Auto-play: A Data Mining Approach to ODI Cricket Simulation and Prediction

Vignesh Veppur Sankaranarayanan, Junaed Sattar and Laks V. S. Lakshmanan
Department of Computer Science
University of British Columbia
Vancouver, B.C. Canada V6T 1Z4
Email: {vsvicky,junaed,laks}@cs.ubc.ca

Abstract

Cricket is a popular sport played by 16 countries, is the second most watched sport in the world after soccer, and enjoys a multi-million dollar industry. There is tremendous interest in simulating cricket and more importantly in predicting the outcome of games, particularly in their one-day international format. The complex rules governing the game, along with the numerous natural parameters affecting the outcome of a cricket match present significant challenges for accurate prediction. Multiple diverse parameters, including but not limited to cricketing skills and performances, match venues and even weather conditions can significantly affect the outcome of a game. The sheer number of parameters, along with their interdependence and variance create a non-trivial challenge to create an accurate quantitative model of a game Unlike other sports such as basketball and baseball which are well researched from a sports analytics perspective, for cricket, these tasks have yet to be investigated in depth. In this paper, we build a prediction system that takes in historical match data as well as the instantaneous state of a match, and predicts future match events culminating in a victory or loss. We model the game using a subset of match parameters, using a combination of linear regression and nearest-neighbor clustering algorithms. We describe our model and algorithms and finally present quantitative results, demonstrating the performance of our algorithms in predicting the number of runs scored, one of the most important determinants of match outcome.

Keywords

Sports prediction, analytics, ridge regression, attribute bagging, nearest neighbors

1 Introduction

Primarily played in the member countries of the Commonwealth, cricket has grown in following across all continents. It has the second largest viewership by population for any sport, next only to soccer, and generates an extremely passionate following among the supporters. There is huge commercial interest in strategic planning for ensuring victory and in *game outcome prediction*. This has motivated thorough and methodical analysis of individual and team performance, as well as prediction of future games, across all formats of the game.

Currently, team strategists rely on a combination of personal experience, team constitution and seat of the pants "cricketing sense" for making instantaneous strategic decisions. Inherently, the methodology employed by human experts is to extract and leverage important information from both past and current game statistics. However, to our knowledge, the underlying science behind this has not been clearly articulated. *One of the key problems that needs to be solved in formulating strategies is predicting the outcome of a game.* Our focus in this paper is to address the problem of accurately modeling game progression towards match outcome prediction. We learn a model for one-day format games by mining existing game data. In principle, our approach is applicable towards modeling any format of the game; however, we choose to focus our testing and evaluation on the most popular format, namely one-day international (ODI). By using a combination of supervised and unsupervised learning algorithms, our approach learns a number of features from a one-day cricket dataset which consists of complete records of all games played in a 19-month period between January 2011 and July 2012. Along with these learned *historical features* of the game, our model also incorporates *instantaneous match state* data, such as runs scored, wickets lost etc., as game progresses, to predict future states of an on-going match. By using a weighted combination of both historical and instantaneous features, our approach is thus able to simulate and predict game progression before and during a match. We motivate the problem of game modeling and outcome prediction

in Section 2. Along with a brief introduction to cricket, Section 3 presents the problem formulation, with details on feature modeling. In Section 4, we present our algorithm for predicting the game progression and outcome, with results discussed in Section 5

2 Related Work

2.1 Data Mining in Other Sports The problem of match outcome prediction has been studied extensively in the context of basketball and soccer. Bhandari et al. [4] developed the Advanced Scout system for discovering interesting patterns from basketball games, which has is now used by the NBA teams. More recently, Schultz [12] studies how to determine types and combination of players most relevant to winning matches. In soccer, Luckner et al. [11] predict the outcome of FIFA World Cup 2006 matches using live Prediction Markets. In baseball, Gartheepan et al. [7] built a data driven model that helps in deciding when to 'pull a starting pitcher'. These works are developed with a sport specific intuition which would render them inapplicable to the sport of cricket.

2.2 Academic Interest in Cricket One of the earliest and pioneering works in cricket was by Duckworth and Lewis [6] where they introduce the Duckworth-Lewis or D-L method, which allows fair adjustment of scores in proportion to the time lost due to match interruption (often due to adverse weather conditions such as rain, poor visibility etc.). This proposal has been adopted by the International Cricket Council (ICC) as a means to reset targets in matches where time is lost due to match interruptions. The method proposed in [6], and subsequently adapted by [14], for capturing the resources of a team during the progression of a match has found independent use in subsequent work in cricket modeling and mining [14][2].

Lewis [10], Lemmer [9], Alsopp and Clarke [1], and Beaudoin [3] develop new performance measures to rate teams and to find the most valuable players. Raj and Padma [15] analyze the Indian cricket team's One-Day International (ODI) match data and mine association rules from a set of features, namely toss, home or away game, batting first or second and game outcome. Kaluarachchi and Varde [8] employ both association rules and naive Bayes classifier and analyze the factors contributing to a win, also taking day/day-night game into account. Both approaches use a very limited subset of high-level features to analyze the factors contributing to victory. Furthermore, they do not address score prediction, nor the progression of the game.

