Social Network Analysis of Indian Premier League (IPL) Cricket Teams and Batsmen.

Mrigank Shukla[1]

[1]*Vellore Institute of Technology (VIT), Chennai, India.*

E-mail: mrigank.shukla2019@vitstudent.ac.in

Abstract

The aim of the paper is to analyze the team and player data using the matches played in the Indian Premier League (IPL) which is franchise-based cricket league played in India, involving players of various cricket playing nations. It is huge franchise and involves millions of rupees. The revenue generated by the league is quite large and the role of the league is very significant for the thousands of cricket players whose performance in this game paves the way for a place in Indian Cricket Team. The paper uses the fundamentals of social network analysis to create a weighted and directed graph for teams participating in this league. The weights of the graph will be based on number of wins against other teams. Then using the diffusion page ranking algorithm ranks of the teams will be based on the "quality" of wins instead of the simple measure of number of wins. Then the main aim of any team is to increase its batting capabilities that can be done by improving the partnerships between different batsmen. Using the various centrality measure I have tried and identified the key batsmen in a team that are effective partnership formers teams should help honing their performance and retain these players as they are very good.

Literature Review

Cricket originated in England is considered one of the most popular sports based on the viewership data. Unlike other major sports such as Baseball [4], Football [5], Basketball [6] has not been investigated up to a significant extent [2]. Unlike deep analysis and prediction-based model development in other data driven networks such as recommendation systems [3] and classification. Limited number of social network analysts diverted their attentions to looking at quantitative cricket data to explore patterns, ranks, monitoring and other similar aspects. The amount of work done on cricket analysis is quite limited when applied in the field of network analysis. The majority of the papers are concerned with the International Cricket Matches especially those of Test and ODI (One Day International). A research in statistical analysis of players to classify rising batsmen against ICC rankings was done by H Ahmed et. Al [1]. Amin and Sharma used the approach of ordered weighted averaging technique mixed with regression structure in order to measure the performance of different players, and to rank T20 batsmen [7]. In another work, Bracewell and Ruggiero developed a parametric chart based on player control for performance monitoring of a batsman in different matches [8]. De Silva et al. employed an approach based off the Bayesian methodology for the evaluation of team strength in ODIs [9]. The authors incorporated home advantage which is that team playing at home have some sort of authority in

the development of the pitch for match as it was in news during recent Indian-England match in year of 2021, the role of choosing to bat or bowl first was utilized in predicting the expected win of a team. An approach which was quite similar was adopted by Allsopp and Clarke, in which the authors employed minimum approach to analyze the team strength in test matches [10]. A different approach was presented by Mukherjee [11], in which network-based approach PageRank [12] was suggested for team and captain ranking. The author utilized the number of winning matches by a team against the other teams of higher competency, but neglected the role of winning margins. It assigned the ranks based on the approach of giving more weightage to a team if it is winning a match against a team which in itself has good ranking. The approach has a crucial flaw it does not consider various factors which can help determine the team ranking directly such as the margin of the win, performance of each player in the match, number of sixes, number of fours and various other factors. Daud et al. [13] pointed out this limitation and presented a newer approach. More precisely, the authors proposed four indices named as Team index, Team-Rank, Weighted Team Rank and Unified Weighted Team Rank. Team index just employs number of win margins in terms of runs or wickets which means that if a team defending the score set in first inning has won. The number of runs by which it has won is a factor in the ranking. It does not incorporate the strength of opponent. Based on the assumption that more wins imply better strength, Team Rank algorithm ranks the teams, but it does not consider any winning margin. To overcome this limitation, Weighted Team Rank algorithm was proposed by the authors, which includes both the number of wins and wining margins. Finally, the authors proposed to merge Team index with Weighted Team Rank and presented Unified Weighted Team-Rank algorithm that incorporates the team strength, number of wins and winning margin. My research paper tries to implement such algorithm using the python code written by me an identify how these algorithms can help in analyzing the various teams and batsmen forming a network together. If time persists, we can improve on the ways which are calculating the weights of the links as these are the key to this paper ma'am.

Proposed Work

Page Rank Algorithm of Team Ranking.

Data was collected from the website of espncricinfo.com for the IPL matches and their results from the year of 2008-2020. For each match the data consisted of the teams which the match was played, date of match, venue of match, result of match, runs or wicket by which match was won. I have analyzed network of teams by taking into the account of head-to-head encounters of competing teams. A match can be represented by a directed link. If a team *i* wins against team *j*, a directed link is drawn from *j* to *i*. A weighted representation of the directed network is obtained by assigning a weight w_{ji} to link, where w_{ji} is equal to the fraction of times team *i* wins against team *j*.

The code used to accomplish the task is given in *Appendix 1*. The number of matches played between two teams are analyzed over the time period of 2008-2020. The team playing matches against each other form a link. Where the outgoing link from team *i* to *j* represent the fraction of

the matches won by team *j* against team *i.* The resulting graph *in Fig.1* is constructed using Gephi software as it gives beautiful graphs that can by analyzed by exporting the data to NetworkX software. The thickness of the edges is proportional to the weights of edges. While size of nodes is based on in-degree strength of nodes.

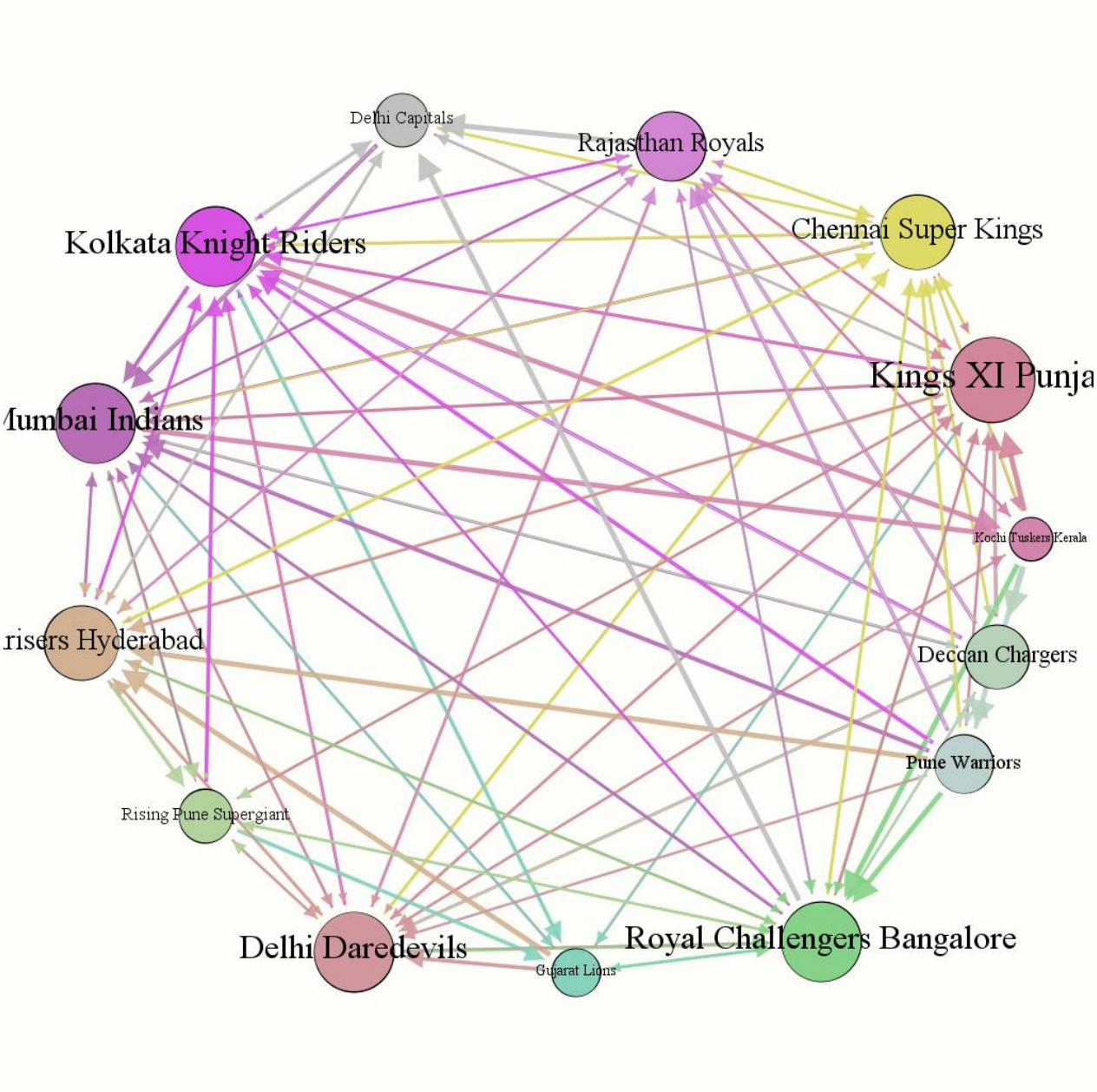

Figure 1. The directed weighted graph for IPL team matches between years 2008-2020 constructed using the approach mentioned above using Gephi.

The edges are colored with respect to the color of the nodes they are directed towards, hence nodes with thicker edges and more incoming edges have won a greater number of matches against the team form which the edges are originating from. The size of the nodes are directly proportional to the in-degree of that node.

The page ranking algorithm for ranking of the reams is as follows.

$$p_i = (1-q)\sum_j p_j \frac{w_{ji}}{S_j^{out}} + \frac{q}{n} + \frac{1-q}{N}\sum_j \delta(S_j^{out})$$

Where w_{ji} is the weight of an edge and the variable $S_j^{out} = \sum_i w_{ji}$ is the out-strength of a link. p_i is the PageRank score assigned to team *i*, and it represents the fraction of the overall "influence" when applied to the steady state of the diffusion process on vertex *i*. The equation is applied to all the teams in the given graph. *q* is in range of [0,1], acts as a control parameter that accounts for the importance of the various terms influencing the score of the rest of nodes, and *N* is the sum of the number of teams in the network. The term $(1-q)\sum_j p_j \frac{w_{ji}}{S_j^{out}}$ is used for the portion of score received by the node *i* in the diffusion process of the hypothesis that nodes redistribute their entire credit to the neighboring nodes. The term $\frac{q}{n}$ stands for a uniform redistribution of credit among all nodes. The term $\frac{1-q}{N}\sum_j \delta(S_j^{out})$ serves as a correction in the case of existence of dangling nodes.

As some of the teams were nor present in all the seasons, a fair analysis consisting of seasons of fixed teams playing matches against each other is done for the years 2018-2020. The teams like Delhi Capitals, Chennai Super kings, Rajasthan Royals, Kings XI Punjab, Kolkata Knight Riders, Sunrisers Hyderabad, Royal Challengers Bangalore, Mumbai Indians were the team who participated in the IPL seasons of 2018-2020. The network analysis diagram constructed with the help of Gephi is presented in the *Figure 2.* The outward directed edge with is of the same color as that of the node from which is originating from.

