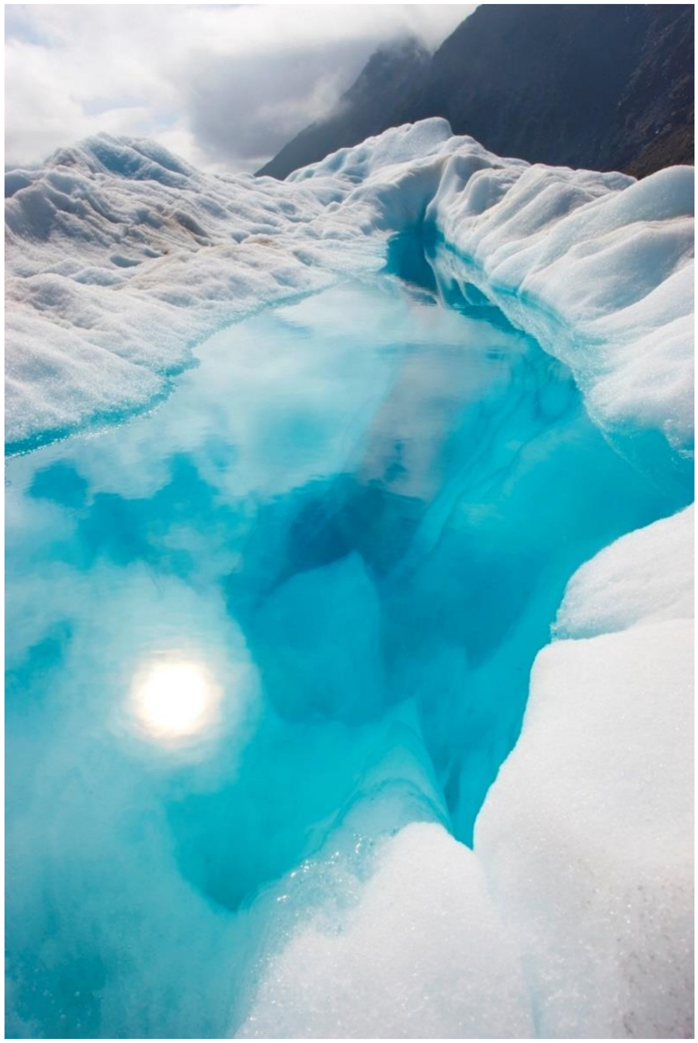

IPL 2016 Match Prediction

Big Data | 11-11-2016

SNo	Name	USN	Class/Section
1	Rakshith Gowda VR	1PI14CS079	CSE H
2	Sunil Pai G	1PI14CS115	CSE H
3	Varun L Pai	1PI14CS126	CSE H
4	Pradyumna Sripad	01FB14ECS152	CSE C

Introduction

This project is to build a prediction model that can calculate the team scores of any IPL 2016 match, given the batting and bowling order. It involves creating a training model using Player Profiles and Player V Player ball data to calibrate the team score.

ALGORITHM/DESIGN

Step I)

- Player V Player data was obtained from cricsheet.org . This data contained Per ball summary of every match in yaml format. We have converted it into CSV files for the ease of reading and manipulating data.
- Player Profiles were scraped from cricketarchive.com. This data consists of two CSV files, one for bowlers and the other for batsmen and the files contain a player's statistics.

```
#webscrape players profiles using BeautifulSoup in python
#website:cricketarchive.com
#algorithm

teams<-list_of_all_the_links_of_each_team_played_in_ipl

for every_team_link in teams:
    players<-urlopen(every_team_link)
    soup<-BeautifulSoup(players)
    allplayers<-soup.find_all_href_to_players

    for each_player_link in allplayers:
        player_profile<-urlopen(each_player_link)
        player_soup<-BeautifulSoup(player_profile)
        profile<-find_player_profile_tables()
        for every_profile_table in profile:
            if(check_t20_format(every_profile_table)):
                player_details<-get_player_details(every_profile_table)
                if(batsman_profile(player_details)):
                    add_entry_into_t20_batsman_profile(player_details)
                else add_entry_into_t20_bowler_profile(player_details)
```