Bailey and Clarke [2] use historical match data and predict the total score of an innings using linear regression. As data of a match in progress streams in, the prediction model is updated. Using this, they analyze betting[1] market's sensitivity to the ups and downs of the game. Swartz et al. [17] use Markov Chain Monte Carlo methods to simulate ball by ball outcome of a match using a Bayesian Latent variable model. Based on the features of current batsman, bowler, and game situation (number of wickets lost and number of balls bowled), they estimate the outcome of the next ball. This model suffers from severe sparsity as noted by the authors themselves: the likelihood of a given batsman having previously faced a given bowler in previous games in the dataset is low.

While both [17] and [2] have built match simulators for ODI cricket, their models rely on games played over 10 years ago. ODI cricket has since undergone a number of major rule modifications. Important examples include powerplays, free hit after an illegal ball delivery, and the use of two new balls (as opposed to just one) in an innings. These changes significantly affect the team strategies, and essentially render old models a poor fit. *Our focus is on the modern and current form of ODI cricket, incorporating all recent changes to the game with support for accommodating future rule modifications.*

3 Game Modeling

3.1 Overview of ODI Cricket Rules and Objectives We provide a brief overview of ODI cricket and review its basic rules as they pertain to game modeling and score prediction. We also introduce several basic notations and terminologies used in the rest of the paper.

Toss: Similar to a number of other sports, an ODI cricket match starts with a toss. The team that wins the toss can choose to bat first or can ask the opponents to bat first. This decision is important and takes into account the nature of the playing field, weather conditions, and relative strengths and weaknesses of the two teams.

Objective: In a game between $Team_A$ and $Team_B$, suppose $Team_A$ wins the toss and chooses to bat first. The period during which $Team_A$ bats is called $innings_1$, in which $Team_A$ has 50 *overs* to score as many *runs* as possible, while $Team_B$ tries to minimize the scoring by getting $Team_A$'s batsmen *out* (more commonly referred to as taking *wickets*). Scoring can also be restricted by $Team_B$, by bowling balls that are difficult to score off and by flawless fielding, where *fielders* stop hits by batsmen of $Team_A$ to deny them

[1] There is a vibrant betting market associated with cricket. See, *e.g.*, http://www.betfair.com/exchange/en-gb/cricket-4/sp/.

opportunities to score runs. $Innings_1$ comes to an end when $Team_A$ loses all its wickets or finishes its quota of 50 overs, whichever happens first. Let $Score_A$ denote the number of runs accumulated by $Team_A$ at this point.When $Team_B$ comes in to bat in $innings_2$, it has the exact same number of 50 overs to play, with the goal of scoring at least $Score_A$+1 runs; $innings_2$ ends when $Score_B$, the number of runs scored by $Team_B$, exceeds $Score_A$, or when $Team_B$ finishes its quota of 50 overs or loses all its wickets, whichever happens first. $Team_B$ is deemed the winner in the first case, and $Team_A$ wins otherwise. A third possibility is a *tie* when $Score_A$ and $Score_B$ are equal at the end of the game.[2]

Scoring: Teams can accumulate runs in two ways. One way of scoring is to power-hit the ball outside the playing area. *Four* runs are awarded if the ball touches the ground before rolling past the boundary of the playing area. If the ball lands directly outside the playing area, *six* runs are awarded. Borrowing a term from baseball, game, for convenience, we collectively term runs scored this way as *home runs*. Home runs yield greater reward in terms of runs scored, but the batsmen have to take risks to hit them, which increases their chance of getting out. The other way of scoring is to hit the ball within the playing area and for the two batsmen to run and exchange their positions. In the mean time, the opponent players try to collect the ball to minimize the number of exchanges. Runs are awarded based on the number of times the batsmen exchange their positions before the ball is returned to one of the positions. There is theoretically no bound on the number of exchanges possible in a given ball but this value typically lies in the range *1–3* runs. This way of scoring has a lower risk of the batsman getting out but yields a lower number of runs. We term these *non-home runs*. Runs are awarded to the batting team when the bowler commits a *foul* while delivering the ball. Runs conceded this way are usually small and are accounted for by *non-home runs* in our model.

Dismissal: There are eleven ways for a batsman to lose his *wicket*, commonly referred to as getting out or dismissed. The common ways to get dismissed are being bowled, caught by opponents, run out and Leg-Before-Wicket (abbreviated as LBW). In our model, we do not distinguish between the different forms of dismissal.

Target score: The number of runs accumulated by $Team_A$ at the end of $innings_1$ is $Score_A$. $Score_A$+1 run is set as the *Target* that the team batting second tries to achieve or exceed in $innings_2$.

Resources: Overs and Wickets are collectively termed as *resource*. The batting team consumes the overs to accumulate runs and loses wickets in the process. A batting team has 50 overs and 10 wickets at their disposal at the start of an innings. This resource continually decreases as the game progresses.

Segment: The batting period of a team is called an *innings* and it lasts till they run out of one of the *resources*. We split the 50-over window into 10 segments of 5 overs each, denoted S_i, $1 \leq i \leq 10$. For a team T, R_i^T and W_i^T denote the the number of runs scored and the number of wickets lost in segment S_i, respectively. The total number of runs scored by team T at the end of their innings is given by $R_{eoi}^T = \sum_{i=1}^{10} R_i^T$. We drop the superscript T when the team is clear from the context. Below, we formalize the problem addressed in this paper.