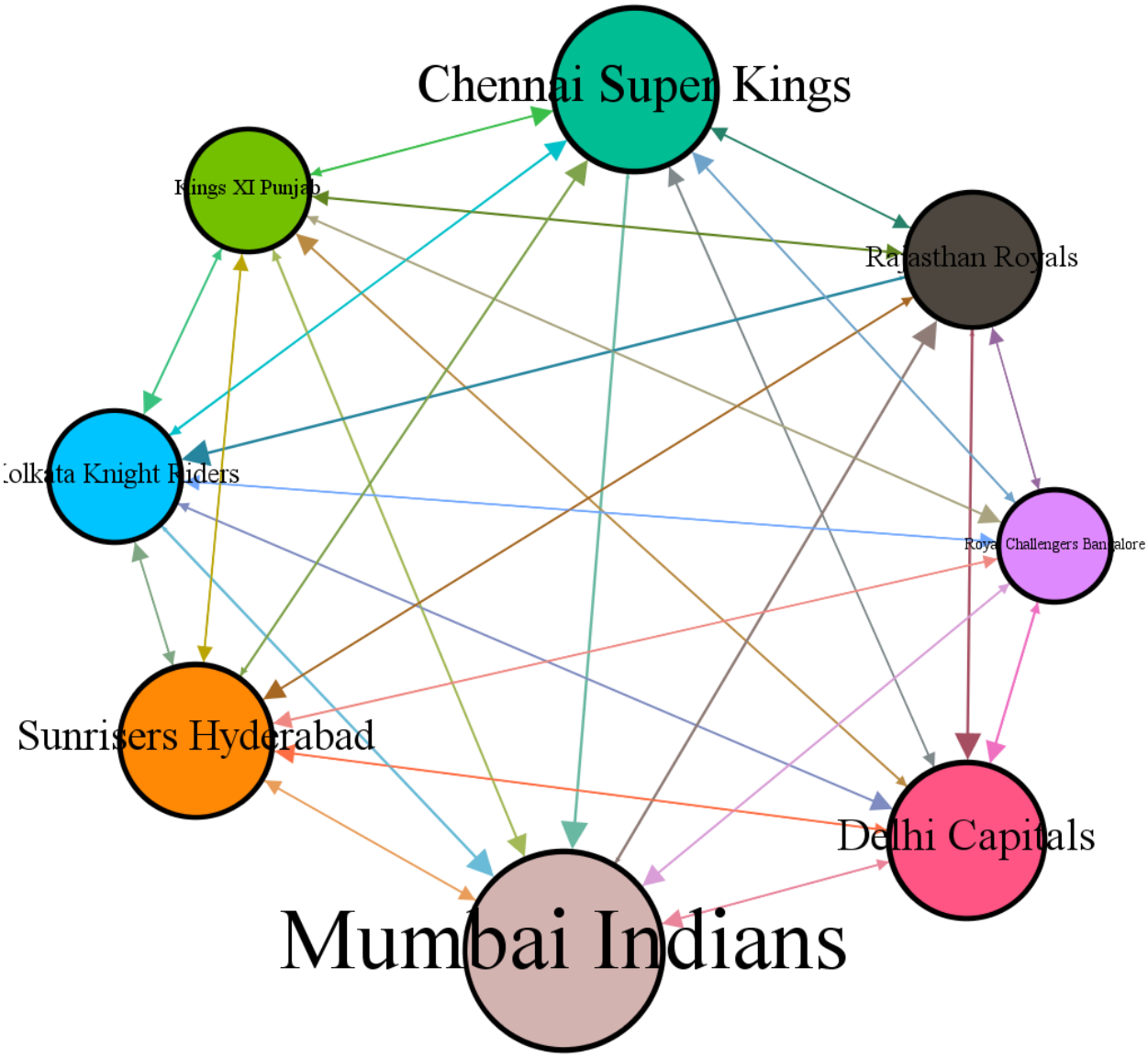

Figure 2. The directed weighted graph for IPL team matches between years 2018-2020 constructed using the approach mentioned above using Gephi.

TeamRank-Index for Team Ranking

Hirsch [14] proposed an indexing measure that is used to rank or index scientists taking into account his/her productivity in terms of papers published and citations received by her papers. The number of papers published by a researcher and the number of times each of those papers are cited provides productivity and impact of his works in the scientific community. It is defined mathematically as

$h = \sqrt{\frac{N_c T}{a}}$ where $N_c T$ is used for denoting the sum of citations received by a scientist for his all papers and "a" is a proportionality constant whose value ranges between 3 and 5.

TeamRank-Index is an adoption of h-index. In *TeamRank-Index* the teams are mathematically equivalent to an author and runs or wickets by which matches are win can be considered equivalent to papers in the h-index. If a team wins a match then by the number of runs or wickets by which the match was run will decide *TeamRank-Index.* The calculation can be explained if a team has won a total of 10 matches, out of these 10 matches 6 matches were won on the basis of number of wickets and 4 matches were won on the basis of number of wickets. *Table 1* gives the matches won by wickets for the above example. *Table 2* gives the matches won and the corresponding margin of the runs by which the match was won. This will help in calculation of *TeamRank-Index*

CALCULATING T1	
MATCHES	**NUMBER OF WICKETS**
2	8
3	9
4	4
5	6
8	1
10	5
TOTAL WICKETS	**33**

Table 1. Example table for calculating T1 value based on number of wickets a match has been won.

CALCULATING T2	
MATCHES	**NUMBER OF RUNS**
1	40
6	20
7	45
9	34
TOTAL RUNS	**139**

Table 2. Example table for calculating T2 value based on number of runs margin by which a match has been won.

$$T_1 = \frac{\sqrt{Total\ number\ of\ Wickets}}{2} = \frac{\sqrt{33}}{2} = 2.8733$$

$$T_2 = \frac{\sqrt{Total\ number\ of\ Runs}}{2} = \frac{\sqrt{139}}{2} = 5.8949$$

$$TeamRank - Index = \frac{\sqrt{T_1 + T_2}}{2} = \frac{\sqrt{8.7682}}{2} = 4.3841$$

In this manner the *TeamRank-Index* is calculated for all the teams. Initially the teams who participated in the IPL for season 2018-2020 are extracted from the data. Then number of wins for each team is calculated and for each win whether it was based on the runs or by wickets is taken into account. This helps calculate the T_1 and T_2 values for each team. Based on these values the *TeamRank-Index* is calculated. This approach helps us take into account the effort put by each team. If a team is able to win a match based on defending a large number of runs, this means that their bowling line up is really good. If a team is able to win a match with a greater number of wickets left in hand, this implies that the team has great batting line up. This helps quantify the win instead of simply treating a win as single measure. This helps rank the teams based on their efforts in the match which led to then winning the match. However, this approach neglects the quality of the opposite team against whom the match was won.

Weighted PageRank with TeamRank Index (WPTR)

This approach is the enhanced version of the PageRank algorithm as here the weight of an edge is calculated differently than simply calculating the ratio of wins to losses against a particular team. I have used number of suns and wickets from which the matches were lost in calculating weights of out links from a node/team. The main reasoning is that if a team is losing a match with greater amount of runs and wickets, that team is in itself a weak team and hence will contribute less to the rank of team from which it lost. To show this with an example consider two teams A and B which have both lost 15 matches. If we simply calculate the edge weight using the old approach of PageRank weight allocation both teams get a score of 1/15. This approach is different, if Team A loses with sum total of 300 runs and 35 wickets in those 15 matches. Whereas Team B loses with sum total of 100 runs and 25 wickets. Using the modified edge weight formula mathematically defined as

$$Weighted\ Outlink = \frac{60 * (lost\ matches) + 20 * (runs) + 20 * (wickets)}{lost\ matches + runs + wcikets}$$

The value of 60,20,20 are taken as even though the margin of win matters, but the final result of the match has much more weightage, Hence the weight of 60 is given to the match result, 20 is allotted to the run margin and the rest 20 for the number of wickets by which all those matches were won/lost.

Then for Team A the weighted out link will be calculated as 60*(15) +20*(35) +20*(300)/ (15+35+300) which gives the value as 152/7. Doing the same calculation for Team B, 60*(15) +20*(25) +20*(100)/ (15+25+100) gives the value as 41/2. When these value will be used for calculating team ranks they will be in denominator hence the value of team A will be 0.0461 and that of team B will be 0.0488. This proves that the team loosing with lesser margin contributes greater t the score of winning team.

This approach is then combined with the *TeamRank-Index* and *PageRank Approach* to give the modified formula defined as

$$WPTR_i = \frac{(1-d)}{N}\left(\frac{TR_i}{\sum_1^N TR}\right) + d\left[\sum_{j=1,J\neq i}^{N} \frac{WPTR_j}{Weighted\ Outlink_{ji}}\right]$$

Here the $WPTR_i$ is the combined rank calculated using the Weighted PageRank combined with TeamRank-Index. The value *d* is the damping factor with the value between 0-1. I have taken the value of *d* as 0.15. The TR_i is the *TeamRank-Index* for team *i* and the denominator is the sum of *TeamRank-Index* for the rest of the teams which the above team played against. The summation part towards the right, the denominator gives the out-link value originating from the loosing team *j* to the team *i* in all the encounters throughout the season of 2018-2020. This process is repeated

for a total of 1000 times to give a converging value of *WPTR* for all the teams participating in IPL.

Complex Network of Batsmen

The next part is an effort to capture the interactions between batsmen of different teams. Although the success or failure of a particular team depends on the combined effort of it's team members, the performance of individual team members can be very crucial for franchise teams for better selection of players and choosing the playing eleven for a particular match. As both of the teams have to bat, and main aim of process is to set a high target which will be very difficult for the next team to chase. The interaction between different batsmen can be modeled into a network. The batting order is very crucial, the openers lay foundation of an innings by facing the new ball which is very hard to bat against. The lower middle order is responsible for scoring more runs in case of top order failure, as in the case during World Cup 2011 with India, or try to save the wickets if a decent score has been set up by top order. The main aspect in all this is the partnership. A good partnership puts pressure on the bowler by scoring massive runs and tackling the fielding set up. The concept of partnership becomes crucial if a player can act as an anchor for other players to form partnership around. I have tried to identify key players in a team with respect to batting, by constructing the partnership network of different teams playing in IPL seasons from 2018-2020. I have used a set of network centrality measures to gain an insight on their application of identifying key batsman in teams.

Clustering Coefficient

Clustering Coefficient (C_i) of a node *i* is defined as the ratio of numbers of links shared by its neighboring nodes to the largest number of links that are possible among them. Then the average clustering coefficient is defined as,

$$C = \frac{1}{N} \sum_{i=0}^{N} C_i$$

Then this average clustering coefficient (C) will be used to capture the global density of interconnected nodes present in a network.

Betweenness Centrality

Betweenness centrality measures the connecting role a particular node performs in the network. It calculates' s a node's centrality based on the number of shortest paths for the pair of vertices in network which are passing through this node. It is used to find node strengths based on their capability of acting as an intermediary of information flow.

$$C_B(i) = \sum_{j<k}^{n} \frac{g_{jk}(i)}{g_{jk}}$$

Where g_{jk} is the number of shortest paths between *j* and *k* and $g_{jk}(i)$ is the number os times those paths passed through node *i*.

Closeness Centrality

It is meant to measure one node to the others nodes' sum distances, if the length of node N's shortest paths with other nodes in the network is small, then node N has a high closeness centrality [15]. It stands for the convenience and ease of connections between the focused node and the other nodes. The fundamental formula Cc is below given equation $\sum_{j=a}^{n} d(N_i, N_j)$ means the total number of "steps" from node N to the other nodes in the network.

$$C_c\ (N_i) = \frac{1}{\sum_{j=1}^{n} d(N_i, N_j)}\ \ (i\ \neq j\ )$$

Betweenness centrality measures the extent to which a node lies on a path to other nodes. In cricketing terms, betweenness centrality measures how the run scoring by a player during a batting partnership depends on another player. Batsmen with high betweenness centrality are crucial for the team for scoring runs without losing his wicket. These batsmen are important because their dismissal has a huge impact on the structure of the network. So, a single player with a high betweenness centrality is also a weakness, since the entire team is vulnerable to the loss of his wicket. In an ideal case, every team would seek a combination of players where betweenness scores are uniformly distributed among players.

Closeness centrality measures how easy it is to reach a given node in the network. In cricketing terms, it measures how well connected a player is in the team. Batsmen with high closeness allow the option for changing the batting order depending on the nature of the pitch or match situation.

Clustering Coefficient of a team in the networks constructed gives us a measure of how the players within the team are interacting, a high clustering coefficient gives us an idea that the network is highly clustered meaning the interaction between players in the way of partnership is good.

