IPL Analysis

Its prediction time.....

Big Data | Aug 2016 - Dec 2016

SNo	Name	USN	Class/Section
1	Raghavendra G	1PI13CS113	7th B
2	Sharath K P	1PI13CS141	7th B
3	Shreyas G	1PI13CS153	7th B
4	Nagashree A C	1PI13CS201	7th B

Introduction

Big data is an evolving term that describes any voluminous amount of structured, semi structured and unstructured data that has the potential to be mined for information.

Big data is being generated by everything around us at all times. Every digital process and social media exchange produces it. Systems, sensors and mobile devices transmit it. Big data is arriving from multiple sources at an alarming velocity, volume and variety. To extract meaningful value from big data, you need optimal processing power, analytics capabilities and skills. . It is also helping in predicting the weather, elections, cyber-attacks, Olympics and various other things. This has thus revolutionized the field of predictive analysis.

Predictive analytics is the practice of extracting information from existing data sets in order to determine patterns and predict future outcomes and trends. Predictive analytics does not tell you what will happen in the future. It forecasts what might happen in the future with an acceptable level of reliability, and includes what-if scenarios and risk assessment. Predictive analysis also is used a great deal in the field of sports to predict the outcome of crucial matches, to lay bets and sometimes for the enjoyment of fans.

The Indian Premier League is a professional Twenty20 cricket league in India contested during April and May of every year by franchise teams representing Indian cities. IPL is the most-attended cricket league in the world and ranks sixth among all sports leagues.

So we are aiming to build a system which would do a predictive analysis of IPL 2016 using the existing player data sets in the preceding year, 2015. Storing and channelizing that data correctly using the fast emerging techniques of Big data, at the end we aim to predict and simulate a match of IPL 2016 successfully as far as possible.

Related work

To gain an insight into Bigdata, we went through numerous videos, articles by well-established names in the field of big data.

The talk by Peter Norwig mainly focused on whether the advent of Big Data into various prediction analysis fields renders domain knowledge redundant. He highlighted various pitfalls in assuming that domain knowledge is obsolete. This assumption gives rise to various problems .So at the end, he highlights the need for domain knowledge and big data to co-exist side by side.

Nate Silver, a renowned name in the field of Political predictive analysis says that an enormous amount of data is being generated these days. This data helps to get many useful co relations but at the same time it gives rise to spurious correlations. So he stresses the need to spot the "signal among all the noise".

We also read up on clustering algorithms and found out that 'K-means' algorithm served the requirement of our project very well to group the players into different types of clusters. K-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This method produces exactly k different clusters of greatest possible distinction. The best number of clusters k leading to the greatest separation (distance) is not known as a priori and must be computed from the data. The objective of KMeans clustering is to minimize total intracluster variance. Also, it gives the best result when data set are distinct or well separated from each other. KMeans was implement in ML Library of PySpark.

It is fast, robust and easier to understand. It is relatively efficient: O(tknd), where n is # objects, k is # clusters, d is # dimension of each object, and t is # iterations. Normally, k, t, d << n.

We have read about Duckworth Lewis method as it also is based on predicting the outcome of a match. It is generally accepted to be the most accurate method of setting a target score.

ALGORITHM/DESIGN

The first stage of building our predictive system involved collecting Player vs Player data in T20'S played in the year of 2015. ESPN Cricinfo has always maintained a consistent database where in all information is made available to the needs of the cricket fans. They also have a search engine called "StatsGuru" which enables cricket fans to give their own customized input query and get the required results. We collected player vs player data and player's data from Cricinfo.

The player's data was arranged in format,

For Batsmen,

Player, Team, Match, Innings, NO, Runs, HS, Ave, BF, SR, 100, 50, 4s, 6s, Ct, St

For Bowler,

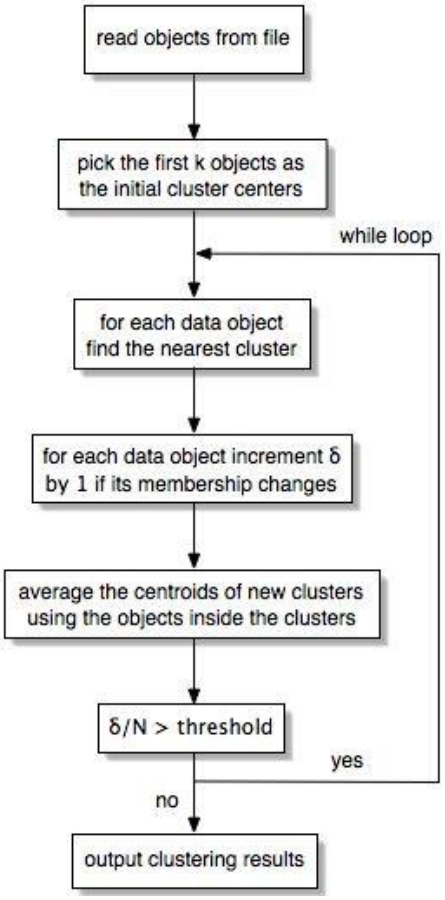
Player, Team, Match, Innings, Balls, Runs, Wickets, BBI, BBM, Average, Economy, SR, 4w, 5w, 10

For player vs player data,

Batsman Name, vs Bowler, 0s, 1s, 2s, 3s, 4s, 5s, 6s, 7+, Dismissal Runs, Balls, SR

Now since we have our data ready, we could begin our prediction stage. But the problem is what would be the factors and data on which the prediction of a batsmen and bowler who haven't encountered each other ever in the career come face to face? So we decided to group similar batsmen and bowlers into the same clusters to overcome this problem. Hence we used K-means algorithm to cluster the players. K-Means finds the best centroids by alternating between assigning data points to clusters based on the current centroids choosing centroids (points which are the center of a cluster) based on the current assignment of data points to clusters. K-means algorithm was implemented in ML-Library i.e. MLlib of PySpark, using which we clustered player data.

K-Means Algorithm:



N: *number of data objects*
K: *number of clusters*

objects[N]: *array of data objects*
clusters[K]: *array of cluster centers*
membership[N]: *array of object memberships*

```
kmeans_clustering( )
1   while δ/N > threshold
2     δ ← 0
3     for i ← 0 to N-1
4       for j ← 0 to K-1
5         distance ← | objects[i] - clusters[j] |
6         if distance < dmin
7           dmin ← distance
8           n ← j
9       if membership[i] ≠ n
10        δ ← δ + 1
11        membership[i] ← n
12      new_clusters[n] ← new_clusters[n] + objects[i]
13      new_cluster_size[n] ← new_cluster_size[n] + 1
14    for j ← 0 to K-1
15      clusters[j][*] ← new_clusters[j][*] / new_cluster_size[j]
16      new_clusters[j][*] ← 0
17      new_cluster_size[j] ← 0
```

We decided to group the players into 10 clusters each after analyzing and experimenting with the input data set. Clustering of batsmen was on the parameters Average, Strike rate and Balls Faced. If we had taken only strike rate, then a tail ender batsman having a high strike rate in just a few innings also would come into the same cluster as an explosive top order batsman and hence would lead to dissimilar players being in same group and thus would lead to a bad prediction and simulation. Hence the parameters together provide a good measure. Similarly, the bowlers we grouped on two parameters namely economy and strike rate. Again, these two parameters taken together serve to get a good clustering. After clustering, we wrote the output into Hive.

Using the cluster data from Hbase, player data and player vs player data in from Hive, we compute cluster vs cluster data consisting of probability of 0's, 1's, 2's, 3's, 4's, 5's, 7's, runs, balls and wicket for each cluster is computed. This cluster vs cluster probability distribution statistics are dumped into HBase. Using this probability distribution it now helps us to

predict the outcome of that delivery by using the technique of generating a random number in a given range and checking the class interval into which it falls.

Now, we are all set to simulate a match prediction. We feed the two teams and its respective batting and bowling orders into our program. The clusters to which the bowler and the batsman belonged to was determined by loading the contents from Hive. Once the clusters are known the outcome of the delivery could be determined by loading and accessing the cluster vs cluster contents from HBase.

The fall of wickets is calculated separately. From statistics, the probability of a batsman getting out is calculated. This probability of getting out is updated after every ball. It is multiplied by the probability of that batsman getting out to that bowler and this value is carried forward for the next iteration. Once this probability of the batsman getting out goes below 0.5 he is declared out and the next batsman comes in.

Like this, the match is simulated using the above model to construct each innings and at the end, the winner is predicted.

EXPERIMENTAL RESULTS

- It was observed that K-Means algorithm ran perfectly and all the batsmen and bowlers were placed in the proper clusters according to the parameters. Eg:-Virat Kohli, Brendon McMcCullum, Suresh Raina and similar big hitters were observed to come into the same cluster.

- The cluster vs cluster data computed could predict the results of a given match correctly. Example: The first match of IPL 2016 between Mumbai and Pune when simulated gave us an accurate result as well as a fair score of each team.

    - Actual Result : MI – 121/ 8, RPS – 126/1
    - Our Simulation: MI – 126/10, RPS – 128/4

  Other matches when simulated, gave the perfect end result and a team score which was very less deviated from the actual score.

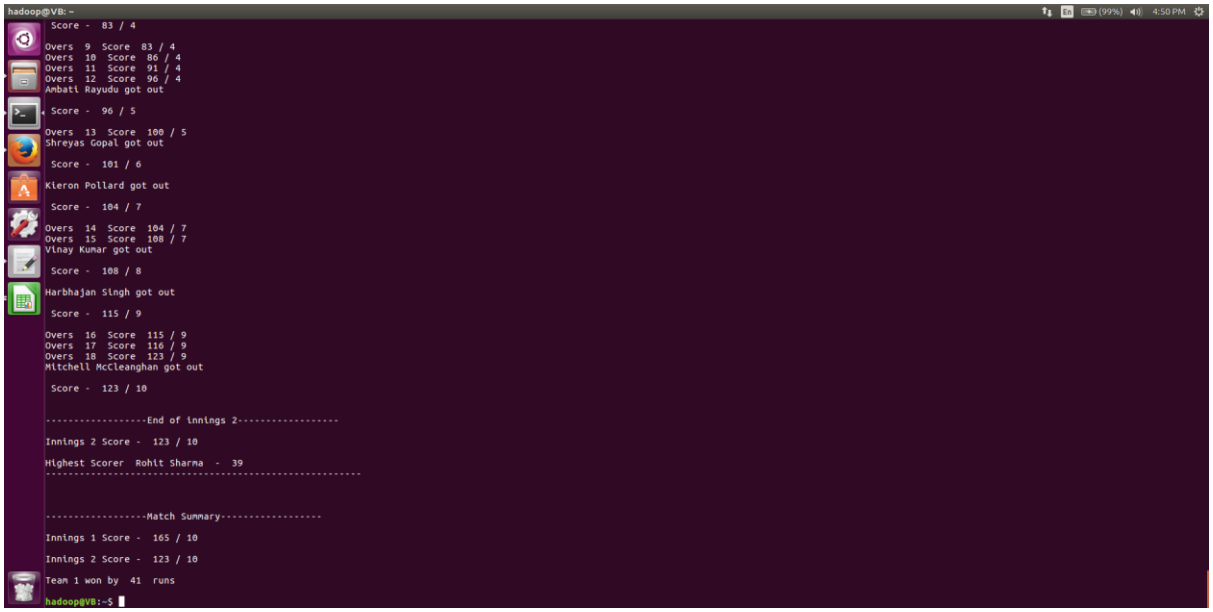
- Cricket is a game of glorious uncertainties and it is also a very multi-dimensional game. Various factors other than the teams order, such as

    - The pitch
    - Fast outfield,
    - Dew factor
    - The weather and clouds overhead

  also play an important role in deciding which way a game of cricket swings too. As these factors were tough to include in our system and not taken into account, we can say that we have achieved a reasonable amount of success.

SIMULATION


hadoop@VB: ~
hadoop@VB:~$ python3 IPL.py
Welcome to IPL
Enter who wins the toss
1 - Team 1
2 - Team 2
1


hadoop@VB: ~
Score - 83 / 4
Overs 9 Score 83 / 4
Overs 10 Score 86 / 4
Overs 11 Score 91 / 4
Overs 12 Score 96 / 4
Ambati Rayudu got out
Score - 96 / 5
Overs 13 Score 100 / 5
Shreyas Gopal got out
Score - 101 / 6
Kieron Pollard got out
Score - 104 / 7
Overs 14 Score 104 / 7
Overs 15 Score 108 / 7
Vinay Kumar got out
Score - 108 / 8
Harbhajan Singh got out
Score - 115 / 9
Overs 16 Score 115 / 9
Overs 17 Score 116 / 9
Overs 18 Score 123 / 9
Mitchell McCleanghan got out
Score - 123 / 10
----------End of innings 2----------
Innings 2 Score - 123 / 10
Highest Scorer Rohit Sharma - 39

----------Match Summary----------
Innings 1 Score - 165 / 10
Innings 2 Score - 123 / 10
Team 1 won by 41 runs
hadoop@VB:~$

FUTURE ENHANCEMENTS

- In order to achieve a higher rate of predicting correctly, the first step is to collect more player vs player data as much as possible.

- As Peter Norwig stressed, domain knowledge also helps a lot in better prediction. So as further enhancement, we can include domain knowledge into our system. Like taking the factor of the nature of pitch into consideration whether it is a green pitch or a dust bowl would obviously lead to a better prediction success rate.

- Based on the boundary size at the grounds, we might have to vary our factors and include some factor to represent the more probability of 6's and 4's.

- We have ignored extras for now. So in future, we have to take that into consideration.

REFERENCES

- K-Means algorithm
  http://stanford.edu/~cpiech/cs221/handouts/kmeans.html

- http://www.espncricinfo.com

- http://spark.apache.org/docs/latest/mllib-clustering.html

- Duck-Worth Lewis System
  https://en.wikipedia.org/wiki/Duckworth%E2%80%93Lewis_method

- http://analyticsindiamag.com/big-data-analytics-a-big-game-changer-in-sports/

EVALUATIONS (Leave this for the faculty)

Date	Evaluator	Comments	Score