3.2 Problem Formulation The main problem we tackle in this paper is given the *instantaneous* match data up to a certain point in the game, predict the progression of the remainder of the game, and in particular, predict the winner. Before we formalize this, we define a *match state* at segment n, $0 \leq n < 10$, as the pair of numbers consisting of the number of runs scored and the number of wickets lost so far, by the batting team. Notice that given a match state, the resources remaining at the batting team's disposal can be easily calculated: the number of balls remaining is $(10-n) \times 5 \times 6$ and the number of wickets remaining is $10 - (\#\text{wickets lost so far})$.

More precisely, given a match state associated with segment n, namely $(R_{known} = \sum_{i=1}^{n} R_i, W_{known} = \sum_{i=1}^{n} W_i)$, predict the number of runs $\hat{R}_i$ for the remaining segments i, $n+1 \leq i \leq 10$. Using these predictions, the total predicted score at the end of the innings can be obtained as

$$\hat{R}_{eoi} = R_{known} + \sum_{i=n+1}^{n} \hat{R}_i \tag{3.1}$$

If an innings has not commenced, as a special case, $n = 0$, $R_{known} = 0$ and $W_{known} = 0$.

We follow this segmented prediction approach to predict $\hat{R}_{eoi}$ for both $innings_1$ and $innings_2$. $Team_A$ is predicted to be the winner if $\hat{R}_{eoi}^A > \hat{R}_{eoi}^B$. $Team_B$ is predicted to be the winner if $\hat{R}_{eoi}^A < \hat{R}_{eoi}^B$.

3.3 Sub-Problem We break down the problem of predicting the number of runs in the next segment S_{n+1}, given the match state up to segment S_n, into two subproblems, by recognizing that home runs and non-home runs are strategized and scored by different means by the batsmen. We have found from our analysis and exploration that the number of runs can be predicted more accurately if we learn separate models

[2]Currently, there are no tie-breakers in ODI the format, possibly because ties are extremely rare.

for predicting the home runs HR_{n+1} and non-home runs NHR_{n+1}. More precisely, for any segment i, $R_i = HR_i + NHR_i$ and $\hat{R}_i = \hat{HR}_i + \hat{NHR}_i$, where $\hat{X}$ is the predicted value of X.

While it may seem counter-intuitive to use two different classes of techniques to predict the overall total score, this decision was driven by observing the inherent nature of the game itself, and has eventually been justified by our experimental results. In a given game, the number of non-home run scoring balls greatly outnumber the home run scoring balls. A linear-regression based approach to predict non-home runs thus runs into the problem of data sparsity. Attribute bagging, on the other hand, enables our system to find matches that have similar home-run scoring patterns, given the set of match features, and thus avoids the sparsity issue altogether. Our experiments have shown (see Section 5) much degraded performance when using ridge regression for HR prediction, with the MAE for $\hat{R}_{eoi}$increasing from 16.5 runs to 29.4 runs.

Prediction of $\hat{HR}_i$ and $\hat{NHR}_i$ is accomplished using two sets of features – *historical features* and the *instantaneous features*, described next. Of these, historical features are critical for predicting runs for the first segment, since by definition, no instantaneous match data is available before the first segment.

3.4 Historical Features Our model consists of 6 *historical features* for each team in the dataset. They are mined from data across all matches played by a given team. The *historical features* of a team are as follows: (1) Average runs scored (by the team) in an innings; (2) Average number of wickets lost in an innings; (3) Frequency of being all-out;[3] (4) Average runs conceded in an innings; (5) Average number of opponent wickets taken in an innings; (6) Frequency of getting opposition all-out.

In what follows, we will use N to denote the total number of matches in the training dataset. Recall, n denotes the segment up to which match state is known. The first feature is calculated by dividing the total runs scored by the given team across the number of matches it played.

$$AverageScore = \frac{\sum_{i=1}^{N}(\text{Runs scored in } match_i)}{N} \quad (3.2)$$

The subsequent five features are self-explanatory and are calculated similarly to (3.2). Out of the 6 features, the first three represent the team's batting ability, while the last three represent the team's bowling ability.

[3]That is, losing all 10 wickets in an innings within 50 overs.

3.5 Instantaneous features In addition to the features mined from past game data, *i.e.*, the historical features, we incorporate several instantaneous match features in our prediction model. What has happened in the game so far is an important indicator for predicting game outcome. We extract the following instantaneous features from the dataset.

1. Home or Away: This is a binary feature describing if the batting team is playing in its *home ground*. If the match is played in a neutral venue, this feature carries no weight for both teams.

2. Powerplay: Powerplay is a restriction on the number of fielders that could be placed by the bowling team outside a certain range from the batsmen (usually 30 yards, approx. 27.432 meters). This restriction enables the batsmen to hit the balls aggressively and try and score home runs, with a relatively reduced risk of getting out. The first 10 overs of the game are mandatory powerplays, with two more instances of powerplay periods arbitrarily chosen by the batting and bowling team each, to occur at any point in the game up to the 45^{th} over. For any segment, the powerplay can occupy between 0 and 5 overs of the segment. Consequently, the value of this feature ranges from 0 to 1 in increments of 0.2.

3. Target: The goal of the team batting second is to achieve the $Target\ Score$, ($= Score_A$ +1 runs). This used as a feature

4. Batsmen performance features: For any given segment S_n, we identify four performance indicators for each of the two currently playing batsmen. They are batsman-cluster (to be described in section 3.6), #runs scored, #balls faced, and #home runs hit till segment S_{n-1}.

5. Game snapshot: This feature is a pair of game state variables, namely current score and #wickets (*i.e.*, #batsmen) left.

Instantaneous features 4 and 5 are explained in detail in Sections 3.6 and 3.7.

3.6 Batsmen Clustering In our dataset, there are more than 200 players who have faced at least one ball. Given data corresponding to 125 matches, learning the features for each of the 200 individual players is fraught with extreme sparsity. To give an example, given a currently playing batsman b and a current bowler ℓ, the probability that b has faced ℓ in earlier matches can be quite low. Even when b has faced ℓ before, the number of such matches can be too small to learn any useful signals from, for purposes of prediction. To quantify, if in a dataset of M matches, the average number of matches played by player b is m_b, and by player l is m_l (where $M \gg m_b$ and $M \gg m_l$), even

assuming independence, the probability that b and l played together is $\frac{m_b}{M} \times \frac{m_l}{M}$. To overcome this sparsity, we cluster the batsmen according to their batting skills, using the following four features: (1) Batting Average; (2) Strike Rate; (3) Home-run hitting ability; and (4) Milestone reaching ability. The first two features are standard metrics used to report batsmen stats in cricket. Although they are used to express a batsman's quality, they do not quite capture his skill as observed by cricket experts and proved by [16] and [1]. Hence for batsman clustering, we use Features 3 and 4 that capture the quality of batsmen more accurately.

Batting Average for a batsman is the ratio of the total number of runs he has scored across all matches, over the number of times he has gotten out. Strike Rate is the average number of runs scored per 100 balls, again calculated across all matches played. Both Batting Average and Strike Rate are standard player statistics used in cricket.

We measure the ability of a batsman to frequently hit home runs using

(3.3)
$$\textit{HR-hittingAbility} = \frac{\sum_{i=1}^{N} \#\text{home runs hit in match}_i}{\sum_{i=1}^{N} \text{balls faced in match}_i}$$

Scoring fifty runs or a hundred runs (commonly referred to as half-century and century) are considered batting milestones in cricket. Players who consistently and frequently reach these milestones are considered to be of very high caliber. To capture this, we define a metric called milestone reaching ability (MRA) as follows:

(3.4)
$$\textit{MRA} = \frac{\#\text{ of 50 \& 100 run scores in } N \text{ matches played}}{N}$$

MRA is thus a good indication of batsman quality. Using the above four statistics, we cluster the batsmen into 5 clusters using the k-nearest neighbor clustering. We chose 5 clusters based on the intuition that a team consists of opening batsmen, middle-order batsmen, all-rounders, wicket-keeper, and tail-enders, having different batting capabilities.

3.7 Game Snapshot Recall that the problem is, given the match state data up to segment $n < 10$, *i.e.*, runs scored R_i and wickets lost W_i in segment i, $1 \leq i \leq n$, we need to predict the number of runs for segment $n+1$. To facilitate this, we aggregate all of the information in segments S_1 to S_{n-1} and retain the information in segment S_n separately. More precisely, we set $R_{1:n-1} = \sum_{i=1}^{n-1} R_i$ and $W_{1:n-1} = \sum_{i=1}^{n-1} W_i$. We then incorporate the instantaneous features $R_{1:n-1}, W_{1:n-1}, R_n, W_n$ in our model. Since our score prediction is done separately for home and non-home runs, we use $HR_{1:n-1} = \sum_{i=1}^{n-1} HR_i$ and $NHR_{1:n-1} = \sum_{i=1}^{n-1} NHR_i$ and use these features instead of $R_{1:n-1}$, and predict the number of home runs and non-home runs for segment n.

For example, to predict the runs in segment S_6 (overs 26 to 30), runs scored and wickets lost in segments S_1 to S_4 are aggregated. Runs and wickets in segment S_5 are retained as such. This approach provides the game information till segment S_{n-1} and the game information in segment S_n separately to the model. This provides a broader snapshot of match state and also gives more importance to the immediately preceding segment.

Our learning algorithm, described in the next section, makes use of the aforementioned historical and instantaneous features up to a given segment to predict scores for subsequent segments and uses that to predict the overall score $\hat{R}_{eoi}$. As a special case, when $n = 0$, the algorithm relies on historical features alone to make its predictions.

4 Algorithm

4.1 Home-Run Prediction Model Using the historical and non-historical features discussed above, we predict the number of home runs $\hat{HR}_i$ for a segment S_i, using attribute bagging ensemble method [5] with nearest-neighbor clustering. Here, we choose random subsets of features for n classifiers with l features each and aggregate the overall results. Different sets of features corresponding to the previous states are chosen randomly and their nearest neighbors are identified from history, thereby leveraging the Markovian nature of segments. Number of features for every classifier is set to be the root value of the total number of features. The number of classifiers is experimentally determined. The intuition behind using nearest-neighbor algorithm is that information from similar match situations can be "borrowed" from the training dataset. We use Spearman's distance metric, that uses rank correlation to identify the top neighbor.

The Spearman distance is a measure of pairwise linear correlation between ranked variables. Suppose a sequence of values of two variables $u = (u_1, ..., u_m)$ and $v = (v_1, ..., v_m)$ are rank-ordered, then Spearman correlation coefficient is defined as:

$$\rho = \frac{\sum_i^m (u_i - \bar{u})(v_i - \bar{v})}{\sum_i^m (u_i - \bar{u})^2 \sum_i^m (v_i - \bar{v})^2} \tag{4.5}$$

It is the same as Pearson correlation coefficient except ranks are used in place of observed values.

Game features are ranked in the training and test dataset separately. The distance between a match in the

test dataset and one in the training dataset is the dot product of the (ranked) feature vectors of the matches.

After running n number of classifiers with l features each, we pick the top 5 neighbors based on frequency counts and average the home run hits. This number of neighbors, *5*, has been determined experimentally.

4.2 Non-Home-Run Prediction Using the same historical and instantaneous features, non-home runs of segment S_i, $N\hat{H}R_i$ is predicted by means of Ridge Regression [13].

The iterative algorithm to predict the runs R_i of future segments and consequently, R_{eoi} of the whole innings is given in Algorithm 1, where we use the following notation:
$N\hat{H}R_i$, predicted non-home runs in segment S_i
$\hat{HR}_i$, predicted home runs in segment S_i
$\hat{R}_i$, predicted total runs in segment S_i
$\hat{R}_{eoi}$, predicted end of innings score
$\Theta \leftarrow$ historical features
$n \leftarrow$ segment number till which match information is available
R_i, Runs scored in segment S_i where $1 \leq i \leq n$
$\Delta_n \leftarrow$ instantaneous features till segment S_i.

Θ is the set of historical features given in Section 3.4, which remain constant through the iterations since they are learned just once from the historical match data; n is the segment number till which, match state data, also called instantaneous features, Δ_n are available. At the start of the algorithm, features Θ and Δ_n are fed as input to the algorithm which proceeds iteratively to predict $\hat{R}_i$, for every segment i, $n{+}1 \leq i \leq 10$.

Algorithm 1: Prediction of Future Segments runs $\hat{R}i$ & End of Innings Score $\hat{R}_{eoi}$

Input : Θ, Δ_n, n
Output: R_i for every $n+1 \leq i \leq 10$, $\hat{R}_{eoi}$
1 **for** $i \in n+1 \leq i \leq 10$ **do**
2 **for** $j \in 1 \leq j \leq L$ **do**
3 $\Gamma_j \leftarrow RandomSubspace(\Theta, \Delta_{i-1})$
4 $\Phi_j \leftarrow NearestNeighbor(\Gamma_j)$
5 **end**
6 $HR_i \leftarrow MajorityVoting(\Phi_{1:L})$
7 $NHR_i \leftarrow RidgeRegression(\Theta, \Delta_{i-1})$
8 $\hat{R}i \leftarrow HR_i + N\hat{H}R_i$
9 $\Delta_i \leftarrow Update(\Delta_{i-1}, \hat{R}i)$
10 **end**
11 $\hat{R}_{eoi} \leftarrow \sum_{i=1}^{n} R_i + \sum_{i=n+1}^{10} \hat{R}i$

Using Θ and Δ_{i-1} (instantaneous features of previous segment), home runs $\hat{HR}_i$ of segment i is predicted using Attribute bagging algorithm as explained in section 4.1*(Line 2-6)*. For every classifier j in L (Number of Classifiers), a random subspace of features (Γ_j) is chosen from the overall feature space (Line 3). The nearest neighbor based on this subspace of features (Γ_j) is found out to be Φ_j (Line 4). Based on *majority voting* among the chosen neighbors ($\Phi_{1:L}$), the closest neighbor from the raining set is found and home-run information is borrowed. (Line 6) Using the same features Θ and Δ_{i-1}, non-home runs $N\hat{H}R_i$ are predicted using Ridge Regression as mentioned earlier in this section (line 7). $\hat{HR}_i$ and $N\hat{H}R_i$ are added together to give $\hat{R}i$, the predicted number of runs scored in segment S_i (Line 8).

It is to be noted that two of the instantaneous features (Section 3.5) namely Home/Away and Batsmen Cluster do not change through the course of prediction while Game Snapshot features are constantly updated based on predictions of the previous iteration as explained in section 3.7. Number of runs for segment i, $\hat{R}$i, predicted in this iteration, is added to the Game Snapshot features of Δ_{i-1}, thereby modifying it to Δ_i *(Line 9)*. This Δ_i is used to predict the $\hat{HR}_{i+1}$ and $N\hat{H}R_{i+1}$ in the next iteration.

The cumulative sum of all known runs R_i, $1 \leq i \leq n$ and all predicted runs $\hat{R}i$, $n < i \leq 10$, is the predicted End-of-Innings Score, $\hat{R}eoi$ *(Line 11)*.

4.3 Cold-Start If prediction is initiated without any match information, *i.e.*, before the start of the actual innings, then $n = 0$, and the algorithm starts prediction from S_1 through S_{10}.

In the first iteration when $i = 1$, no instantaneous features are available. As the opening pair of batsmen are known before the start of a game, their cluster ID numbers (from the feature Batsmen cluster) are used. In order predict $\hat{R}1$ accurately, we leverage the match venue information and use it as a feature along with the other historical features Θ. The match venue is incorporated as a multinomial feature called "Venue Class", which classifies the location of the venue into one of the following four continent clusters: (i) Australia/New Zealand; (ii) Indian Subcontinent; (iii) The British Isles; and (iv) The Caribbean/South Africa. This classification is significant since the pitch (*i.e.*, the area where the ball is bowled and pitched) and weather conditions across these region clusters are known to be substantially different, for the purposes of the game. Using the venue class as a feature is based on the intuition that a team tends to start differently in different venue conditions.

5 Experiments

5.1 Dataset Our dataset consists of 125 complete matches played between January 2011 and July

2012 among the 9 full-time ICC teams of Australia, Bangladesh, England, India, Sri Lanka, Pakistan, South Africa, New Zealand and the West Indies who have played more than 20 matches each excluding all rain-interrupted and rain-abandoned games. We split the dataset into training and test set with 100 and 25 matches respectively. We perform ten-fold cross validation on the 100 training matches and and present our results by testing on the remaining 25 matches. We crawled the data from `http://www.espncricinfo.com`, where ball-by-ball data on all the matches are available publicly. Team and batsmen statistics for mining historical features and batsmen clustering are also queried from their publicly-accessible statistics databases. Since ball-by-ball commentary data consists of transcription of the human interpretation of actual events, there are occasional missing values and errors. They were fixed either manually or by automated consistency checks with the end-of-over summary, scorecards, partnership information and other relevant game data. After the required data is gathered, they were aggregated and rolled-up to 5-over levels (since a segment is a collection of 5 overs) without loss of necessary information. Once the data is available in the processed form, running time for the model to learn the parameters across all the segments and testing by 10-fold cross-validation takes less than 5 seconds on a 4-core, 2.66 GHz machine with 8 GB RAM, running OpenSuse 12.3. The data is gathered from publicly available sources, as mentioned above. Owing to limited space, we combine and present the $innings_1$ and $innings_2$ prediction performances.

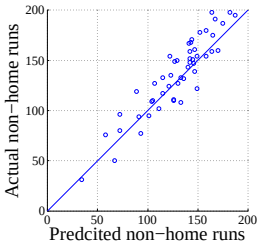

Figure 1: Total *non-home runs* scatter plot

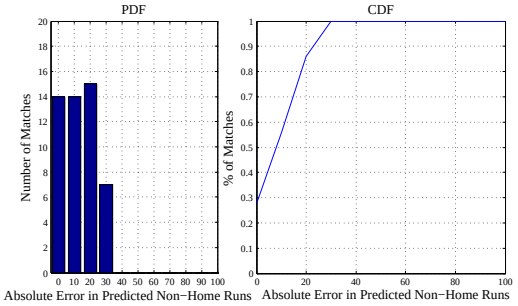

Figure 2: PDF and CDF of Total non-home run prediction error

5.2 Non-Home Run Prediction Performance Prediction of $N\hat{H}R_{eoi}$, which is the sum of individual $N\hat{H}R_i$ is shown in Figures 1 and 2. Figure 1 shows a scatter plot between the predicted and actual total non-home runs for both $innings_1$ and $innings_2$ combined. and demonstrates good agreement between the predicted and actual Non-Home Runs. Figure 2 shows the total non-home run prediction error distribution across all the matches. The figure on the left gives the Probability Density Function (PDF) and the one on the right gives the Cumulative Distribution Function (CDF). It can be seen that for more than 55% of the matches, the error margin is less than or equal to 10 runs in both innings. The median number of non-home runs in an innings in our dataset is 125. It is evident that our approach is very accurate for the majority of matches in the dataset, and performs poorly on only a small percentage of the games.

5.3 Home Run Prediction Performance We experimented with a number of distance metrics (namely, Jaccard, Hamming and Cosine measures) and compared them with the performance of Spearman distance metric., and the Spearman metric was observed to perform best (*i.e.*, have the lowest minimum average error). We report the performance of both nearest neighbor and attribute bagging with nearest neighbor technique and expectedly, attribute bagging performs better than nearest neighbors.

Figure 3 shows a scatter plot between predicted and actual total home runs. We have slightly worse

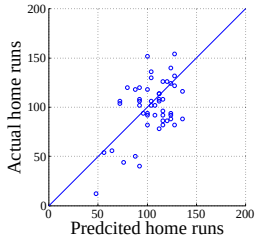

Figure 3: Total *home runs* scatter plot

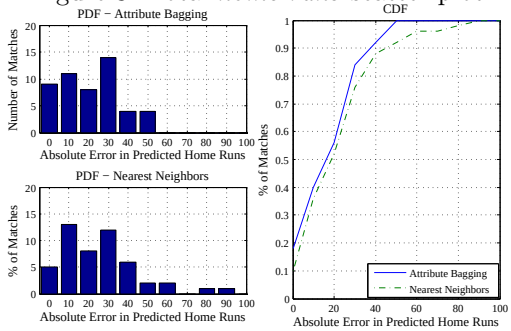

Figure 4: PDF and CDF of Total home run prediction error

agreement between predicted and actual, compared to non-home run prediction (Figure 1). Figure 4 shows that the error margin for the top 55% of the matches are less or equal to 20 runs. As mentioned in section 3, home runs are awarded either 4 or 6 runs based on where the ball lands. Hence, a single mis-prediction can induce a maximum error of 6 runs. It is also a more difficult problem to predict the number of runs scored through home runs, with the uncertainty arising from the very nature of the game as described in section 3. This is reflected in Figure 3 and 4. Figure 4 also shows that attribute bagging with nearest neighbors performs better than plain nearest neighbor algorithm.

5.4 End-of-Innings Run Prediction Performance Figure 5 shows the scatter plot for $\hat{R}_{eoi}$ for every match in the dataset.

Figure 6 shows the total score error distribution across the all the matches in the dataset. It can be observed that, for 50% of matches, prediction error has a maximum of 16 runs in Attribute bagging method, while for nearest neighbor method, it is close to 30 runs. 80% of the matches fall under the same prediction error ceiling, in attribute bagging method.

5.5 Runs in a Segment, $\hat{R}_i$ As match data streams in, we update the model (in five-over intervals) with ground truth and make $\hat{R}_i$ prediction for the next segment. Figure 7 shows the mean absolute error values for $\hat{R}_i$ scores across segments S_i, given match state data till segment S_{i-1}. MAE_i for both $innings_1$ and $innings_2$ lies within the range of 4 and 12 runs for all the segments, with errors increasing towards the later segments during the innings. Generally, until the middle overs (*i.e.*, up to over 35), teams are focusing on building a good foundation and consolidating their run scoring efforts. On the other hand, in the last 2 or 3 segments (from overs 35 to 50), it is common for the batsmen to try to hit most of the deliveries for home runs to maximize total runs; subsequently, a large chunk of the total score is accumulated in these last three segments. In doing so, batsmen take high risk and subsequently may get dismissed. Hence the match could turn in favor of any of the two teams with more or less equal probability. Because of such unpredictable nature of the game during these segments, it is difficult to estimate $\hat{R}_i$. Accordingly, the performance of our algorithm suffers somewhat in these segments, as demonstrated by the plots.

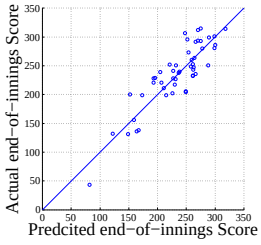

Figure 5: R_{eoi} scatter plot

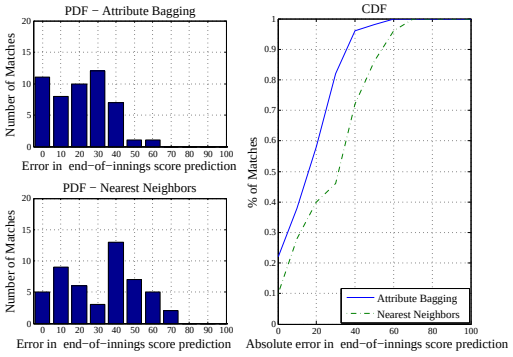

Figure 6: PDF and CDF of R_{eoi} prediction error

5.6 Performance Comparison with Baseline Model Bailey et al. [2] propose a model that predicts the $\hat{R}_{eoi}$ of a game in progress which is used to analyze the sensitivity of betting markets. Although addressing a different requirement, their framework allows making $\hat{R}_{eoi}$ predictions at the end of each innings. In Figure 8, we demonstrate the accuracy of our model considering their model as a baseline. At the end of each segment, $\hat{R}_{eoi}$ is calculated and compared with the actual R_{eoi} obtained at the end of innings from match data. As shown in the plot, both our model and that of Bailey et al. make better predictions, as more segments from the match in progress are input to the model. However, MAE for our model is significantly better for all the segments. It can be observed that our model significantly outperforms the baseline for both the innings.

We used our framework for predicting $\hat{R}_{eoi}$ for both innings, to predict the game winner. We found that the accuracy of this prediction is between 68% and

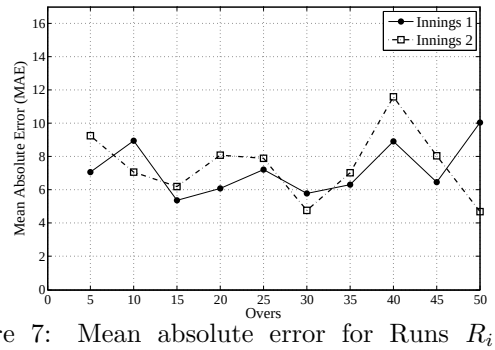

Figure 7: Mean absolute error for Runs R_i across each segments S_i, or, 5-over intervals for innings 1 and innings 2. Since the first and fore-most prediction R_1 for $i = 1$ gives the runs scored at the end of over number 5, the plots start from over 5.

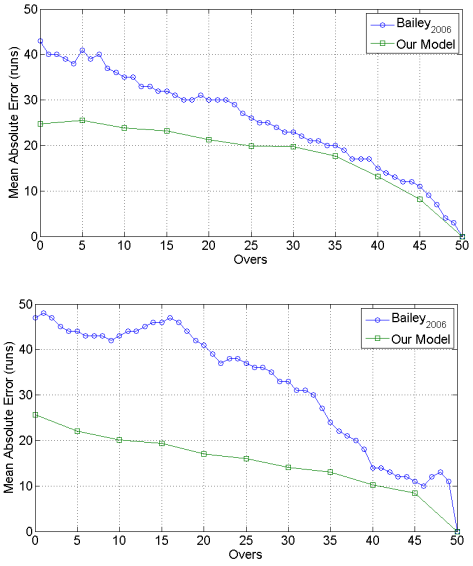

Figure 8: Mean absolute error in $\hat{R}_{eoi}$ prediction for innings 1 (top) and innings 2 (bottom) for both [2] and our model.

70%, which is robust regardless of the number of known segments. To our knowledge, this is the highest winner prediction accuracy reported in for ODI cricket.

6 Conclusion and Future work

The main goal of this paper is to learn a model for predicting game progression and outcome in one-day cricket. We developed separate models for home runs and non-home runs using historical features as well as instantaneous match features from past games that we identified. Ridge Regression and attribute bagging algorithms are used on the features to incrementally predict the runs scored in the innings. We demonstrated the quality and accuracy of our predictions with an extensive set of experiments on real ODI cricket data. In addition to predicting runs for future segments, our winner prediction accuracy is by far the highest reported in ODI cricket mining literature.

While our technique is significantly more accurate than the state-of-the-art, we are currently working to further reduce the prediction error. Furthermore, to make the prediction engine functionally complete, we intend to predict fall of wickets, overcoming the challenges presented from data sparsity. Finally, we aim to leverage bowler's features (in addition to the batsmen's) to improve the prediction accuracy even further.

References

[1] P. E. Allsopp and S. R. Clarke. Rating teams and analysing outcomes in one-day and test cricket. *Journal of the Royal Statistical Society. Series A (Statistics in Society)*, 167(4):pp. 657–667, 2004.

[2] M. Bailey and S. R. Clarke. Predicting the match outcome in one-day international cricket matches, while the game is in progress. *Journal of sports Science and Medicine*, 5(4):480–487, 2006.

[3] D. Beaudoin. *The best batsmen and bowlers in one-day cricket*. PhD thesis, Simon Fraser University, 2003.

[4] I. Bhandari, E. Colet, and J. Parker. Advanced Scout: Data mining and knowledge discovery in NBA data. *Data Mining and Knowledge Discovery*, 1(1):121–125, 1997.

[5] R. Bryll, R. Gutierrez-Osuna, and F. Quek. Attribute bagging: improving accuracy of classifier ensembles by using random feature subsets. *Pattern Recognition*, 36(6):1291–1302, June 2003.

[6] F. C. Duckworth and A. J. Lewis. A fair method for resetting the target in interrupted one-day cricket matches. *The Journal of the Operational Research Society*, 49(3):pp. 220–227, 1998.

[7] G. Gartheeban and J. Guttag. A data-driven method for in-game decision making in mlb: when to pull a starting pitcher. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '13, pages 973–979, New York, NY, USA, 2013. ACM.

[8] A. Kaluarachchi and A. Varde. CricAI: A classification based tool to predict the outcome in ODI cricket. In *5th International Conference on Information and Automation for Sustainability*, pages 250–255, 2010.

[9] H. H. Lemmer. An analysis of players' performances in the first cricket Twenty20 World Cup series. *South African Journal for Research in Sport, Physical Education and Recreation*, 30(2):71–77, 2008.

[10] A. J. Lewis. Towards fairer measures of player performance in one-day cricket. *The Journal of the Operational Research Society*, 56(7):pp. 804–815, 2005.

[11] S. Luckner, J. Schröder, and C. Slamka. On the forecast accuracy of sports prediction markets. In *Negotiation, Auctions, and Market Engineering, International Seminar, Dagstuhl Castle*, volume 2, pages 227–234, 2008.

[12] D. Lutz. A cluster analysis of NBA players. In *MIT Sloan Sports Analytics Conference*, 2012.

[13] D. W. Marquardt. Ridge regression in practice. *The American Statistician*, 29(1):3–20, February 1975.

[14] I. G. McHale and M. Asif. A modified Duckworth-Lewis method for adjusting targets in interrupted limited overs cricket. *European Journal of Operational Research*, 225(2):353–362, March 2013.

[15] K. Raj and P. Padma. Application of association rule mining: A case study on team India. In *International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–6, 2013.

[16] T. B. Swartz, P. S. Gill, D. Beaudoin, and B. M. deSilva. Optimal batting orders in one-day cricket. *Comput. Oper. Res.*, 33(7):1939–1950, July 2006.

[17] T. B. Swartz, P. S. Gill, and S. Muthukumarana. Modelling and simulation for one-day cricket. *Canadian Journal of Statistics*, 37(2):143–160, 2009.