I have constructed a weighted and undirected graph between players, of the same team where the weights if the link between two players is the sum of the run scored by them throughout the three seasons. Hence players which are having more links are better in forming partnership and hence are quite crucial in teams. I have then run the following algorithms explained above to these team networks to get the values of these centrality on different players. The graphs for the various teams are given below.

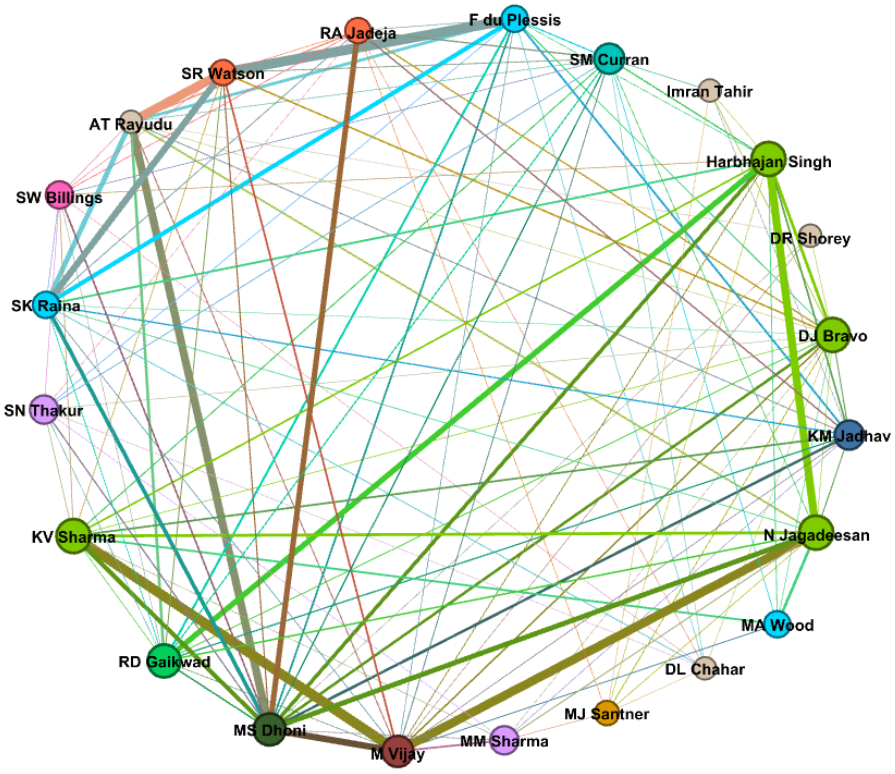

Figure 3. The Batting Partnership Network for Chennai Super Kings.

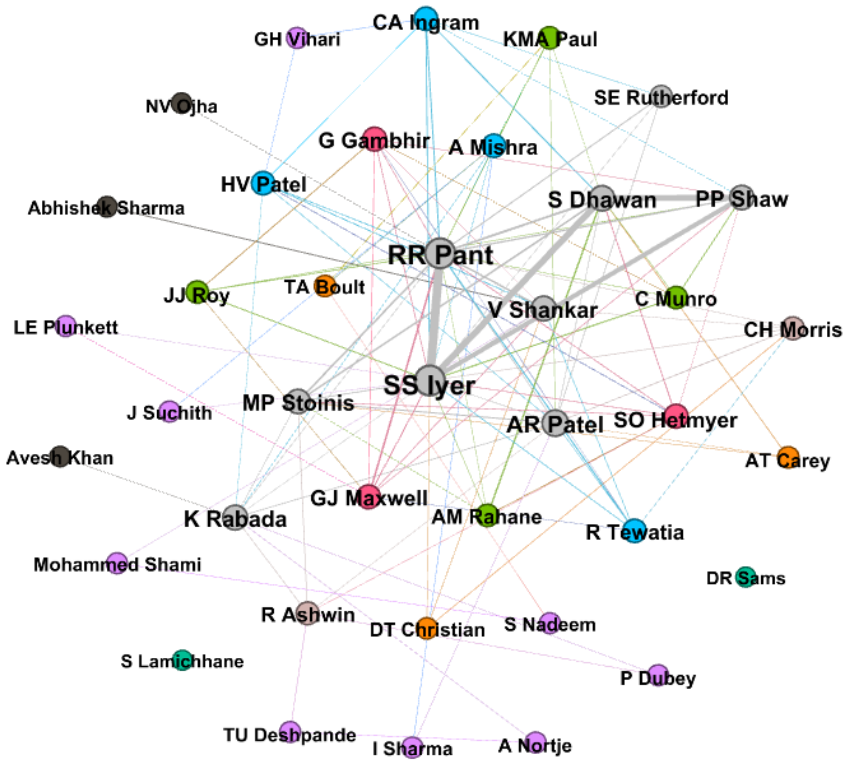

Figure 4. The Batting Partnership Network for Delhi Capitals

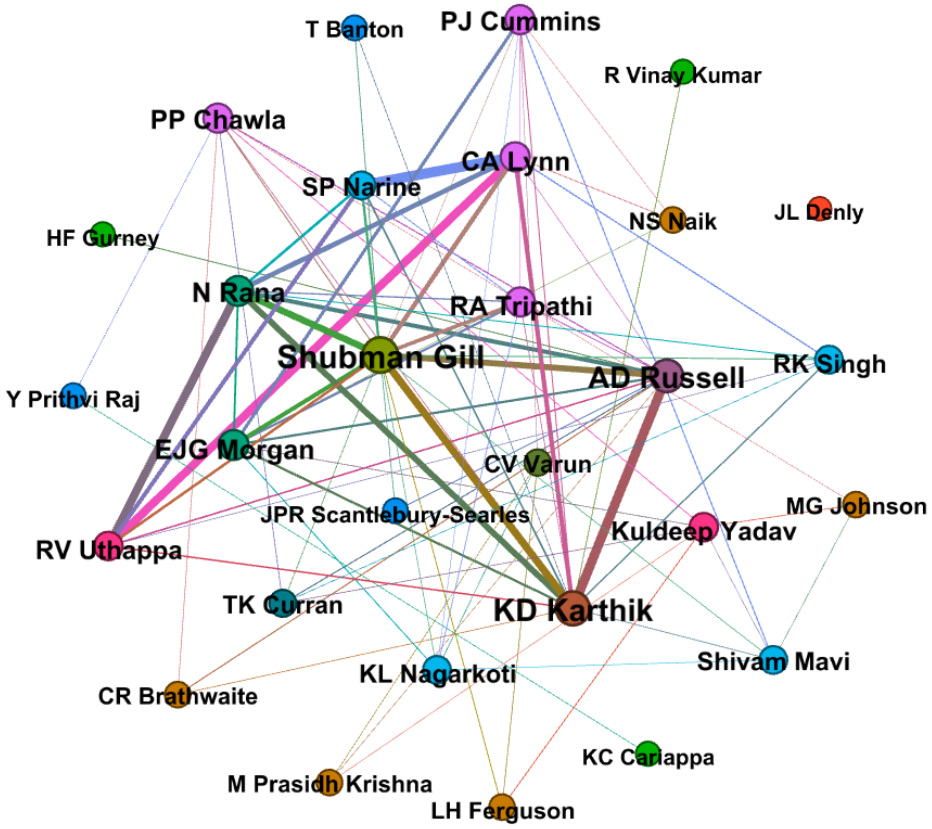

Figure 5. The Batting Partnership Network for Kolkata Knight Riders.

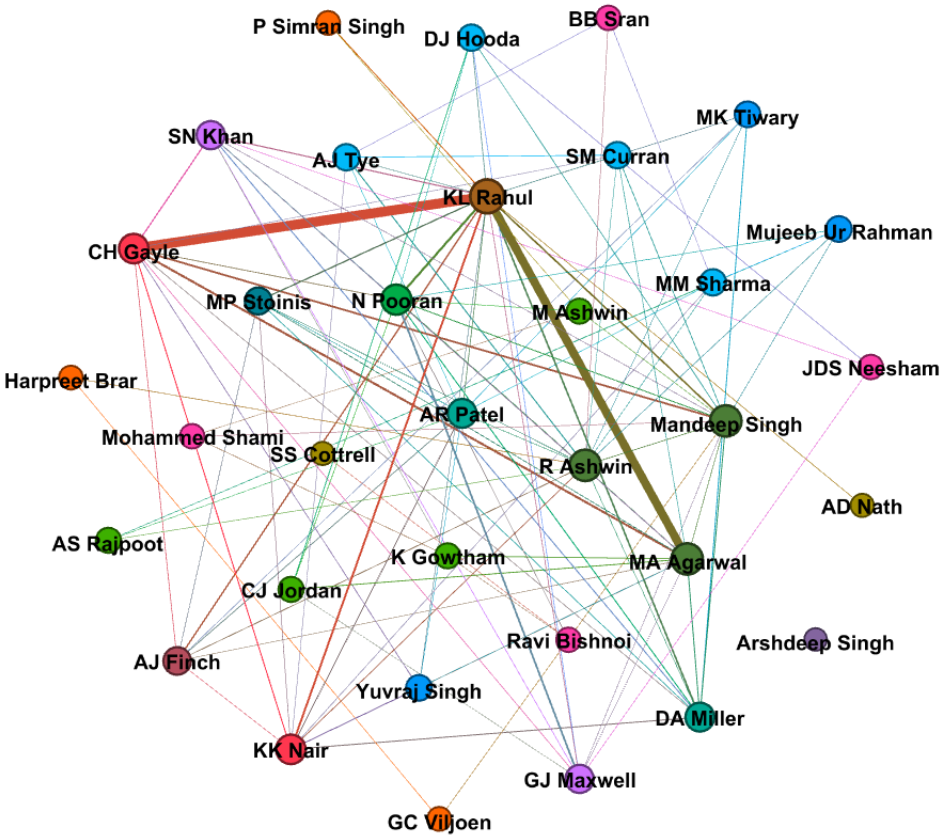

Figure 6. The Batting Partnership Network for Kings XI Punjab.

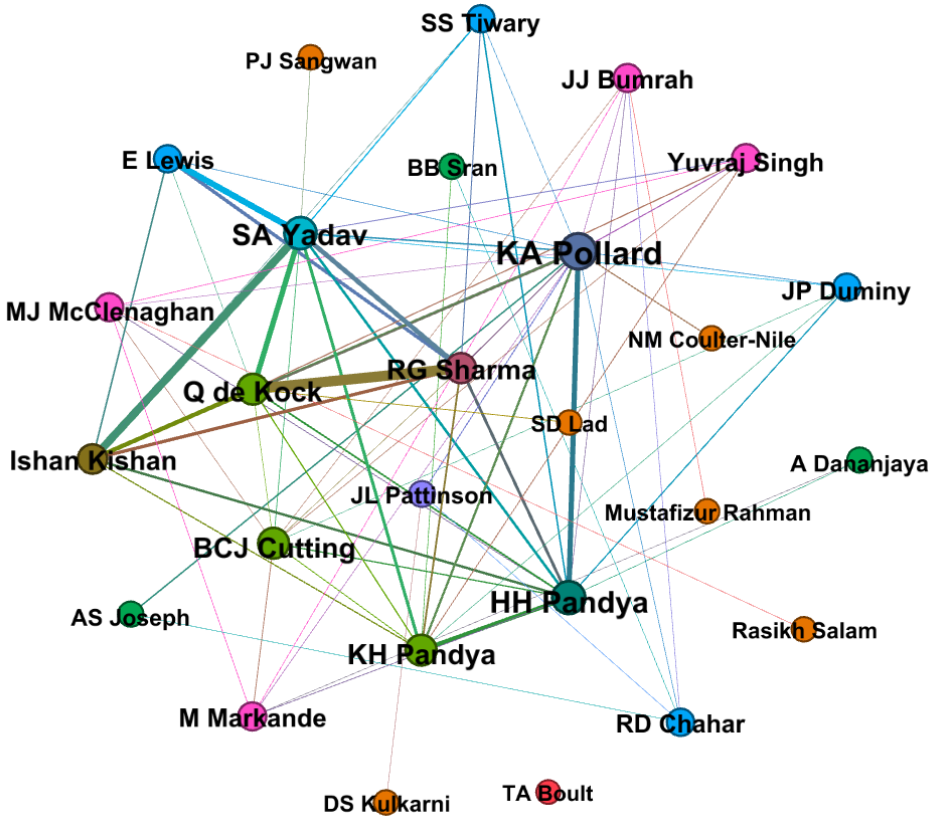

Figure 7. The Batting Partnership Network for Mumbai Indians.

