *gp* is -15, which means that the away team is too much stronger than home team, there are more circumstances that home team defeated away team. However, it’s easy to understand. We can imagine that after Round 1 of a season, team A won the match in Round 1. Therefore, its score is 3, meanwhile, team B lost the match in Round 1 so its score is 0. In this case, if B would encounter A at home in Round 2, we can calculate the *gp* between B and A is -15. Therefore, when the absolute value of *gp* is big enough (in this case, larger than 10), it will stand a good chance that it happened in the first few rounds of the season, which will course extreme *gp*. Therefore, we will choose *gp* only from -10 to 10, which will be much easy to fit the data.

From the data and the corresponding fitting curve, we can find out that the change tendency of home win rate and away win rate are easy to understand. With the growth of *gp*, the ability gap between home and away teams will increase. Of course, the home win rate will increase, too. The same reason is suitable for the away win rate. However, although the fitting curve of draw rate is also easy to understand, which is when *gp* between home and away team tend to 0, the more similar their strength are, the more possible the game result would be draw. However, we can observe that the degree of fitting is not very high. The fitting curve can only represent the general trend of primitive curve. As a matter of fact, this is the problem of different style of different teams. For some teams such as Arsenal, the attack is weak, but the defense is very stable. The ability for Arsenal to draw the game is much higher than some other teams which is more likely to devote their energy to attack. Therefore, we can only get a tendency of the relationship between *gp* and draw rate. Because it will be much more complete to analyze different teams by their style. We can fix that in this project by calculate the draw rate of a match by the equation: . However, in such case, the draw rate will be a linage tendency, which cannot fit our fitting curve. In order to satisfy the basic equation: , we decide to use equation: , because we find that the squared coefficient of the P(Draw) equation is only -0.001, which means the tendency curve approach to a linear line.

For every match which has been scheduled but haven’t kickoff, we can estimate the possibility of home team win, draw or away team win by utilizing the equation above. According to the possibility, we can calculate the expectational point of each team in each match. The home team’s expectational point is , while the away team’s expectational point is .

With the expectational point equation, we can calculate the expectational point in the end of the season by calculate every match’s expectational point of both teams.

A screenshot of text

Description automatically generated

Because the contingency will be excluded by calculating the expectational point, which means the score board will not looks more dispersedly. However, it’s enough for us to estimate the general ranking in the end of the incomplete season. There is no doubt that Liverpool will win the champion, Norwich City and Aston Villa will fall on eval days, they cannot stay in EPL next season. Bournemouth is very likely to face promotion and relegation play-off.

Next step we will use the java.util.Random to simulate the matches left. Because of uncertainty is getting involved, the scoreboard will look more differently:

A screenshot of a cell phone

Description automatically generated

We can see that Liverpool can almost win the champion in every simulation, because its advantage accumulated is very huge. However, when we focus on the relegation zone, which is the last three teams, we can find that it may change because of the uncertainty. Even from the expectational point table, the gap of the expectational points of the relegation zone is only several points, which means that the situation may change a lot if one of the teams won one more match. As the prober says: The football is round, everything could happen. Therefore, the expectational scoreboard can just provide us a possibility of the last result. We cannot deny that the result must be filled with complexity.

**Java Implementation**

In order to statistic the relationship between *gp* and the match result, we will put every relationship in a HashMap whose key is *gp* and value is an array with counts of three different results:

**A screenshot of a cell phone

Description automatically generated**

To calculate the probability of three results:

A screenshot of a cell phone

Description automatically generated

To simulate one match with calculated probability of three results:

A screenshot of a cell phone

Description automatically generated

To calculate the expectational scoreboard:

A screenshot of a cell phone

Description automatically generated

**Limitation of the Algorithm**

* **Computational accuracy**

When Java calculating float and double numbers, it will cause precision loss because its own malpractice. For example, when calculating the expectational scoreboard, we can find that several teams who has very similar expectational points will get different results, like these:

A screenshot of a cell phone

Description automatically generated A screenshot of a cell phone

Description automatically generated

We can find that the position of Sheffield United and Wolverhampton Wanderers are different.

To fix this problem, we can use java.math.BigDecimal to get more accurate calculation. However, it is proved that precision loss will still happen when we handle double value with java.math.BigDecimal. Therefore, we have to accept the error.

* **Not Suitable for Initial Phase**

Our estimation is based on current scoreboard after round 29, which means we can get a general level of each team. That means, we can get a more reasonable *gp* because the current scoreboard and finished matches can embody the general level of each team. However, try to image that if we want to use this algorithm from the first several rounds of one season. We will find that it is hard for us to get a reasonable result, because there will be more contingency. For example, the defending champion Liverpool lost the round 1 unfortunately while Norwich City won the first match luckily. If these two teams met in round two, we will get an extreme *gp* which is 15, which means it’s very hard for Liverpool to win. However, the truth tells us that Liverpool would still have a very high probability to win because the ability gap. Therefore, we can conclude that the algorithm is suitable when there are enough training data which can embody the level of each teams.

**Conclusion**

When facing an upcoming EPL match, we can get such equations to estimate the probability of result: