SEMINAR REPORT ON

PREDICTIN OF THE RESULT OF A T20 CRICKET MATCH BASED ON TEAM COMPOSITION AND CURRENT MATCH SITUATION

submitted in partial fulfilment of the

requirement for the award of the degree

of

INTEGRATED DUAL DEGREE

in

COMPUTER SCIENCE AND ENGINEERING
(WITH SPECIALIZATION IN INFORMATION TECHNOLOGY)

By:

RONGALI INDRA KUMAR
(10211021)

under guidance of

DR. R. BALASUBRAMANIAN

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

ROORKEE-247667 (INDIA)

CERTIFICATE

This is to certify that the project entitled **–Prediction of the result of a T20 cricket match based on team composition and current match situation** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering**,** submitted in the Department of Computer Science and Engineering of Indian Institute of Technology, Roorkee, India, is an authentic record of his own work carried out during the period from August 2017 to June 2018, under our supervision and guidance.

Date: 18-06-2018

Place: Roorkee

Dr. R. BalaSubramanian

Associate Professor

Dept. of Computer Science and Engineering

Indian Institute of Technology, Roorkee

Roorkee, 247667

ACKNOWLEDGEMENT

I express my sincere gratitude to each and every one who has helped me in completion of this project. First of all, I offer a sincere thanks to Almighty, for always being with me through thick and thin. I also thank my parents who have been constant moral supports throughout. I feel immense pleasure and privilege to express my deep sense of gratitude, indebtedness and thankfulness towards my mentor **Dr. R. BalaSubramanian**, for motivating us with constant appreciation and appraisal.

I would also thank my Institution and entire staff of Central Library, IIT Roorkee who provided me with facilities for various books, research papers and internet.

I shall remain ever grateful to all the persons, who have helped, inspired and encouraged me and above all made me an ever more learned person.

Last but not the least, I would like to convey my hearty thanks to my friends and fellow mates who have directly or indirectly helped me in the compilation of this report.

Rongali Indra Kumar

Dept. of Computer Science and Engineering

IIT Roorkee

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

As the usage of statistical modelling became prevalent in sports field, the most fundamental problem appears to be predicting which of the teams wins the game. In this project, we try to explore a variety of supervised learning methods to forecast results of one of the most popular team games of the present age that is Cricket. We explore methods based on team compositions, team strengths, which are based on individual player's performances in previous games as well as career performance, to predict the results for T20 cricket matches. We also try to select external features that are not dependent of individual players or teams to help achieve more accurate results.

1. INTRODUCTION

With almost 2 billion fans all over the world, cricket is played as a sport in 106 affiliated nations under International Cricket Council (ICC). But, most of the worldwide interest and finance is focused within the 12 full status nations and more significantly in India, Australia and England sometimes referred to as “Big Three” . Legal gambling market in cricket is worth almost 10 billion dollars per year and illegal gambling market is worth almost 50 billion dollars. This range of betting market all over the world gives us incentive to develop advanced prediction models to gain advantage over betting companies and professional gamblers.

Cricket is played across three different formats called Tests, ODIs and T20s. Test matches are played over 5 days, ODI matches are played over a single day and T20 matches are played for 20 overs per side. T20s are a much condensed fromat and fast paced.

Contrary to One Day matches, T20 matches have always had the same basic format and rules since the inception of the T20 format. The signature rules like, the powerplay overs restricting the fielding and the limit on how many overs a bowler can bowl in an innings have contributed to T20 matches being ideal for statistical analysis. As the nature of the matches is also compressed, patterns evolve, which makes this format ideal for computing pattern probabilities by checking number of times the pattern happened in the past and the result at the end. Events like what chance does a team has to defend ten runs in the last over.

In this project we develop prediction models using features that include player personal features based on historical statistics and recent performaces of all the players playing in the match, team strength features and current match situation related features to achieve best possible results.

2. RELATED WORK

Since Twenty20 format of the game is relatively new there is not much research done in this area. And since the betting market is not regulated in the Asian markets, research done is quite low. But we have incentive to develope some good prediction models to not only have an advantage over betting companies but also a lot T20 franchises and even international sides are using machine learning algorithms to predict certain scenarios to gain advantage over the opposition team.

The D/L method[1] to reset targets in matches that were interrupted, was adopted in 1998 by International Cricket Council (ICC) which was actually proposed by Duckworth and Lewis. The graphical representation methods for comparing players are explored in [6], [7], and [8]. The team strengths along with other factors to model performance of players has been done in [9]. In [10], the authors consider the various factors and conditions that affect the game that include advantages to the home teams, effect of day and night fall and even toss etc. they use Bayesian classifiers for predicting the results of the matches.

However in [11], a combination of nearest-neighbor and linear regression algorithms were used for predicting the result of the matches by using both current state of the match and historical data. In [12], the Naïve Bayes Classifiers and Support Vector Machines were used over similar feature sets for predictive modelling.

In this project, we try to explore not only the previously mentioned features but also the changes in team composition over time. The eleven players that play the game are continually replaced over time depending on match conditions, opponent teams, venues, injured players etc. So, completely relying on the historical data is insufficient as it can't portray the team's current competence. Considering such factors might give incorrect results. By considering the above mentioned features we do not get the full picture without the current match situation because once the game starts a lot of things could happen which is why we also look at the models which consider features based on current match situation.

3. METHODOLOGY

In this section, we explain our approach to the problem in detail, including the definitions, dataset, features and the mechanics of various algorithms used to predict the result of a t20 match.

3.1 DATASET

The dataset consists of 150,460 balls from 636 T20 matches played during the season 1 and season 10 of IPL. In this dataset every ball has all the information what happened during that instant(after the ball is delivered). From this dataset, features are constructed that form the input for some of the models we discuss below.

3.2 FEATURES

Features we used in this project are considered based on two scenarios. current_score, balls_remaining, and wickets are features which takes current match situation into account. Venue, toss, $strength_{A/B}$ are features which takes historical statistics of teams and players.

Consider every ball in a match as a unique event. After a particular ball; current_score is the total runs scored up to that instant, balls_remaining is the number of balls left after in an innings and wickets is the number of batsmen who were dismissed. These features are considered to represent the current match situation but not the historical statistics of teams and players.

Now, we consider features which take player statistics into account. Let A and B be the two teams playing in a match m. LetP(T,m)be the complete set of all the players of team T playing match m, and let ∅(p) be the set of the career statistics of player p. The most useful career statistics of a player p (as used by [14], [15], [16] and [17]) are explained in Table 3.1.

Table 3.1: Player Features and their Notations

Notation	Definition
$\emptyset_{Matches\_Played}$	No. of matches player played
$\emptyset_{Batting\_Innings}$	No. of matches in which the player batted
$\emptyset_{Batting\_Average}$	No. of runs scored divided by the no. of times the player got out
$\emptyset_{Num\_Sixties}$	No. of times the player scored $\geq$ 60 runs in a match
$\emptyset_{Num\_Thirties}$	No. of times player scored $\geq$ 30 but less than 60 runs in a match
$\emptyset_{Bowling\_Innings}$	No. of matches in which player bowled
$\emptyset_{Wkts\_Taken}$	No. of wickets taken by player
$\emptyset_{TWkts\_Hauls}$	No. of times player has taken $\geq$ 3 wickets in a match
$\emptyset_{Bowling\_Average}$	No. of runs conceded by the player for each dismissal
$\emptyset_{Bowling\_Economy}$	Average no. of runs given by the player per over bowled

3.3 MODELING BATSMAN

Knowing the batting ability of individual players can significantly contribute to predicting the outcome of a match. A team generally has 6-7 good batsmen among the 11 players. To accurately model a batsman's score, we can look into two kinds of statistics to get the required insights into the player's performance. First, career performance is examined and potency as a contender is determined. Second, we look at his recent match scores and analyze the form he is in. A batsman's form determines the contribution he has made to the team in recentmatches, which in turn reflects his confidence levels.

Algorithm 1 Modeling Batsmen

Input: Players p ∈ {P(A,m) ∪ P(B,m)}, Career Statistics of player p: Ø(p)

Output: Batsmen_Score of all the players: $\emptyset_{Batsman\_Score}$

1: **for** all players p ∈ {P(A,m) ∪ P(B,m)} **do**

2: Ø ← Ø(p)

3: u ← $\sqrt{}$ ($\emptyset_{Bat\_Inngs} \div \emptyset_{Matches\_Played}$)

4: v ← 10 *$\emptyset_{Num\_Sixties}$ + 5 *$\emptyset_{Num\_Thirties}$

5: w ← 0.3 * v + 0.7 *$\emptyset_{Bat\_Avg}$

6: $\emptyset_{Career\_Score}$ ← u * w

7: M ← Last 4 matches played by p

8: $\emptyset_{Recent\_Score}$ ← mean(M^{p}_{Runs})

9: **end for**
10: **for** all players p ∈ {P(A,m) ∪ P(B,m)} **do**
11: $\emptyset_{Career\_Score} \leftarrow \emptyset_{Career\_Score} \div \max(\emptyset_{Career\_Score})$
12: $\emptyset_{Recent\_Score} \leftarrow \emptyset_{Recent\_Score} \div \max(\emptyset_{Recent\_Score})$
13: $\emptyset_{Batsman\_Score} = 0.4 * \emptyset_{Career\_Score} + 0.6 * \emptyset_{Recent\_Score}$
14: **end for**

Algorithm 1 gives the pseudo code to model the batsman performance for a give match. Lines 2-6 give the player's Career Score given his overall career statistics. The variable u in line 3 is the ratio of the matches in which the batsman batted to the total matches he played. It determines whether the player is a specialist batsman or not. Larger values of u indicate the player often batting at the top of the batting order and hence that batsman gets to bat in almost all the matches. On the other hand, smaller values of u convey us that the player bats low in the batting order and the chance that he gets to bat in the next match is also slim. Variable $\emptyset_{Career\_Score}$(line 6) considers all of the career statistics, and hence gives the Career Score of a given batsman. Similarly, lines 7-8 give the Recent Score of a given batsman. Variable M (line 7) takes the most recent matches played by a given player. Variable $\emptyset_{Recent\_Score}$(line 8) gives the Recent Score of a given batsman, which is the average runs scored by a player in his recent games. As the Recent Score and the Career Score of players have different ranges, we normalize them(lines 11-12) so that they lie in a similar range of [0,1]. Finally, variable $\emptyset_{Batsman\_Score}$(line 13) gives the Batsman Score of a player that is the combination of his Recent Score and Career Score.

3.4 MODELING BOWLER

It is true that batsman play key role in cricket, but one cannot under weigh the importance of specialist bowlers in a team. Typically there a set of 4-5 specialist bowlers out of the 11 players. And to estimate the potential of a bowler, we model him examining his previous performances.

Algorithm 2 Modeling Bowlers

Input: Players p ∈ {P(A,m) ∪ P(B,m)}, Career Statistics of player p: ∅(p)

Output: Bowler_Score of all the players: $\emptyset_{Bowler\_Score}$
1: **for** all players p ∈ {P(A,m) ∪ P(B,m)} **do**

2: $\emptyset \leftarrow \emptyset(p)$

3: $u \leftarrow \sqrt{(\emptyset_{Bowl\_Innings} \div \emptyset_{Matches\_Played})}$

4: $v \leftarrow 5 * \emptyset_{TWkts\_Hauls} + \emptyset_{Wkts\_Taken}$

5: $w \leftarrow \emptyset_{Bowl\_Avg} * \emptyset_{Bowl\_Eco}$

6: $\emptyset_{Bowler\_Score} \leftarrow (u*v)/w$

7:**end for**

The pseudo code above (Algorithm 2) shows how to model bowlers for a given match. In line 3, variable u is the ratio of the number of matches inwhich the bowler bowled to the total number of matches he played. It helps us determine whether the player is a full-time specialist bowler or not. A player who often bowls at the top of the bowling order is shown by higher value of u and thus he is chosen to bowl in most matches. But for a part-timer, who doesn't bowl in every match he plays, u values are low and so are his chances to play in the next match. Also notice variables v and w (lines 4-5) which show other statistically significant features of a bowler. Lastly there is Variable $\emptyset_{Bowler\_Score}$(line 6) which takes everything into account, and henceforth signifies the player's bowler score. An interesting observation we can make here is that recent performances of a bowler is not taken into account unlike batsmen. As one does not have match-wise individual performances of every bowler, there is a lack of data and thus we cannot use recent performances in the case of bowler.

3.5 MODELING TEAMS

We understand that the batsmen and the bowlers are the fundamental building blocks of any team. And using the modeled batsmen and bowlers, one can define overall team score with respect to each other. The sum of all the batting scores of a teams' players is defined as batting score of that team. In same fashion, sum of all the bowling scores of a teams' players is defined as bowling score of that team. Variable u from the previous algorithms already takes care of the weighted contribution of individual players to the team score, so we can safely use the scores of all the players as the team score. Below is Algorithm 3 which helps us in finding the relative strength between two teams, say, A and B,competing against each other in any match m. Before anything, we need to normalize the Batsman and the Bowler Scores as they have different ranges. We normalize them to the range of [0,1] (lines 1-4). The

batting and bowling scores of both the teams are then calculated (lines 5-8). The relative strength of team A against team B is captured in the variable S (A/B) (line 9). In the entire process, we are fundamentally using the aspect of the game strategy where the bowlers of a team work against the other teams' batsmen and vice-versa. This is clearly shown in Table 3.2 where certain bowlers and batsmen work against one another.

Algorithm 3 Relative strength between two teams

Input: Players p ∈ {P(A,m) ∪ P(B,m)}, Batsman_Score: $\emptyset^{p}_{Batsman\_Score}$, Bowler_Score: $\emptyset^{p}_{Bowler\_Score}$

Output: Strength of Team A against Team B: $S_{A/B}$

1: **for** all players p ∈ {P(A,m) ∪ P(B,m)} **do**

2: $\emptyset_{Batsman\_Score} \leftarrow \emptyset_{Batsman\_Score} \div \max(\emptyset_{Batsman\_Score})$

3: $\emptyset_{Bowler\_Score} \leftarrow \emptyset_{Bowler\_Score} \div \max(\emptyset_{Bowler\_Score})$

4: **end for**

5: $Bat\_Strength_A \leftarrow \left(\sum_{p \in P(A,m)} \emptyset^{p}_{Batsman\_Score}\right)$

6: $Bowl\_Strength_A \leftarrow \left(\sum_{p \in P(A,m)} \emptyset^{p}_{Bowler\_Score}\right)$

7: $Bat\_Strength_B \leftarrow \left(\sum_{p \in P(B,m)} \emptyset^{p}_{Batsman\_Score}\right)$

8: $Bowl\_Strength_B \leftarrow \left(\sum_{p \in P(B,m)} \emptyset^{p}_{Bowler\_Score}\right)$

9: $S_{A/B} = \left(Bat\_Strength_A \div Bowl\_Strength_B\right) - \left(Bat\_Strength_B \div Bowl\_Strength_A\right)$

Table 3.2: Batsmen against Bowler

	batsman	bowler	No_of_Dismissals
0	CH Gayle	R Ashwin	4
0	V Kohli	A Nehra	6
0	SK Raina	Harbhajan Singh	5
0	AB de Villiers	KH Pandya	4
0	MS Dhoni	Z Khan	7
0	G Gambhir	Z Khan	6
0	RG Sharma	R Vinay Kumar	6
0	RV Uthappa	A Mishra	4
0	S Dhawan	Z Khan	4
0	DA Warner	L Balaji	3

3.6 FEATURE CONSTRUCTION

When it comes to predicting the game (winner/ loser), venue of the match and the outcome of the toss as two important factors alongside the relative strength of one team against the other. Hence we have three features for every match played between team A and B: Toss, Venue, and $Stength_{A/B}$. Where team A gets to bat first, its toss value is 1 or 0 otherwise. When it home ground for a team, the value of Venue becomes 1 for that particular team and 0 if it is home ground for the other team. In case it is none of the above, then we assign a value of 2 for the Venue. We know that Algorithm 3 helps us in calculating the relative strength of team Aagainst team B and $Stength_{A/B}$ is obtained from there. Target is a binary variable which helps us determine the winner of a match and it takes the value of 1 if the winner of the match is team A and 0 when team B wins.Applying machine learning algorithms to these three features, we can predict the winner of a match.

Also note that, as any of the two competing teams can be team A, one has to change all the feature values and the target value accordingly.

4. EXPERIMENTATION AND RESULTS

The dataset and the selected features from the previous sections form input to few different machine learning algorithms.

4.1 MODEL 1

Player and team statistics related features form input to a few different machine learning algorithms.The dataset includes all the IPL matches played between 2008 and 2017. The dataset contains the basic match details including the two competing teams, the outcome of the toss, the date when it was held, the venue and the winner of the match for all the matches. Along with these, the career statistics of the participating players and their performances in every match is also included.Finally, we divided the dataset into two parts, namely, the test data and the training data. The training dataset contains all the matches played during the years 2008 to 2016, and the test dataset contains all the matches played in the year 2017.

4.1.1 Learning Weights:

To assign the weights to various features in the Algorithms1 and 2, we have used season 4 of IPL. A series of consecutive matches was deliberately chosen to study the impact of the recent scores of a batsman on his upcoming performances. The estimated scores of the players are compared against their actual performances.After exhaustive experimentation, the final weights are chosen such that the top 6 performing batsmen and bowlers (in terms of runs scored and wickets taken respectively) from both the teams match with the top 6 batsmen and bowlers estimated by our algorithms.

4.1.2 Binary Classifiers:

Using various binary and numeric features and the outcome of the match as the label, we evaluated a large number of binary classifiers using their scikit-learn implementations [18] to generate supervised classification models, including SVM, Random Forests, Logistic Regression, Decision Treesand kNN. To experiment with all the possible values and combinations of the parameters for all the algorithms we used the sweep feature. The efficacy of the kNN algorithm, with k=4, was statistically superior to those obtained by the best models of other classifiers, as shown in Figure 4.1.

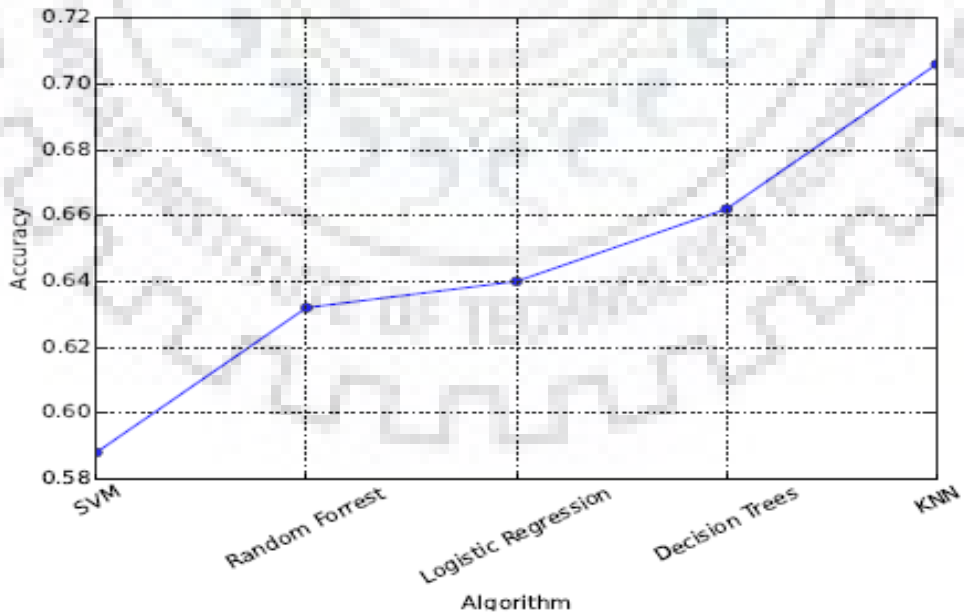

Fig. 4.1: Accuracy of different algorithms used

On the other hand, we compared our model with two other baseline models :- the team winning the toss is the match winner(Model a) as shown in Figure 4.2, and as calculated in algorithm 3, the team with positive relative strength is the match winner(Model b).

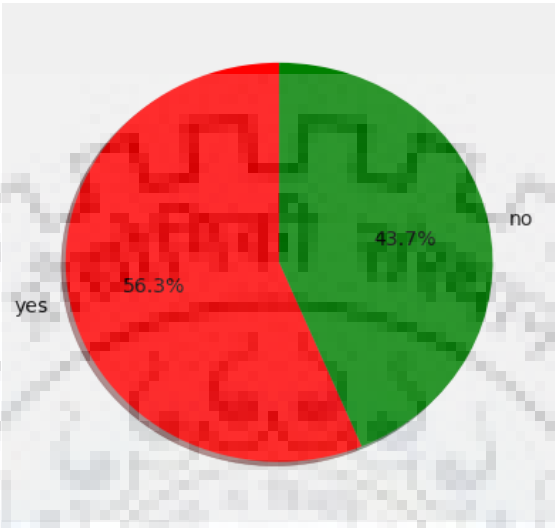

Fig. 4.2: Win percentage for teams which won the Toss

The results are shown in Table 4.1. The dominance of our model compared to the other models justifies the importance of the variousfeatures used.

Table 4.1: Comparing our kNN based model with other baseline models

Model	Accuracy
Model a	0.56
Model b	0.63
Our Model	0.71

4.2 MODEL 2

Match situation related features form input to a few different machine learning algorithms. Prediction of the final score in the 2nd innings (there by winner) is a problem of regression, because it is a continuous variable output. We take that the 150,460 balls in the 2nd innings to be independent of one another.

4.2.1 Linear Regression:

This is the first algorithm we take a look at. This algorithm tries to draw a line along the multi-dimensional data so that the sum of the distances from the line and all the data points is as low as it can be. This is basically the best fit line in two dimensions.

In Python, by using sickit-learn's package of machine learning algorithms, we could model this algorithm simply as shown below.

```
from sklearn.linear_model import LinearRegression

#df1 is 1st innings ball-by-ball data
X = df1[['current_score', 'balls_remaining', 'wickets']]
y = df1.final_score

lin = LinearRegression()
lin.fit(X, y)
```

After fitted, the co-efficient of the linear regression equation is calculated by the algorithm and the equation is:

final_score = 1.158*balls_remaining – 4.037*wickets + 1.083*current_score + 16.2

This algorithm has 0.547 as the value of R^2. This algorithm gives us the average score at the start of the second innings is almost 155 (1.158*120 + 16.2). As the match builds up, we could provide more relevant information about the wickets and current score to output a much better guess. This algorithm also shows that each wicket would save almost four runs for the team which is bowling. It's also not self-consistent because in extreme cases it could output predictions which are lesser than current_score.

4.2.2 KNeighborsRegressor:

This is the second algorithm we used to predict the win. If we take an instance, this algorithm looks for a predetermined number of almost alike instances and the average of the final scores from these instances gives the final_score. Implementation in Python is shown below.

```
from sklearn.neighbors import KNeighborsRegressor

knr = KNeighborsRegressor(n_neighbors=26)
knr.fit(X, y)
```

We get the smallest error when the number of neighbours was 26. Less than 26 is a small sample size whereas the neighbours begin to become a bit too unlike if greater than 26. This algorithm has 0.580 as the value of R^2, which is to some extent improved than the value we got in case of linear regression. However we do not have an equation we can interpret. We simply provide the ball details and the algorithm predicts an output.

4.2.3 RadiusNeighborsRegressor:

Now we look at a similar algorithm to the previous one. This algorithm looks for all the neighbours that are inside a specified radius rather than searching for a predetermined number of nearby neighbours.

```
from sklearn.neighbors import RadiusNeighborsRegressor

rnr = RadiusNeighborsRegressor(radius=1, weights='distance')
rnr.fit(X, y)
```

To provide some outlook, the most occurred distinctive score after completion of over 6 is 50 runs for two wickets after over 7, which happened 59 times. So, we can be convincingly confident that prediction is good if we consider the average final score from these 59 instances. If we relax the necessities to permit any of the overs, current_score and wickets to vary by maximum value of 1 (chosen value of radius), e.g. 51 runs for one wicket from over 7.1, we have 395 alike instances. Indeed, out of the 53,573 distinctive combinations of

wickets, balls_remaining and current_score, 41,475 have ten close neighbours at least. This algorithm is somewhat superior with R^2 value of 0.607. In some extreme cases we cannot predict with much accuracy. A score similar to 191 runs for 3 wickets after 20 balls remaining has only four close neighbours. This algorithm will take the average final score of these four instances, whereas *KNeighboursRegressor* would look for 26 somewhat unalike neighbours. In this case, error margins will be probably high for both of these algorithms.

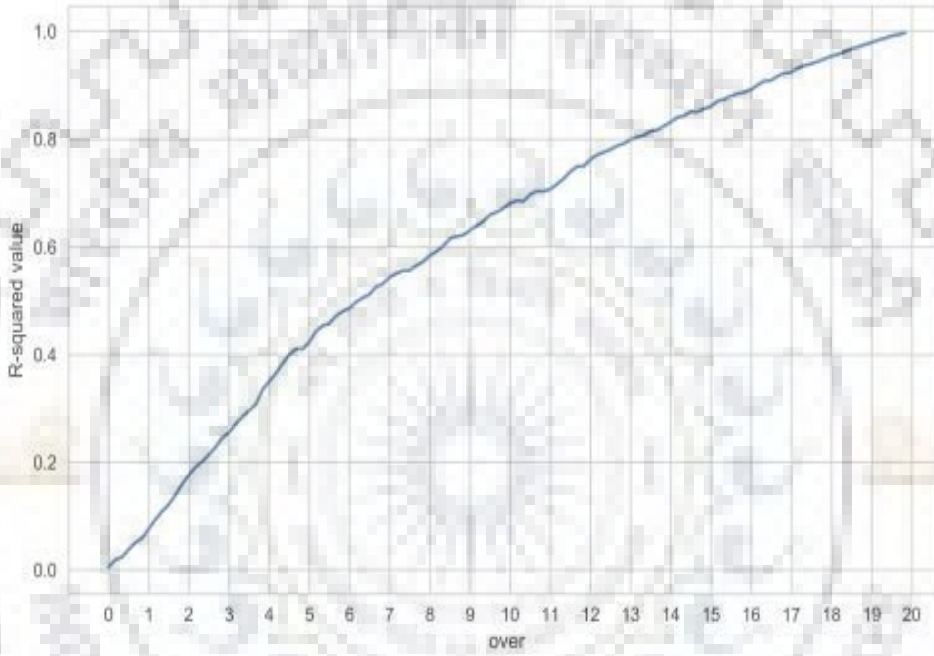

Fig. 4.3: Confidence of *RadiusNeighborsRegressor*

Using the *RadiusNeighborsRegressor* algorithm, as the innings progresses our predictions become more confident as shown in Figure 4.3. To support final predictions, we don't have a lot of information at the begining. The finest we could do is providing final score average based on historical data. Just about the 13th over this algorithm becomes 80% confident and at around two overs to finish it becomes 95% confident. In the final couple of overs a lot of things can occur, but any differences have a tendency to even out over a huge number of matches.

4.2.4 RandomForestRegressor:

This is final algorithm we implemented. This is a type of an assembly model which considers a large number of somewhat dissimilar prediction models. These models are combined together in such a manner so that the performance of this combination on the whole is superior than every individual models' performance.

```
from sklearn.ensemble import RandomForestRegressor,

rfr = RandomForestRegressor(n_estimators=1000, max_features=None)
rfr.fit(X, y)
```

This algorithm is a combination of thousand models, particularly decision trees, and takes all the features while searching for a better split. This algorithm has 0.603 as the value of R^2 which is close to *RadiusNeighborsRegressor* model. Nonetheless, the advantage of this algorithm is that it gives us information regarding the importance of a feature i.e. out of these three features which one is mainly important in prediction of final result.

Table 4.2: Feature Importance

Feature	Importance
current_score	0.495
balls_remaining	0.287
wickets	0.238

In the second innings, the better predictor of final result is current_score as shown in the Table 4.2. It's relevance is almost close to both the balls_remaining and wickets values added. The above observation makes perfect sense because predicting the final score using only the balls remaining and number of wickets. This also might confirm the long debated opinion that wickets in hand are overrated. We usually hear commentators say that a team score of 170-3 from 20 overs could have scored a few more runs with only three wickets down.

5. FUTURE WORK PLAN

In this paper we discussed about predicting the winner of an IPL T20 match using features based on player and team statistics for model 1 and features based on current match situation for model 2. Both the models gave similar results with decent accuracy. In the future we would like to consider features based on both player & team statistics and current match situation for a single model to predict the final score and match winner while the innings is in progress..

6. REFERENCE LIST

1. Duckworth, Frank C., and Anthony J. Lewis. "A fair method for resetting the target in interrupted one-day cricket matches." Journal of the Operational Research Society 49.3 (1998): 220-227.

2. Beaudoin, David, and Tim B. Swartz. "The best batsmen and bowlers in one-day cricket." South African Statistical Journal 37.2 (2003): 203.

3. Lewis, A. J. "Towards fairer measures of player performance in one-day cricket." Journal of the Operational Research Society 56.7 (2005): 804-815.

4. Swartz, Tim B., Paramjit S. Gill, and David Beaudoin. "Optimal batting orders in one-day cricket." Computers and operations research 33.7 (2006): 1939-1950.

5. Norman, John M., and Stephen R. Clarke. "Optimal batting orders in cricket." Journal of the Operational Research Society 61.6 (2010): 980-986.

6. Kimber, Alan. "A graphical display for comparing bowlers in cricket." Teaching Statistics 15.3 (1993): 84-86.

7. Barr, G. D. I., and B. S. Kantor. "A criterion for comparing and selecting batsmen in limited overs cricket." Journal of the Operational Research Society 55.12 (2004): 1266-1274.

8. Van Staden, Paul Jacobus. "Comparison of cricketers bowling and batting performances using graphical displays." (2009).

9. Lemmer, Hermanus H. "The allocation of weights in the calculation of batting and bowling performance measures." South African Journal for Research in Sport, Physical Education and Recreation (SAJRSPER) 29.2 (2007).

10. Kaluarachchi, Amal, and S. Varde Aparna. "CricAI: A classification based tool to predict the outcome in ODI cricket." 2010 Fifth International Conference on Information and Automation for Sustainability. IEEE, 2010.

11. Sankaranarayanan, Vignesh Veppur, Junaed Sattar, and Laks VS Lakshmanan. "Auto-play: A Data Mining Approach to ODI Cricket Simulation and Prediction."SDM. 2014.

12. Khan, Mehvish, and Riddhi Shah. "Role of External Factors on Outcome of a One Day International Cricket (ODI) Match and Predictive Analysis."

13. ESPN Cricinfo, http://www.espncricinfo.com

14. Barr, G. D. I., and R. van den Honert. "Evaluating batsman's scores in test cricket." South African Statistical Journal 32.2 (1998): 169-183.

15. Croucher, J. S. "Player ratings in one-day cricket." Proceedings of the fifth Australian conference on mathematics and computers in sport. Sydney, NSW: Sydney University of Technology, 2000.

16. Lemmer, Hermanus H. "The combined bowling rate as a measure of bowling performance in cricket." South African Journal for Research in Sport, Physical Education and Recreation 24.2 (2002): 37-44.

17. Barr, G. D. I., C. G. Holdsworth, and B. S. Kantor. "Evaluating performances at the 2007 cricket world cup." South African Statistical Journal 42.2 (2008): 125.

18. Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." Journal of Machine Learning Research 12.Oct (2011): 2825-2830.