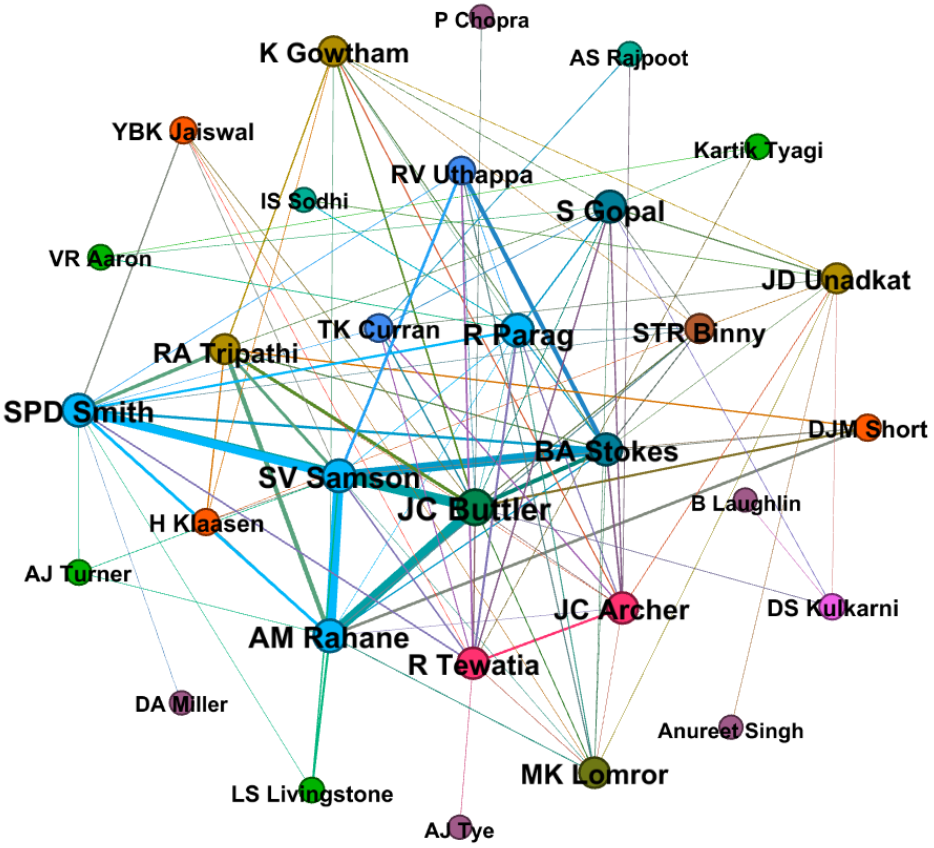

Figure 8. The Batting Partnership Network for Rajasthan Royals.

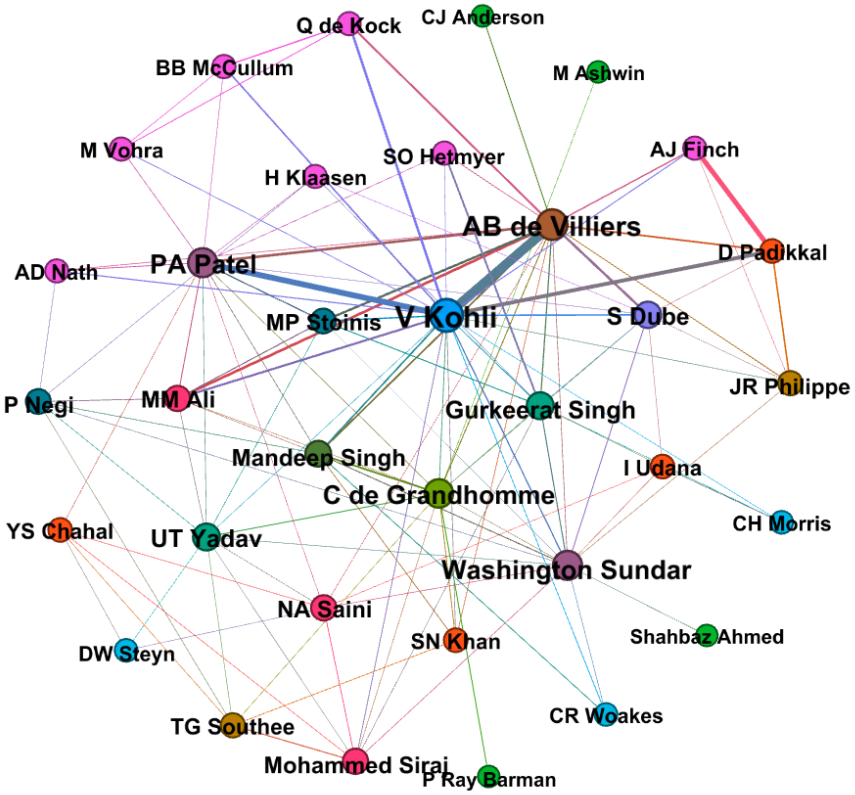

Figure 9. The Batting Partnership Network for Royal Challengers Bangalore.

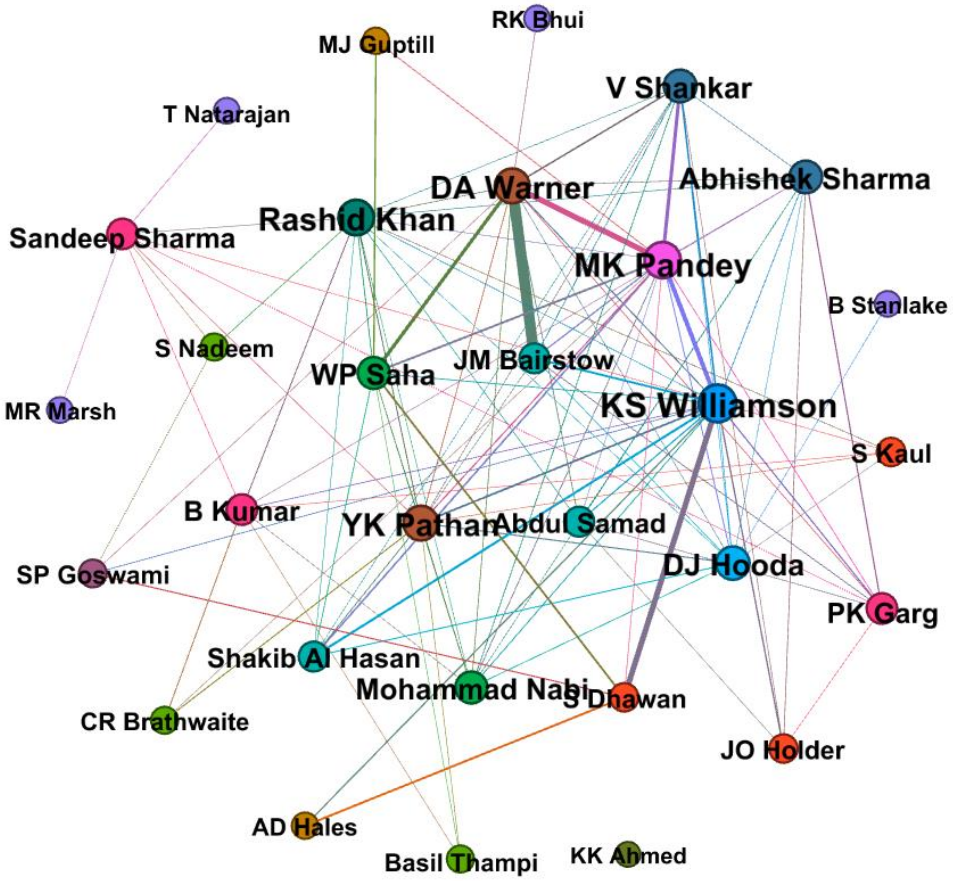

Figure 10. The Batting Partnership Network for Sunrisers Hyderbad.

Results and Discussions.

Page Rank Algorithm of Team Ranking.

The results of running the page ranking algorithm whose code is given in the *Appendix 2.* The results are categorized in the tabular format given in *Table 3.*

TEAM NAME	PAGERANK SCORE
Gujarat Lions	0.012768723128150422
Pune Warriors	0.012768723128150422
Kochi Tuskers Kerala	0.012370091249498012
Kings XI Punjab	0.0121267114363177
Delhi Daredevils	0.012083910656862186
Royal Challengers Bangalore	0.011922778310676719
Rajasthan Royals	0.011820521244828248
Mumbai Indians	0.011741504421218067
Delhi Capitals	0.011536060679831598
Kolkata Knight Riders	0.011471183708867448
Deccan Chargers	0.011466924593927667
Sunrisers Hyderabad	0.011396087993490778
Chennai Super Kings	0.011301267832532774
Rising Pune Supergiant	0.01122789506775189

Table 3. The table ranking the different teams based on the page ranking algorithm for teams playing in years 2008-2020.

The analysis of the page ranking is accurate with the results of the points table and the general performance of the teams during IPL. The effect of the number of wins and the number of losses is now affected by the teams against which the matches were won. There is small flaw in the analysis of the teams, even though Chennai Super Kings have won many IPL seasons and has been a constant performer. The ranking of the team is quite low, this is due to the two years ban

to Rajasthan Royals and Chennai Super Kings in the season of 2016 and 2017 of IPL. The teams such as Gujarat Lions, Pune Warriors and Kochi Tuskers Kerala played for these two seasons as a replacement. In these two seasons they were top performers. Hence the number of matches played is neutralized by the performance in matches. Therefor a separate analysis for the teams playing from the years of 2018-2020 has been performed using similar methodology and the Page Ranking are given in the *Table 4.*

TEAM NAME	PAGERANK SCORE
Chennai Super Kings	0.022687855078591578
Kolkata Knight Riders	0.021750270536069774
Kings XI Punjab	0.021375236719061053
Mumbai Indians	0.021375236719061053
Sunrisers Hyderabad	0.021362220360987
Royal Challengers Bangalore	0.02100020290205233
Delhi Capitals	0.02071892753929579
Rajasthan Royal	0.020500157812707368

Table 4. The table ranking the different teams based on the page ranking algorithm for teams playing in years 2018-2020.

This ranking is more accurate and semantically correct as each team has participated in a match played against all other teams. The results are in sync with the points table for the seasons. As the winner of the 2018 season was Chennai Super Kings played against Sunrisers Hyderabad, for the 2019 Season the Winner was Mumbai Indian winning against Chennai Super Kings, for the 2020 season the winner was Mumbai Indians winning against Delhi Capitals. The teams like Kolkata knight Riders and Kings XI Punjab are having higher ranking even though winning no tournament trophy this can be explained by the fact that these teams always got knocked out in the playoffs, teams like Mumbai Indians and Sunrisers Hyderabad performance were not stellar throughout these seasons but they were able to perform well during the playoffs and the finals. Hence though Mumbai Indians have clinched the trophy twice in the year 2019 and 2020. The matches won throughout the seasons were not much but were just enough to qualify them for finals.

TeamRank-Index for Team Ranking

The results of running the *TeamRank-Index* algorithm whose code is given in the *Appendix 3.* The results for the same are given in *Table 4.*

TEAM NAME	T_1 VALUE	T_2 VALUE	TEAMRANK-INDEX
Mumbai Indians	4.444097208657794	12.031209415515	2.0294892598985195
Sunrisers Hyderabad	4.6097722286464435	10.198039027185569	1.9240459490246076
Kolkata Knight Riders	4.092676385936225	9.447221813845593	1.8398300328958255
Delhi Capitals	3.427827300200522	9.912113800799505	1.8261942052394116
Chennai Super Kings	5.454356057317857	7.262919523166975	1.7830644674607836
Royal Challengers Bangalore	3.872983346207417	6.800735254367722	1.6335328739097308
Kings XI Punjab	3.9370039370059056	6.557438524302	1.6197563444317717
Rajasthan Royals	4.2130748865881795	4.743416490252569	1.4963698888343708

Table 4. The TeamRank value of teams as a cumulative of the performance in IPL through seasons of 2018-2020.

The result can be understood if we take into account the quality of wins. If a team A wins lesser number of matches than a Team B, but all the matches A has won are by a large margin of runs and wickets this will give it a higher ranking than Team B who has won a greater number of matches than Team B but with lesser margins than Team A. This is in sync with Points Table, as though some teams may have won lesser number of matches giving them a lower rank in Page Rank approach previously discussed. But the matches they have won are with huge margins which can tell us with more certainly about the performance of team than simply winning or losing. This approach does not take into account the performance in comparison to team against with match was played which we took in Page Rank approach. As winning against a weaker Team with greater margin is not that impressive than defeating a good team with lesser margin.

Weighted PageRank with TeamRank Index (WPTR)

The result of running this algorithm for all the teams using the code given in *Appendix 4.* The results obtained are given as follows in *Table 5.*

TEAM NAME	WPTR SCORE
Mumbai Indians	0.41491955898012656
Kolkata Knight Riders	0.3763703378041377
Delhi Capital	0.37336718062878793
Chennai Super Kings	0.3645706703487219
Royal Challengers Bangalore	0.3340185375098747
Sunrisers Hyderabad	0.33396147842871265
Kings XI Punjab	0.33119585318124534
Rajasthan Royals	0.3062679902952873

Table 5. The team names and the calculated WPTR score used to rank the teams.

This result is the most accurate rating system as compared with other approach. As it takes into account the margins of runs and wickets by which each team has won. Though previously teams such as Kings XI Punjab and Sunrisers Hyderabad score better as they won more matches as compared with other teams, but if we take into account the runs and wicket margin, teams such as Mumbai Indian and Kolkata Knight Riders have won with large margins and defeated teams which themselves have high ranking. Chennai Super Kings has seen large slip in performance as though it performed largely well in the season of 2018 and 2019, it failed to perform in the season of 2020. The victory margins during those two years could only compensate for huge losses during the year of 2020. Mumbai Indians have come at top. Proving that they deserve to be the champions in two consecutive years of 2019 and 2020.

Complex Network of Batsmen

The first way is to analyze all the given teams at once using the average clustering coefficient of the teams. Teams with higher average clustering coefficient are having good partnership formation between players which tells us that they will tend to perform better, but it is not guaranteed as we are not considering the bowling attack of the teams. A team with poor batting performance can always compensate with good and efficient bowling attack.

TEAM NAME	A**VERAGE CLUSTERING COEFFCIENT**
Chennai Super Kings	0.679
Delhi Capitals	0.546
Kolkata Knight Riders	0.503
Kings XI Punjab	0.474
Mumbai Indians	0.613
Rajasthan Royals	0.605
Royal Challengers Bangalore	0.570
Sunrisers Hyderabad	0.585

Table 6. The average clustering coefficient of teams.

The teams such as Mumbai Indians and Chennai Super Kings are the teams with highest clustering coefficient this can due to the lesser number of changes in the whole structure of teams both of these teams are known for retaining most of the players, and also allowing greater variation of playing eleven giving players to form partnership with different team members. The team Kings XI Punjab is having lowest clustering coefficient as the team has many players and some of the players are not given chances in playing eleven this allows only limited members to form the partnership during gameplay.

Chennai Super Kings

Player Name	Betweenness Centrality	Player Name	Closeness Centrality
DJ Bravo	0.111406	DJ Bravo	0.913043
SM Curran	0.051918	MS Dhoni	0.840000
MS Dhoni	0.044962	SM Curran	0.777778
Harbhajan Singh	0.036697	M Vijay	0.777778
M Vijay	0.036221	Harbhajan Singh	0.750000

Table 7. The top five players according to the centrality measures.

The player DJ Bravo is the player with highest betweenness centrality this is correct according to real life statistics DJ Bravo plays in the lower middle order and is an explosive player, whenever the top order collapses DJ Bravo handles the situation, he has the strike rate of 154.94 which is very good for a last order hitter. The same can be said for players such as SM Curran which was the player who single handedly managed to take care of the batting order of Chennai Super Kings in the year 2020 when most of the top order players were failing to perform. The players shown here are quite good for team structure as MS Dhoni and Murli Vijay are top order players, which are good batsmen. These two players are handling the team in top order forming good partnership setting a good starting score, the players like Harbhajan Singh, SM Curran and DJ Bravo are primarily bowlers and get to bat at middle and lower middle order, this gives the team an added stability as both the top order and lower middle order are able to set a good batting score. This allows a good fallback option to the team in case the top order fails.

Delhi Capitals

Player Name	Betweenness Centrality	Player Name	Closeness Centrality
SS Iyer	0.225988	SS Iyer	0.679245
RR Pant	0.206925	RR Pant	0.679245
K Rabada	0.157802	AR Patel	0.580645
V Shankar	0.121337	K Rabada	0.553846
AR Patel	0.103403	S Dhawan	0.529412

Table 8. The top five players according to the centrality measures for Delhi Capitals.

The players identified by betweenness centrality and closeness centrality measures coincide with the match finding. Shreyas Iyer has a massive strike rate of 132.93, scoring constantly throughout the seasons, he has the average of 35.86, this is very good for opening batsmen. Shreyas Iyer forms crucial partnership with the top batting order of other player such as Rishabh Pant, V Shankar and AR Patel who themselves have comparable strike rates. The only contender missing from Betweenness Centrality is Shikhar Dhawan this is due to his absence in the season of 2018. The only loser middle order batsmen is K Rabada, he is mainly a bowler. Hence it can be seen that the team Delhi Capitals is mainly benefitting from the performance of its top order batsmen. Hence if the top order batsmen fail to perform the team can face some problems as only K. Rabada is able to handle the batting in the lower middle order.

Kings XI Punjab

Player Name	Betweenness Centrality	Player Name	Closeness Centrality
KL Rahul	0.216407	KL Rahul	0.666667
Mandeep Singh	0.174240	Mandeep Singh	0.627451
N Pooran	0.142992	N Pooran	0.592593
R Ashwin	0.138394	MA Agarwal	0.592593
MA Agarwal	0.087071	DA Miller	0.581818

Table 9. The top five players according to the centrality measures for Kings XI Punjab.

KL Rahul is a deserving candidate to be on the top list with the massive average of 54.90. He is a constant performer for the team. Scoring five fifties and one century in the 2020 season of the IPL. He has good batting average this means that he seldom gets out at the beginning of the match. It can be seen that KL Rahul forms good partnerships with top order and middle order players, hence our approach has identified correctly. Mandeep Singh being an all rounder usually bats in the middle order he is known for his explosive performance, it can also be observed that he forms good partnership with the top order players and also the middle order players, which is also true for Mandeep Singh. The only pure bowler appearing on the list is R. Ashwin as he is the last leg hitter, going after the lower middle order, scoring in the last part of the game. Hence it can be said the team has almost a good balance between players of top, middle and lower middle order. This gives the team a great batting advantage.

Kolkata Knight Riders

Player Name	Betweenness Centrality	Player Name	Closeness Centrality
Shubham Gill	0.226133	Shubham Gill	0.710526
KD Karthik	0.182348	KD Karthik	0.675000
AD Russell	0.171440	AD Russell	0.642857
PP Chawla	0.171003	PP Chawla	0.562500
Y Prithvi Raj	0.068783	EGJ Morgan	0.562500

Table 10. The top five players according to the centrality measures for Kolkata Knight Riders.

Shubham Gill has been identified as the player with highest centrality this is in tune with actual gameplay, as he is the opener for Kolkata Knight Riders team. He was the highest run scorer for the team in the year 2020. The most important fact is that being an opener he is very dependable, as he forms good partnership with the openers as well as middle order batsmen, which is very good for a team. This is the reason he has been retained in the team with the contract of Rs 1.8 Crore. KD Karthik being the captain of the team is also very dependable player, he is bats at one down and usually forms good partnership with Shubham Gill and the lower middle order player. This is very good for a team as the top order players are very dependable. AD Russell and PP Chawla are the two all-rounders in the team, they assist by batting and bowling. Russell was the highest scoring player for Kolkata Knight Riders in the year 2019. The only problem here is that there seems to be no lower order player in the centrality measure ranking this implies that the Kolkata Knight Riders are inly dependent on their top order batsmen and the all-rounders.

Mumbai Indians

Player Name	Betweenness Centrality	Player Name	Closeness Centrality
KA Pollard	0.378306	KA Pollard	0.757576
HH Pandya	0.115044	HH Pandya	0.657895
SA Yadav	0.094104	SA Yadav	0.609756
JJ Bumrah	0.844448	Q de Kock	0.581395
Q de Kock	0.082463	KH Pandya	0.581395

Table 11. The top five players according to the centrality measures for Kolkata Knight Riders.

The centrality measure ranking has put the perfect player on the top of the list as the die-hard fans of Mumbai Indians can agree upon. Pollard is an all-rounder player which assists the team with the bat and the bowl. He has a massive strike rate of 191.42. He is the most dependable player on the roster. As he plays in the middle order and forms good partnership with the top order players and the lower order players. The most important thing is that he performs well when the top order fails. HH Pandya and his brother KH Pandya are the two all rounders for the team which deserve their rightful position on the table. As it can be seen HH Pandya also has the massive strike rate of 191.42 proving very useful when paired with Pollard. Another member which is most interesting appearing on the centrality ranking is JJ Bumrah who is the pace bowler for Mumbai Indians, his being in the centrality listing is due to his excellent assist to middle order players, he is one of the big hitters but is very dependable in case of top order failure to perform. Q de Kock and SA Yadav are the top order players who are excellent batsmen both of these player have been the top three scorers in all the three seasons for Mumbai Indians. The ranking has given us an insight into the strength of Mumbai Indians this is due to the fact that they have two all-rounders, two top order players and a bowler who are the players excellent in forming partnerships. This gives the team all the chances in case of failure to perform by any of the players.

Rajasthan Royals

Player Name	Betweenness Centrality	Player Name	Closeness Centrality
JC Butler	0.180350	JC Butler	0.794444
SPD Smith	0.124222	R Parag	0.722222
R Tewatia	0.107791	SPD Smith	0.722222
JD Unadkat	0.101685	SV Samson	0.716667
S Gopal	0.090459	BA Stokes	0.700000

Table 12. The top five players according to the centrality measures for Rajasthan Royals.

The top position is occupied by the most dependable player for the Rajasthan Royals Teams purchased for the whooping price of Rs. 4.4 Crore. He maintains an excellent average of 54.80. He is the opening batsmen for the Rajasthan Royals Team, which is very good as opening batsmen forming good partnership with lower order batsmen signifies an excellent opener. SPD Smith plays for the middle order and is an excellent middle order batsman, he gives the team a solid standing in the middle order play, forming good partnership with the lower order batsmen. R Tewatia is a bowler for the Rajasthan Royals team but has been made an all-rounder due to his massive hitting capability. JD Unadkat and S Gopal are both lower order batsmen which are primary bowlers for the teams. The main problem for the Rajasthan Royals can be identified is the absence of all-rounders in the centrality ranking a presence of most bowlers, usually a bowler appearing in the list is considered good as it gives some dependency for the team in case of top order failure, but three bowlers is worrying signifying constant dependence of batting team on the bowlers. Also, there is large difference in players appearing on closeness centrality and the betweenness centrality, this gives us an idea that most of the players who can give an option to change the batting order are not forming good partnerships. Hence Rajasthan Royals team should aim to replace the players on betweenness centrality with the closeness centrality players, this will definitely help the team.

Royal Challengers Bangalore

Player Name	Betweenness Centrality	Player Name	Closeness Centrality
Virat Kohli	0.180350	Virat Kohli	0.794444
AB De Villers	0.124222	AB De Villers	0.722222
C de Grandhomme	0.107791	Washington Sundar	0.722222
Washington Sundar	0.101685	C de Grandhomme	0.716667
PA Patel	0.090459	PA Patel	0.700000

Table 13. The top five players according to the centrality measures for Royal Challengers Bangalore.

Virat Kohli being in the top of the list comes as no shock, as he is the highest scorer for the Royal Challengers Bangalore Team for the year 2018 and 2019. Being the highest scorer alone does not bag him this position, he also has the high batting average meaning he remains not out most of the time. Thus, he helps in forming good batting partnership with the top order players such as AB de Villers and also the lower middle order players. AB de Villers also is one of the most reliable player for Royal Challengers Bangalore having a strike rate of 174.40. He usually bats in the middle order for creating big scores after initial start by the openers. He form good partnerships with the top order player and the lower order players. C de Grandhomme presence in the list is questionable although he doesn't perform that well with the bat, scoring below average, he is good in saving the wickets thus giving a little pressure on the bowling teams. Washington Sundar is primarily a bowler but his excellent performance in the lower middle order forming excellent partnerships with middle order players as well as lower order players has earned him this chance and also a slot in the playing eleven for Indian national team.

Sunrisers Hyderabad

Player Name	Betweenness Centrality	Player Name	Closeness Centrality
KS Williamson	0.14811	KS Williamson	0.14811
Sandeep Sharma	0.145039	Rashid Khan	0.145039
Rashid Khan	0.108673	YK Pathan	0.108673
YK Pathan	0.103169	MK Pandey	0.103169
MK Pandey	0.100471	DA Warner	0.100471

Table 14. The top five players according to the centrality measures for Sunrisers Hyderabad.

The top player in the centrality ranking is Kane Williamson, which is accurate. He is one of the highest scorers for the team forming crucial partnership in the middle order and the lower order. Having a whooping average of 52.40, he single handedly carried the team to the finals in the year 2018. Sandeep Sharma is an all rounder for the team, and usually bats at the lower order. He is included in the list due to his performance in forming good partnership with players the lower batting order. Rashid Khan is in the list for similar reasons, being a bowler, his explosive batting style has helped in forming crucial partnership among players in lower order. YK Pathan is an excellent middle order player. MK Pandey is a gift to Sunrisers Hyderabad, batting in the top order Pandey has shown excellent batting performance while creating good partnership with players such as Warner, Rashid, Pathan and Williamson. The only player missing in the betweenness list is DA Warner, although being a good performer he is not able to form efficient partnerships, also his performance in the 2020 season of IPL was below par. Overall the team appears to be well balanced.

References

1. Ahmad, Haseeb, et al. "Prediction of rising stars in the game of cricket." *IEEE Access* 5 (2017).
2. Swartz, Tim B. "Research directions in cricket." *Handbook of Statistical Methods and Analysis in Sports, editors JH Albert, ME Glickman, TB Swartz and RH Koning, Chapman & Hall/CRC Handbooks of Modern Statistical Methods: Boca Raton, FL* (2016).
3. Ali, Rahman, Sungyoung Lee, and Tae Choong Chung. "Accurate multi-criteria decision-making methodology for recommending machine learning algorithm." *Expert Systems with Applications* 71 (2017).
4. Petersen, Alexander M., Woo-Sung Jung, and H. Eugene Stanley. "On the distribution of career longevity and the evolution of home-run prowess in professional baseball." *EPL (Europhysics Letters)* 83.5 (2008).
5. Ribeiro, H. V., et al. "Dynamics of tournaments: the soccer case." *The European Physical Journal B* 75.3 (2010).
6. Koster, Jeremy, and Brandy Aven. "The effects of individual status and group performance on network ties among teammates in the National Basketball Association." *PloS one* 13.4 (2018).
7. Amin, Gholam R., and Sujeet Kumar Sharma. "Measuring batting parameters in cricket: A two-stage regression-OWA method." *Measurement* 53 (2014).
8. Bracewell, Paul J., and Katya Ruggiero. "A parametric control chart for monitoring individual batting performances in cricket." *Journal of Quantitative Analysis in Sports* 5.3 (2009).
9. De Silva, Basil M., Greg R. Pond, and Tim B. Swartz. "Applications: Estimation of the Magnitude of Victory in One-day Cricket RMIT University, Mayo Clinic Rochester and Simon Fraser University." *Australian & New Zealand Journal of Statistics* 43.3 (2001).
10. Davis, Jack, Harsha Perera, and Tim B. Swartz. "A simulator for Twenty20 cricket." *Australian & New Zealand Journal of Statistics* 57.1 (2015).
11. Mukherjee, Satyam. "Identifying the greatest team and captain—A complex network approach to cricket matches." *Physica A: Statistical Mechanics and its Applications* 391.23 (2012).
12. Page, Lawrence, et al. *The PageRank citation ranking: Bringing order to the web*. Stanford InfoLab, (1999).
13. Daud, Ali, et al. "Ranking cricket teams." *Information Processing & Management* 51.2 (2015).
14. Hirsch, Jorge E. "An index to quantify an individual's scientific research output." *Proceedings of the National academy of Sciences* 102.46 (2005)

Appendix

1.

```
import pandas as pd
from openpyxl import load_workbook
import xlsxwriter

xl =
pd.ExcelFile("D:\\WINTER_SEMESTER_2020_21\\CSE3021_SOCIAL_AND_INFORMATION_
NETWORKS\\New folder\\archive\\IPL_MATCHES_2008_2020_ANALYZED.xlsx")
df = xl.parse("IPL_MATCHES_2008_2020_ANALYZED")

dict1 = {"Chennai Super Kings" :{}, "Deccan Chargers":{},"Delhi Capitals":{},"Delhi
Daredevils":{},"Gujarat Lions":{},"Kings XI Punjab":{},"Kochi Tuskers Kerala":{},"Kolkata
Knight Riders":{},"Mumbai Indians":{},"Pune Warriors":{},"Rajasthan Royals":{},"Rising Pune
Supergiant":{},"Royal Challengers Bangalore":{},"Sunrisers Hyderabad":{}}
team_to_node={"Chennai Super Kings" :0, "Deccan Chargers":1,"Delhi Capitals":2,"Delhi
Daredevils":3,"Gujarat Lions":4,"Kings XI Punjab":5,"Kochi Tuskers Kerala":6,"Kolkata
Knight Riders":7,"Mumbai Indians":8,"Pune Warriors":9,"Rajasthan Royals":10,"Rising Pune
Supergiant":11,"Royal Challengers Bangalore":12,"Sunrisers Hyderabad":13}
set1 = {"Chennai Super Kings"}
team_names= []
for i in range(0,816):
        set1.add(df['team1'][i])
team_names = sorted(set1)
outWorkbook1 = xlsxwriter.Workbook("EDGES_TEAMS_DATA.xlsx")

outSheet1 = outWorkbook1.add_worksheet()
outSheet1.write("A1","source")
outSheet1.write("B1","target")
outSheet1.write("C1","type")
outSheet1.write("D1","weight")
row=1
for i in team_names:
        for j in team_names:
                num_matches=0
                num_wins_team2=0
                for k in range(0,815):
                        if((df['team1'][k]==i and df['team2'][k]==j) or (df['team1'][k]==j and
df['team2'][k]==i)):
                                num_matches+=1
                                if(df['winner'][k]==j):
```

```
                    num_wins_team2+=1

            if num_matches!=0:
                dict1[i][j]=num_wins_team2/num_matches
                outSheet1.write(row,0,team_to_node[i])
                outSheet1.write(row,1,team_to_node[j])
                outSheet1.write(row,2,"Directed")
                outSheet1.write(row,3,dict1[i][j])
                row+=1

outWorkbook = xlsxwriter.Workbook("NODES_TEAMS_DATA.xlsx")

outSheet = outWorkbook.add_worksheet()

outSheet.write("A1","id")
outSheet.write("B1","label")
for item in range(len(team_names)):
    outSheet.write(item+1,0,item)
    outSheet.write(item+1,1,team_names[item])
outWorkbook.close()
outWorkbook1.close()
```

2.

```
page_rank_init={"Chennai Super Kings" :1/14,"Deccan Chargers":1/14,"Delhi Capitals":1/14,"Delhi Daredevils":1/14,"Gujarat Lions":1/14,"Kings XI Punjab":1/14,"Kochi Tuskers Kerala":1/14,"Kolkata Knight Riders":1/14,"Mumbai Indians":1/14,"Pune Warriors":1/14,"Rajasthan Royals":1/14,"Rising Pune Supergiant":1/14,"Royal Challengers Bangalore":1/14,"Sunrisers Hyderabad":1/14}

page_rank_next={"Chennai Super Kings" :1,"Deccan Chargers":1,"Delhi Capitals":1,"Delhi Daredevils":1,"Gujarat Lions":1,"Kings XI Punjab":1,"Kochi Tuskers Kerala":1,"Kolkata Knight Riders":1,"Mumbai Indians":1,"Pune Warriors":1,"Rajasthan Royals":1,"Rising Pune Supergiant":1,"Royal Challengers Bangalore":1,"Sunrisers Hyderabad":1}

for k in range(0,50):
    for i in team_names:
        team_against=dict1[i].keys()
        for j in team_against:
            if i!=j:
                S_j_out=0
                Sum_rank=0
                team_played=dict1[j].keys()
```

```
                        for l in team_played:
                            if l!=j:

                                S_j_out+=dict1[j][l]
                        Sum_rank=page_rank_init[j]*dict1[i][j]
                        Sum_rank=Sum_rank/S_j_out
                page_rank_next[i]=Sum_rank*(0.85)+0.15/14
        for i in team_names:
                page_rank_init[i]=page_rank_next[i]
sorted_page = sorted(page_rank_next.items(), key=lambda x: x[1],reverse=True)
for i in sorted_page:
        print(i)
```

3.

```
import pandas as pd
from openpyxl import load_workbook
import xlsxwriter
import tabulate
import math
xl =
pd.ExcelFile("D:\\WINTER_SEMESTER_2020_21\\CSE3021_SOCIAL_AND_INFORMATION_
NETWORKS\\New folder\\archive\\IPL_2018_20_UWR_APPROACH.xlsx")
df = xl.parse("IPL_2018_20_UWR_APPROACH")

dict1_wins = {"Chennai Super Kings" :{},"Delhi Capitals":{},"Kings XI Punjab":{},"Kolkata
Knight Riders":{},"Mumbai Indians":{},"Rajasthan Royals":{},"Royal Challengers
Bangalore":{},"Sunrisers Hyderabad":{}}
dict1_t_val = {"Chennai Super Kings" :{},"Delhi Capitals":{},"Kings XI Punjab":{},"Kolkata
Knight Riders":{},"Mumbai Indians":{},"Rajasthan Royals":{},"Royal Challengers
Bangalore":{},"Sunrisers Hyderabad":{}}
team_to_node={"Chennai Super Kings" :0,"Delhi Capitals":1,"Kings XI Punjab":2,"Kolkata
Knight Riders":3,"Mumbai Indians":4,"Rajasthan Royals":5,"Royal Challengers
Bangalore":6,"Sunrisers Hyderabad":7}
set1 = {"Chennai Super Kings"}
team_names= []
for i in range(0,173):
        set1.add(df['team1'][i])
team_names = sorted(set1)
for i in team_names:
        num_wins=0
```

```
        num_wickets=0
        num_runs=0
        sum_wickets=0
        sum_runs=0
        for k in range(0,173):
                if(df['winner'][k]==i):
                        num_wins+=1
                        if(df['result'][k]=="wickets"):
                                num_wickets+=1
                                sum_wickets+=df['result_margin'][k]
                        elif(df['result'][k]=="runs"):
                                num_runs+=1
                                sum_runs+=df['result_margin'][k]
        dict1_wins[i]["wins"]=num_wins
        dict1_wins[i]["by_runs"]=num_runs
        dict1_wins[i]["by_wickets"]=num_wickets
        dict1_wins[i]["sum_runs"]=sum_runs
        dict1_wins[i]["sum_wickets"]=sum_wickets
for i in team_names:
        t_wickets = math.sqrt(dict1_wins[i]['sum_wickets'])/2
        t_runs = math.sqrt(dict1_wins[i]['sum_runs'])/2
        t_val = math.sqrt(t_wickets+t_runs)/2
        dict1_t_val[i]['t_wickets'] = t_wickets
        dict1_t_val[i]['t_runs'] = t_runs
        dict1_t_val[i]['t_val'] = t_val
sorted_t_val=(sorted(dict1_t_val, key=lambda x: dict1_t_val[x]['t_val'],reverse=True))
for i in sorted_t_val:
        print(i,"   ",dict1_t_val[i]['t_wickets']," ",dict1_t_val[i]['t_runs']," 
",dict1_t_val[i]['t_val'])
for i in sorted_t_val:
        print(i,"    ",dict1_wins[i]["wins"],"   ",dict1_wins[i]["sum_runs"]," 
",dict1_wins[i]["sum_wickets"])
```

4.

```
for i in team_names:
        for j in team_names:
                num_losses=0
                num_losses_runs=0
                num_losses_wickets=0
                for k in range(0,173):
```

```
                    if((df['team1'][k]==i and df['team2'][k]==j) or (df['team1'][k]==j and
df['team2'][k]==i)):
                        if(df['winner'][k]==j):
                            num_losses+=1
                            if(df['result'][k]=="wickets"):
                                num_losses_wickets+=df['result_margin'][k]
                            elif(df['result'][k]=="runs"):
                                num_losses_runs+=df['result_margin'][k]
            list1=[num_losses,num_losses_wickets,num_losses_runs]
            if(num_losses!=0):
                weight_outlink =
60*(num_losses)+20*(num_losses_runs)+20*(num_losses_wickets)/(num_losses+num_losses_r
uns+num_losses_wickets)
                list1.append(weight_outlink)
                dict1_weighted_outlinks[i][j] = list1
    print(i,"    ",dict1_weighted_outlinks[i])
    print(" ")

team_uwtr_rank={"Chennai Super Kings" :1/8,"Delhi Capitals":1/8,"Kings XI
Punjab":1/8,"Kolkata Knight Riders":1/8,"Mumbai Indians":1/8,"Rajasthan Royals":1/8,"Royal
Challengers Bangalore":1/8,"Sunrisers Hyderabad":1/8}
team_uwtr_rank_next={"Chennai Super Kings" :1/8,"Delhi Capitals":1/8,"Kings XI
Punjab":1/8,"Kolkata Knight Riders":1/8,"Mumbai Indians":1/8,"Rajasthan Royals":1/8,"Royal
Challengers Bangalore":1/8,"Sunrisers Hyderabad":1/8}

for k in range(0,1000):
    for i in team_names:
        damp_factor=0.15
        sum_wtr=0
        sum_t_val=dict1_t_val[i]['t_val']
        for j in team_names:
            if (i!=j):
                sum_wtr=team_uwtr_rank[j]/dict1_weighted_outlinks[j][i][3]
                sum_t_val=dict1_t_val[j]['t_val']
        team_uwtr_rank_next[i] =  (1-
damp_factor)*(dict1_t_val[i]['t_val'])/len(team_names)*sum_t_val+damp_factor*sum_wtr
    for val in team_names:
        team_uwtr_rank[val] = team_uwtr_rank_next[val]
print("-----------------------------------------------------------------------------------------")
sorted_team_uwtr = sorted(team_uwtr_rank.items(), key=lambda x: x[1],reverse=True)
for i in sorted_team_uwtr:
    print(i)
```

```
import pandas as pd
from openpyxl import load_workbook
import xlsxwriter
import tabulate
import math
import math
xl =
pd.ExcelFile("D:\\WINTER_SEMESTER_2020_21\\CSE3021_SOCIAL_AND_INFORMATION_
NETWORKS\\New folder\\archive\\IPL_BALL_BY_BALL_2018_2020.xlsx")
df = xl.parse("IPL_BALL_2018_2020")

Chennai_Super_Kings=[]
Delhi_Capitals=[]
Kings_XI_Punjab=[]
Kolkata_Knight_Riders=[]
Mumbai_Indians=[]
Rajasthan_Royals=[]
Royal_Challengers_Bangalore=[]
Sunrisers_Hyderabad=[]
set1=set()
for j in range(0,43089):
        if(df['batting_team'][j]=="Mumbai Indians"):
                set1.add(df['batsman'][j])
                set1.add(df['non_striker'][j])
Mumbai_Indians=list(set1)
set1.clear()
for j in range(0,43089):
        if(df['batting_team'][j]=="Delhi Capitals"):
                set1.add(df['batsman'][j])
                set1.add(df['non_striker'][j])
Delhi_Capitals=list(set1)
set1.clear()
for j in range(0,43089):
        if(df['batting_team'][j]=="Kings XI Punjab"):
                set1.add(df['batsman'][j])
                set1.add(df['non_striker'][j])
Kings_XI_Punjab=list(set1)
set1.clear()
for j in range(0,43089):
        if(df['batting_team'][j]=="Kolkata Knight Riders"):
                set1.add(df['batsman'][j])
                set1.add(df['non_striker'][j])
```

```
Kolkata_Knight_Riders=list(set1)
set1.clear()
for j in range(0,43089):
        if(df['batting_team'][j]=="Chennai Super Kings"):
                set1.add(df['batsman'][j])
                set1.add(df['non_striker'][j])
Chennai_Super_Kings=list(set1)
set1.clear()
for j in range(0,43089):
        if(df['batting_team'][j]=="Rajasthan Royals"):
                set1.add(df['batsman'][j])
                set1.add(df['non_striker'][j])
Rajasthan_Royals=list(set1)
set1.clear()
for j in range(0,43089):
        if(df['batting_team'][j]=="Royal Challengers Bangalore"):
                set1.add(df['batsman'][j])
                set1.add(df['non_striker'][j])
Royal_Challengers_Bangalore=list(set1)
set1.clear()
for j in range(0,43089):
        if(df['batting_team'][j]=="Sunrisers Hyderabad"):
                set1.add(df['batsman'][j])
                set1.add(df['non_striker'][j])
Sunrisers_Hyderabad=list(set1)
set1.clear()
Chennai_Super_Kings.sort()
Delhi_Capitals.sort()
Kings_XI_Punjab.sort()
Kolkata_Knight_Riders.sort()
Mumbai_Indians.sort()
Rajasthan_Royals.sort()
Royal_Challengers_Bangalore.sort()
Sunrisers_Hyderabad.sort()
# outWorkbook1 = xlsxwriter.Workbook("CSK_Nodes_Partnership.xlsx")

# outSheet1 = outWorkbook1.add_worksheet()
# outSheet1.write("A1","id")
# outSheet1.write("B1","label")
# row=0
# for i in range(0,len(Chennai_Super_Kings)):
#         outSheet1.write(row,0,i)
```

```
#         outSheet1.write(row,1,Chennai_Super_Kings[i])
#         row+=1
#         print(i,Chennai_Super_Kings[i])
# outWorkbook1.close()
# outWorkbook2 = xlsxwriter.Workbook("CSK_Partnership.xlsx")

# outSheet2 = outWorkbook2.add_worksheet()
# outSheet2.write("A1","node1")
# outSheet2.write("B1","node2")
# outSheet2.write("C1","type")
# outSheet2.write("D1","weight")
# row=0
# for i in range(0,len(Chennai_Super_Kings)):
#         for j in range(0,len(Chennai_Super_Kings)):
#                 partnership_scores=0
#                 for k in range(0,43089):
#                         if(Chennai_Super_Kings[i]!=Chennai_Super_Kings[j]):
#                                 if(df['batting_team'][k]=="Chennai Super Kings"):
#                                         if((df['batsman'][k]==Chennai_Super_Kings[i] and
df['non_striker'][k]== Chennai_Super_Kings[j]) or
(df['batsman'][k]==Chennai_Super_Kings[j] and
df['non_striker'][k]==Chennai_Super_Kings[i])):
#                                                 partnership_scores+=df['total_runs'][k]
#                 if(partnership_scores > 0):
#                         print(i,"-",Chennai_Super_Kings[i],"  ",j,"-",Chennai_Super_Kings[j],"
",partnership_scores)
#                         outSheet2.write(row,0,i)
#                         outSheet2.write(row,1,j)
#                         outSheet2.write(row,2,"Undirected")
#                         outSheet2.write(row,3,partnership_scores)
#                         row+=1
# outWorkbook2.close()
# print("Hello World")

# =============DELHI CAPITALS=========================

# outWorkbook1 = xlsxwriter.Workbook("DC_Nodes_Partnership.xlsx")

# outSheet1 = outWorkbook1.add_worksheet()
# outSheet1.write("A1","id")
# outSheet1.write("B1","label")
# row=0
```

```
# for i in range(0,len(Delhi_Capitals)):
#         outSheet1.write(row,0,i)
#         outSheet1.write(row,1,Delhi_Capitals[i])
#         row+=1
#         print(i,Delhi_Capitals[i])
# outWorkbook1.close()
# outWorkbook2 = xlsxwriter.Workbook("DC_Partnership.xlsx")

# outSheet2 = outWorkbook2.add_worksheet()
# outSheet2.write("A1","node1")
# outSheet2.write("B1","node2")
# outSheet2.write("C1","type")
# outSheet2.write("D1","weight")
# row=1
# for i in range(0,len(Delhi_Capitals)):
#         for j in range(0,len(Delhi_Capitals)):
#                 partnership_scores=0
#                 for k in range(0,43089):
#                         if(Delhi_Capitals[i]!=Delhi_Capitals[j]):
#                                 if(df['batting_team'][k]=="Delhi Capitals"):
#                                         if((df['batsman'][k]==Delhi_Capitals[i] and
df['non_striker'][k]== Delhi_Capitals[j]) or (df['batsman'][k]==Delhi_Capitals[j] and
df['non_striker'][k]==Delhi_Capitals[i])):
#                                                 partnership_scores+=df['total_runs'][k]
#                 if(partnership_scores > 0):
#                         print(i,"-",Delhi_Capitals[i],"  ",j,"-",Delhi_Capitals[j],"
",partnership_scores)
#                         outSheet2.write(row,0,i)
#                         outSheet2.write(row,1,j)
#                         outSheet2.write(row,2,"Undirected")
#                         outSheet2.write(row,3,partnership_scores)
#                         row+=1
# outWorkbook2.close()
# print("Hello World")

# ================================KINGS XI
PUNJAB============================================

# outWorkbook1 = xlsxwriter.Workbook("KXIPun_Nodes_Partnership.xlsx")

# outSheet1 = outWorkbook1.add_worksheet()
# outSheet1.write("A1","id")
```

```
# outSheet1.write("B1","label")
# row=1
# for i in range(0,len(Kings_XI_Punjab)):
#         outSheet1.write(row,0,i)
#         outSheet1.write(row,1,Kings_XI_Punjab[i])
#         row+=1
#         print(i,Kings_XI_Punjab[i])
# outWorkbook1.close()
# outWorkbook2 = xlsxwriter.Workbook("KXIPun_Partnership.xlsx")

# outSheet2 = outWorkbook2.add_worksheet()
# outSheet2.write("A1","node1")
# outSheet2.write("B1","node2")
# outSheet2.write("C1","type")
# outSheet2.write("D1","weight")
# row=1
# for i in range(0,len(Kings_XI_Punjab)):
#         for j in range(0,len(Kings_XI_Punjab)):
#                 partnership_scores=0
#                 for k in range(0,43089):
#                         if(Kings_XI_Punjab[i]!=Kings_XI_Punjab[j]):
#                                 if(df['batting_team'][k]=="Kings XI Punjab"):
#                                         if((df['batsman'][k]==Kings_XI_Punjab[i] and
df['non_striker'][k]== Kings_XI_Punjab[j]) or (df['batsman'][k]==Kings_XI_Punjab[j] and
df['non_striker'][k]==Kings_XI_Punjab[i])):
#                                                 partnership_scores+=df['total_runs'][k]
#                 if(partnership_scores > 0):
#                         print(i,"-",Kings_XI_Punjab[i],"  ",j,"-",Kings_XI_Punjab[j],"
",partnership_scores)
#                         outSheet2.write(row,0,i)
#                         outSheet2.write(row,1,j)
#                         outSheet2.write(row,2,"Undirected")
#                         outSheet2.write(row,3,partnership_scores)
#                         row+=1
# outWorkbook2.close()
# print("Hello World")


# ==========================================Kolkata
Knight Riders=========================================

# outWorkbook1 = xlsxwriter.Workbook("KKR_Nodes_Partnership.xlsx")
```

```
# outSheet1 = outWorkbook1.add_worksheet()
# outSheet1.write("A1","id")
# outSheet1.write("B1","label")
# row=1
# for i in range(0,len(Kolkata_Knight_Riders)):
#         outSheet1.write(row,0,i)
#         outSheet1.write(row,1,Kolkata_Knight_Riders[i])
#         row+=1
#         print(i,Kolkata_Knight_Riders[i])
# outWorkbook1.close()
# outWorkbook2 = xlsxwriter.Workbook("KKR_Partnership.xlsx")

# outSheet2 = outWorkbook2.add_worksheet()
# outSheet2.write("A1","node1")
# outSheet2.write("B1","node2")
# outSheet2.write("C1","type")
# outSheet2.write("D1","weight")
# row=1
# for i in range(0,len(Kolkata_Knight_Riders)):
#         for j in range(0,len(Kolkata_Knight_Riders)):
#                 partnership_scores=0
#                 for k in range(0,43089):
#                         if(Kolkata_Knight_Riders[i]!=Kolkata_Knight_Riders[j]):
#                                 if(df['batting_team'][k]=="Kolkata Knight Riders"):
#                                         if((df['batsman'][k]==Kolkata_Knight_Riders[i] and
df['non_striker'][k]== Kolkata_Knight_Riders[j]) or
(df['batsman'][k]==Kolkata_Knight_Riders[j] and
df['non_striker'][k]==Kolkata_Knight_Riders[i])):
#                                                 partnership_scores+=df['total_runs'][k]
#                 if(partnership_scores > 0):
#                         print(i,"-",Kolkata_Knight_Riders[i],"  ",j,"-",Kolkata_Knight_Riders[j],"
",partnership_scores)
#                         outSheet2.write(row,0,i)
#                         outSheet2.write(row,1,j)
#                         outSheet2.write(row,2,"Undirected")
#                         outSheet2.write(row,3,partnership_scores)
#                         row+=1
# outWorkbook2.close()
# print("Hello World")
```

```
# ==========================================MUMBAI INDIANS==========================================


# outWorkbook1 = xlsxwriter.Workbook("MI_Nodes_Partnership.xlsx")

# outSheet1 = outWorkbook1.add_worksheet()
# outSheet1.write("A1","id")
# outSheet1.write("B1","label")
# row=1
# for i in range(0,len(Mumbai_Indians)):
#         outSheet1.write(row,0,i)
#         outSheet1.write(row,1,Mumbai_Indians[i])
#         row+=1
#         print(i,Mumbai_Indians[i])
# outWorkbook1.close()
# outWorkbook2 = xlsxwriter.Workbook("MI_Partnership.xlsx")

# outSheet2 = outWorkbook2.add_worksheet()
# outSheet2.write("A1","node1")
# outSheet2.write("B1","node2")
# outSheet2.write("C1","type")
# outSheet2.write("D1","weight")
# row=1
# for i in range(0,len(Mumbai_Indians)):
#         for j in range(0,len(Mumbai_Indians)):
#                 partnership_scores=0
#                 for k in range(0,43089):
#                         if(Mumbai_Indians[i]!=Mumbai_Indians[j]):
#                                 if(df['batting_team'][k]=="Mumbai Indians"):
#                                         if((df['batsman'][k]==Mumbai_Indians[i] and
df['non_striker'][k]== Mumbai_Indians[j]) or (df['batsman'][k]==Mumbai_Indians[j] and
df['non_striker'][k]==Mumbai_Indians[i])):
#                                                 partnership_scores+=df['total_runs'][k]
#                 if(partnership_scores > 0):
#                         print(i,"-",Mumbai_Indians[i],"  ",j,"-",Mumbai_Indians[j],"
",partnership_scores)
#                         outSheet2.write(row,0,i)
#                         outSheet2.write(row,1,j)
#                         outSheet2.write(row,2,"Undirected")
#                         outSheet2.write(row,3,partnership_scores)
#                         row+=1
```

```
# outWorkbook2.close()
# print("Hello World")


#
==========RAJASTHAN ROYALS==========

# outWorkbook1 = xlsxwriter.Workbook("RR_Nodes_Partnership.xlsx")

# outSheet1 = outWorkbook1.add_worksheet()
# outSheet1.write("A1","id")
# outSheet1.write("B1","label")
# row=1
# for i in range(0,len(Rajasthan_Royals)):
#         outSheet1.write(row,0,i)
#         outSheet1.write(row,1,Rajasthan_Royals[i])
#         row+=1
#         print(i,Rajasthan_Royals[i])
# outWorkbook1.close()
# outWorkbook2 = xlsxwriter.Workbook("RR_Partnership.xlsx")

# outSheet2 = outWorkbook2.add_worksheet()
# outSheet2.write("A1","node1")
# outSheet2.write("B1","node2")
# outSheet2.write("C1","type")
# outSheet2.write("D1","weight")
# row=1
# for i in range(0,len(Rajasthan_Royals)):
#         for j in range(0,len(Rajasthan_Royals)):
#                 partnership_scores=0
#                 for k in range(0,43089):
#                         if(Rajasthan_Royals[i]!=Rajasthan_Royals[j]):
#                                 if(df['batting_team'][k]=="Rajasthan Royals"):
#                                         if((df['batsman'][k]==Rajasthan_Royals[i] and
df['non_striker'][k]== Rajasthan_Royals[j]) or (df['batsman'][k]==Rajasthan_Royals[j] and
df['non_striker'][k]==Rajasthan_Royals[i])):
#                                                 partnership_scores+=df['total_runs'][k]
#                 if(partnership_scores > 0):
#                         print(i,"-",Rajasthan_Royals[i],"  ",j,"-",Rajasthan_Royals[j],"
",partnership_scores)
#                         outSheet2.write(row,0,i)
```

```
#                       outSheet2.write(row,1,j)
#                       outSheet2.write(row,2,"Undirected")
#                       outSheet2.write(row,3,partnership_scores)
#                       row+=1
# outWorkbook2.close()
# print("Hello World")


# ================================================ROYAL
CHALLENGERS=========================================

# outWorkbook1 = xlsxwriter.Workbook("RCB_Nodes_Partnership.xlsx")

# outSheet1 = outWorkbook1.add_worksheet()
# outSheet1.write("A1","id")
# outSheet1.write("B1","label")
# row=1
# for i in range(0,len(Royal_Challengers_Bangalore)):
#       outSheet1.write(row,0,i)
#       outSheet1.write(row,1,Royal_Challengers_Bangalore[i])
#       row+=1
#       print(i,Royal_Challengers_Bangalore[i])
# outWorkbook1.close()
# outWorkbook2 = xlsxwriter.Workbook("RCB_Partnership.xlsx")

# outSheet2 = outWorkbook2.add_worksheet()
# outSheet2.write("A1","source")
# outSheet2.write("B1","target")
# outSheet2.write("C1","type")
# outSheet2.write("D1","weight")
# row=1
# for i in range(0,len(Royal_Challengers_Bangalore)):
#       for j in range(0,len(Royal_Challengers_Bangalore)):
#               partnership_scores=0
#               for k in range(0,43089):
#                       if(Royal_Challengers_Bangalore[i]!=Royal_Challengers_Bangalore[j]):
#                               if(df['batting_team'][k]=="Royal Challengers Bangalore"):
#                                       if((df['batsman'][k]==Royal_Challengers_Bangalore[i]
and df['non_striker'][k]== Royal_Challengers_Bangalore[j]) or
(df['batsman'][k]==Royal_Challengers_Bangalore[j] and
df['non_striker'][k]==Royal_Challengers_Bangalore[i])):
#                                               partnership_scores+=df['total_runs'][k]
```

```
#                 if(partnership_scores > 0):
#                         print(i,"-",Royal_Challengers_Bangalore[i],"  ",j,"-
",Royal_Challengers_Bangalore[j]," ",partnership_scores)
#                         outSheet2.write(row,0,i)
#                         outSheet2.write(row,1,j)
#                         outSheet2.write(row,2,"Undirected")
#                         outSheet2.write(row,3,partnership_scores)
#                         row+=1
# outWorkbook2.close()
# print("Hello World")



#
============================================================
=SUNRISERS HYDERABAD=========================================

outWorkbook1 = xlsxwriter.Workbook("SRH_Nodes_Partnership.xlsx")

outSheet1 = outWorkbook1.add_worksheet()
outSheet1.write("A1","id")
outSheet1.write("B1","label")
row=1
for i in range(0,len(Sunrisers_Hyderabad)):
        outSheet1.write(row,0,i)
        outSheet1.write(row,1,Sunrisers_Hyderabad[i])
        row+=1
        print(i,Sunrisers_Hyderabad[i])
outWorkbook1.close()
outWorkbook2 = xlsxwriter.Workbook("SRH_Partnership.xlsx")

outSheet2 = outWorkbook2.add_worksheet()
outSheet2.write("A1","source")
outSheet2.write("B1","target")
outSheet2.write("C1","type")
outSheet2.write("D1","weight")
row=1
for i in range(0,len(Sunrisers_Hyderabad)):
        for j in range(0,len(Sunrisers_Hyderabad)):
                partnership_scores=0
                for k in range(0,43089):
                        if(Sunrisers_Hyderabad[i]!=Sunrisers_Hyderabad[j]):
```

```
                        if(df['batting_team'][k]=="Sunrisers Hyderabad"):
                            if((df['batsman'][k]==Sunrisers_Hyderabad[i] and
df['non_striker'][k]== Sunrisers_Hyderabad[j]) or (df['batsman'][k]==Sunrisers_Hyderabad[j]
and df['non_striker'][k]==Sunrisers_Hyderabad[i])):
                                partnership_scores+=df['total_runs'][k]
            if(partnership_scores > 0):
                print(i,"-",Sunrisers_Hyderabad[i],"  ",j,"-",Sunrisers_Hyderabad[j],"
",partnership_scores)
                outSheet2.write(row,0,i)
                outSheet2.write(row,1,j)
                outSheet2.write(row,2,"Undirected")
                outSheet2.write(row,3,partnership_scores)
                row+=1
outWorkbook2.close()
print("Hello World")
```