Fig1: algorithm to scrape the data.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Name	Matches	Innings	Not_Outs	Runs	High_Score	Average	No_Of_100	No_Of_50	Strike_Rate	Catches_Taken	Stumpings
2	Aakash Shyamlal Chopra	21	19	1	334	72*	18.55	0	1	91.25	4	NA
3	Aaron James Finch	162	158	15	4945	156	34.58	2	37	133.97	64	NA
4	Aavishkar Madhav Salvi	19	2	2	16	10*	NA	0	0	69.56	5	NA
5	Abhimanyu Mithun	45	22	8	91	12*	6.5	0	0	126.38	15	NA
6	Abhinav Kumar	2	2	1	56	55*	56	0	1	186.66	0	NA
7	Abhinav Mukund	27	26	5	609	67*	29	0	2	111.53	15	NA
8	Abhinav Mukund	27	26	5	609	67*	29	0	2	111.53	15	NA
9	Abhishek Arunkumar Jhunjhunwala	38	32	6	472	54*	18.15	0	2	103.96	20	NA
10	Abhishek Mohan Nayar	88	75	18	1193	79	20.92	0	2	122.35	21	NA

Fig2: batsmen profiles snippet.

Name	Balls	Maidens	Runs	Wickets	Best_Bowling	Average	5_Wicket_Hauls	10_Wicket_Hauls	Strike_Rate	Economy
Aaron James Finch	203	0	318	6	1-9	53	0	0	33.83	9.39
Aavishkar Madhav Salvi	389	0	527	18	3-28	29.27	0	0	21.61	8.12
Abhimanyu Mithun	834	0	1149	41	5-37	28.02	0	1	20.34	8.26
Abhinav Mukund	105	0	101	11	3-22	9.18	0	0	9.54	5.77
Abhinav Mukund	105	0	101	11	3-22	9.18	0	0	9.54	5.77
Abhishek Arunkumar Jhunjhunwala	121	0	172	3	1-5	57.33	0	0	40.33	8.52
Abhishek Mohan Nayar	553	0	770	22	3-13	35	0	0	25.13	8.35
Abhishek Santosh Raut	152	0	215	9	3-22	23.88	0	0	16.88	8.48
Abrar Anjum Kazi	66	0	91	2	2-25	45.5	0	0	33	8.27

Fig3: bowler profiles snippet.

Step II)

- 10 Batsman clusters are created on the Batsman Player Profiles using the Spark MLLib K-Means Clustering function. Clustering is done based on the player's Batting Average and Strike Rate.
- 10 Bowler clusters are created on the Bowler Player Profiles using the Spark MLLib K-Means Clustering function. Clustering is done based on Balls Bowled, Wickets Taken and Economy.

```
val Batvectors = allBatDF.rdd.map(r => Vectors.dense( r.getDouble(1),r.getDouble(2)))
Batvectors.cache()
val BatkMeansModel = KMeans.train(Batvectors, 10, 100)
val Batpredictions = BatrowsRDD.map(r => (r._1, BatkMeansModel.predict(Vectors.dense(r._2,r._3))))
val BatpredDF = Batpredictions.toDF("Batsman_name", "CLUSTER")
```

Fig4: Snippet of the scala code used to cluster the batsmen. Similar code is used for bowlers.

Batsman_name	avgScore	avgStrikeRate	CLUSTER
David Andrew Miller	34.83	137.39	3
Kuldeep Yadav	22	122.22	0
Ravinder Singh Bopara	27.48	119.59	0
Harmeet Singh Bansal	4.4	110	8
Rohan Sunil Gavaskar	16.14	99.12	8
Samuel Badree	12.72	86.95	2
Musavir Asmat Khote	29.4	137.38	3
Sanjay Bapusaheb Bangar	15.24	124.5	4
Vijay Hari Zol	27.09	122.63	0

Fig5: Snippet of clustered batsmen file.

Bowler_name	Balls	Wickets	Economy	CLUSTER
David Andrew Miller	18	0	10.33	3
Kuldeep Yadav	520	37	7.2	1
Ravinder Singh Bopara	3262	165	7.35	4
Harmeet Singh Bansal	594	27	7.8	1
Rohan Sunil Gavaskar	87	3	7.72	3
Samuel Badree	3135	146	5.61	7
Musavir Asmat Khote	132	6	8.5	3
Sanjay Bapusaheb Bangar	578	31	7.35	1
Aiden Craig Blizzard	6	0	10	3

Fig6: Snippet of clustered bowlers file.

Step III)

-Player vs Player Match Data is used to find what run a batsman may score for every ball bowled by a particular bowler. Aggregated Cluster vs Cluster probability is used in case the batsman has not faced the bowler. Algorithm used for achieving this is as given below in the figure.

```
#algorithm to generate player vs player probability
#input ->ipl season 3-8 match data,clustered_player_profiles
#assumption ->if player has not played->probability is zero for every type of score against any bowler

no_of_zero=empty_dict()
no_of_one=empty_dict()
no_of_two=empty_dict()
no_of_three=empty_dict()
no_of_four=empty_dict()
no_of_five=empty_dict()
no_of_six=empty_dict()
no_of_seven=empty_dict()
no_of_outs=empty_dict()
type_of_run=[no_of_zero,no_of_one,no_of_two,no_of_three,no_of_four,no_of_five,no_of_six,no_of_seven,no_of_outs]

all_matches=list_of_all_match_files
for every_match_file in all_matches:
    for every_row in every_match_file:
        batsman_name<-every_row['Batsman']
        bowler_name<-every_row['Bowler']
        run<-every_row['runs']
        type_of_run[run][batsman_name][bowler_name]+=1

for every_batsman in player_profiles:
    for every_bowler in bowler_profiles:
        for every_type_of_score in range(0,9):#8 being out.
            if(check_if_count_of_runs_is_zero(type_of_run,every_type_of_score,every_batsman,every_bowler)):
                names=get_list_of_all_the batsmen_in_the_corresponding_cluster
                for every_name in names:
                    total_count+=get_the_count_of_runs_against_the_same_bowler(type_of_run,every_type_of_score,every_name,every_bowler)

                avg_count=ceil(total_count/names.length)
                type_of_run[every_type_of_score][every_batsman][every_bowler]=avg_count


        probab=compute_probability_of_each_type_of_run.#tuple of probability values
        file_write(every_batsman,every_bowler,probab)
```

Fig7:algorithm to generate player vs player statistics using the cluster information.

This algorithm generates a CSV file for every batsman .These CSV files contain the player probabilities against all the bowlers that played in IPL.An example is shown on the next page.

	A	B	C	D	E	F	G	H	I	J	K
1	Batsman	Bowler	0s_prob	1s_prob	2s_prob	3s_prob	4s_prob	5s_prob	6s_prob	outs_prob	extras_prob
2	MS Dhoni	AJ Finch	0	0	0	0	0.5	0	0.5	0	0
3	MS Dhoni	Aavishkar Madhav Salvi	0	0	0	0	0	0	0	0	0
4	MS Dhoni	A Mithun	0.105263	0.421053	0.052632	0.052632	0.105263	0	0.157895	0.052631579	0.052631579
5	MS Dhoni	A Mukund	0	0	0	0	0	0	0	0	0
6	MS Dhoni	AA Jhunjhunwala	0.2	0.2	0.2	0	0	0	0.2	0.2	0
7	MS Dhoni	AM Nayar	0.125	0.25	0.125	0	0.125	0	0.125	0.125	0.125
8	MS Dhoni	AS Raut	0.333333	0.333333	0	0	0.333333	0	0	0	0
9	MS Dhoni	Abrar Anjum Kazi	0	0	0	0	0	0	0	0	0
10	MS Dhoni	AN Ahmed	0.2	0.1	0.2	0	0.2	0	0.1	0.1	0.1
11	MS Dhoni	AC Voges	0.2	0.4	0	0	0.2	0	0	0	0.2
12	MS Dhoni	AC Gilchrist	0	0	0	0	0	0	0	0	0
13	MS Dhoni	Adam Fraser Milne	0	0	0	0	0	0	0	0	0
14	MS Dhoni	Adam John Hollioake	0	0	0	0	0	0	0	0	0
15	MS Dhoni	A Zampa	0.25	0.25	0	0	0	0	0	0.25	0.25
16	MS Dhoni	AP Dole	0.166667	0.166667	0	0	0.166667	0	0.166667	0.166666667	0.166666667
17	MS Dhoni	AD Mascarenhas	0.25	0.125	0.125	0	0.125	0	0.125	0.125	0.125
18	MS Dhoni	AC Blizzard	0	0	0	0	0	0	0	0	0
19	MS Dhoni	AM Rahane	0	0	0	0	0	0	0	0	0
20	MS Dhoni	AB Agarkar	0.297297	0.324324	0.054054	0	0.162162	0	0.027027	0.027027027	0.108108108
21	MS Dhoni	A Chandila	0.285714	0.142857	0.142857	0	0.142857	0	0.142857	0.142857143	0
22	MS Dhoni	AD Nath	0	0	0	0	0	0	0	0	0

Fig8:Snippet of the player vs player information for MS Dhoni

If the player vs player probability turns out to be zero then it is likely that the batsman has not played against that bowler.In these cases, we use the cluster vs cluster probabilities which looks like the following.

	A	B	C	D	E	F	G	H	I	J	K
1	Batsman_cluster	Bowler_cluster	0s_prob	1s_prob	2s_prob	3s_prob	4s_prob	5s_prob	6s_prob	outs_prob	extras_prob
2	0	0	0.32851	0.386957	0.065777	0.003007	0.102236	0.000376	0.033828	0.04284909	0.036459312
3	0	1	0.343337	0.341908	0.061808	0.00393	0.123616	0.000357	0.028939	0.0375134	0.058592354
4	0	2	0.364488	0.339448	0.056374	0.00308	0.121184	0.000536	0.023433	0.04365292	0.047803964
5	0	3	0.257827	0.434622	0.062615	0.003683	0.103131	0.001842	0.047882	0.03683241	0.051565378
6	0	4	0.367945	0.350022	0.059102	0.004592	0.104281	0.000148	0.025922	0.03969782	0.048289142
7	0	5	0.295826	0.3902	0.065336	0	0.087114	0	0.038113	0.05989111	0.063520871
8	0	6	0.355742	0.346648	0.0551	0.003031	0.110735	0.000357	0.032454	0.04243937	0.053495007
9	0	7	0.363753	0.337514	0.06566	0.00384	0.105721	0.000896	0.027902	0.04185332	0.052860617
10	0	8	0.373278	0.352116	0.058071	0.003445	0.089321	0.000492	0.021161	0.04896654	0.053149606
11	0	9	0.317321	0.376905	0.063741	0.006005	0.110855	0.000462	0.044804	0.03833718	0.041570439
12	1	0	0	0	0	0	0	0	0	0	0
13	1	1	0	0	0	0	0	0	0	1	0
14	1	2	0.666667	0	0.166667	0	0	0	0	0.16666667	0
15	1	3	0	0	0	0	0	0	0	1	0
16	1	4	1	0	0	0	0	0	0	0	0
17	1	5	0	0	0	0	0	0	0	0	0
18	1	6	0.888889	0.111111	0	0	0	0	0	0	0
19	1	7	1	0	0	0	0	0	0	0	0
20	1	8	0.5	0	0	0	0	0	0	0.5	0
21	1	9	0	0	0	0	0	0	0	0	0
22	2	0	0.429603	0.32491	0.046931	0	0.061372	0	0.028881	0.06859206	0.039711191

Fig9:A snippet of the cluster vs cluster probabilities.

Step IV)

-This part involves creating a model to simulate the IPL match and predict the winner based on their scores.

We have made an attempt to create two different models and the difference between the two is in choosing the possible score that for a batsman facing the ball -one model generates random number between 0 and 1 to pick the score according to the cumulative range probabilities that we compute every time , and the other model picks the score which has the highest probability. The rest of both algorithms work in an identical way.

<u>Working</u>:

-Both of our score predicting algorithm are sensitive to the batting and the bowling order of each side which should be fed to the model manually by creating a CSV. Here we explain the working of the algorithms for team 1 i.e team which bats first. The same applies for team 2.

-Once both batting and bowling order of the opposite sides is read, we create a list which contains the bowler name with his wicket taking probability (total number of wickets/number of balls he bowled) and also a list of all the batsmen from team 1 with their scoring probabilities and out's probability against the bowlers of team 2. From this we compute the probability of batsman getting not out.

-If in case the probability of scoring is zero for a batsman that means it is likely that he has not faced the bowler. In this case we use the cluster vs cluster probabilities computed in step 3 which results in a list which looks like < Batsman Name, Bowler Name, 0Run, 0Run, 1Run, 1Run, 2Run, 2Run, 3Run, 3Run, 4Run, 4Run, 5Run, 5Run, 6Run, 6Run, Not Out Probability, Out Probability> and for the bowlers < Bowler Name, Wicket Probability, Wicket Probability >

-One can notice that we are duplicating the probabilities for each type of score. The first one is used to get the cumulative probability and the second one is for resetting and incrementing it. Let the second probability be called as increment constant.

-Now that we have the two lists , we run a loop for 20 overs by setting the Strike and Non-strike batsman to 1 and 2 respectively at the beginning. We use the bowling order specified by the user to choose the bowler for every over and iterate 6 times, once per each ball.

-From the beginning of the over we check for batsman getting out by comparing if bowler's wicket probability is higher than the probability of the strike batsman not getting out. If out, increment wicket by 1, ball by 1, and change Strike to Wicket+2(new batsman number) and also divide the bowler's wicket probability by a factors say 5. If there is no wicket , find out which run is being scored.

-To find which run is scored , In the first model we use the cumulative probabilities which we find by summing over the probabilities of every score. These cumulative probabilities are divided by the total cumulative sum hence resulting numbers in range 0-1.Hence we get a fixed range for every score. Now we generate a random number between 0 and 1 and check in which range this number falls. Choose the score accordingly.

In the second model we choose that run with the highest probability being scored.

- Now that we know which run is scored ,reset its probability to the original i.e increment constant and the probability of all the other runs that were not scored is incremented by their respective increment constant except for Not out probability . Also ,we increase bowler's wicket taking probability by its increment constant which ensures that every run not scored gains an increased probability to score in the next turn.

- Add the run scored to total runs. Increment ball by 1.If run scored was odd, swap strike and non-strike batsman. At the very end check if total wickets are less than 10. If so then break out , else increment over count and continue looping until 20 overs get completed.

-We carry out the same for team 2 .Finally we check the total scores. Whichever team scores highest is declared as the winner.

EXPERIMENTAL RESULTS

Model 1:-

We ran the code for model 1 15 times each on three random matches of IPL 2016 namely Match 1(MI Vs RPS), Match 5(KKR Vs MI) and Match 29(RPS Vs MI).

-For match 1 , out of 15 ,the model predicted that Rising Pune Giants would win 11 times and Mumbai would win 4 times. Actual result of the match -RPS won.

-For match 5, out of 15, the model predicted that Mumbai would win 11 times and kkr would win only 4 times . Actual result of the match -MI won. The predicted scores as well as results are depicted in the image below.

-For match 29 ,out of 15,the model predicted that Mumbai would win 12 times and Pune would win only 4 times. Actual result of the match -MI won.

Looking at the frequency of wins per each time in a match ,from above, we can say that our model works very well in predicting the winner of the game and the scores predicted are also comparable with the original scores.

		KKR Vs MI		Match 5			
	KKR Runs	KKR Wickets		MI Runs	MI Wickets		Winner
	184	10		213	6		MI
	212	10		243	6		MI
	241	10		243	6		MI
	242	10		225	5		KKR
	193	10		229	6		MI
	197	10		236	6		MI
	190	10		239	5		MI
	206	10		204	6		KKR
	198	10		203	6		MI
	188	10		227	6		MI
	177	10		249	6		MI
	176	10		220	6		MI
	215	10		222	6		MI
	221	10		210	6		KKR
	228	10		225	6		KKR
Actual scores/Result	187	5		188	4		MI

Figure 10: Results and scores predicted by model 1 of match KKR vs MI.

Model 2:-

We ran the model 2 code on 8 different matches in IPL 2016.The analysis is depicted in the table.

Match No.	Match	Innings 1	Runs 1	Wickets 1	Innings 2	Runs 2	Wickets 2	Winner	actual Innings 1	actual wickets	actual score	actual wickets	actual winner
1	MI vs RPS	MI	116	9	RPS	117	4	RPS	121	8	126	1	RPS
2	DD vs KKR	DD	78	10	KKR	124	4	KKR	98	10	99	1	KKR
3	KXIP vs GL	KXIP	76	10	GL	125	5	GL	161	6	162	5	GL
4	RCB vs SRH	RCB	94	10	SRH	134	4	SRH	227	4	182	6	RCB
5	KKR vs MI	KKR	100	10	MI	115	6	MI	187	5	188	4	MI
6	RPS vs GL	RPS	95	9	GL	120	2	GL	163	5	164	3	GL
7	KXIP vs DD	KXIP	96	9	DD	157	3	DD	111	9	113	2	DD
29	RPS vs MI	RPS	93	10	MI	146	3	MI	159	5	161	2	MI

From the above table it is evident that Model 2 predicts right 11 times out of the 12 random matches we chose although the scores are comparable only to a certain extent. From this we can conclude that , since model's successes ratio being very high , can be used to predict the outcome of an IPL match in 2016.

FUTURE ENHANCEMENTS

- Creating a user-friendly GUI.
- Using data from every IPL 2016 match predicted played to further enhance the training model to include changes in form and consecutively predicting IPL 2017.
- Creating a model which takes into account of the type of pitch, weather conditions and other external factors that effect a match to predict scores.
- Make the probabilities sensitive to the current "form" of the player to get accurate results.

REFERENCES

- Spark MLlibDocumentation :
  https://spark.apache.org/mllib/
- BeautifulSoup Documentation:
  https://www.crummy.com/software/BeautifulSoup/bs4/doc/
- Player profile data source:
  http://cricketarchive.com/
- IPL match data source:
  http://cricsheet.org/
- Stack Overflow
  http://stackoverflow.com/

EVALUATIONS (Leave this for the faculty)

Date	Evaluator	Comments	Score

CHECKLIST

SNo	Item	Status
1.	Source code documented	complete
2	Source code uploaded to CCBD server	incomplete
3	Recorded video of demo	incomplete
4	Instructions for building and running the code. Your code must be usable out of the box.	complete
5	Dataset used for project uploaded. Please include a description of the dataset format. This includes input file format.	Data set formats: 1)**Match data**: Batsman name(onstrike),ball number,bowler name,runs scored. 2)**Player profile data**: a)Batsmen : name,matches,innings,not outs,runs,high score,average,no_of_100,no_of_50,strike rate,catches,stumping. b)bowler: name,balls,maidens,runs,wickets,best bowling,average, 5 wickets,10 wickets,strike rate,economy.