Predicting Outcomes in Limited-Overs Cricket Matches

Natwar Modani, Manoj Kilaru
nmodani,kilaru@adobe.com
Adobe Research
Bangalore, India

Anjan Kaur *
ankaur@adobe.com
Adobe
Bangalore, India

Ritwik Sinha
risinha@adobe.com
Adobe Research
San Jose, US

Harsh Khetan
hkhetan@adobe.com
Adobe
Noida, India

ABSTRACT

Cricket is a popular sport in the commonwealth countries, particularly the limited over formats. As with any sport, predicting the outcome of the game of cricket is of popular interest. For the first innings, the task is to predict the eventual score that the team batting first will reach. For the second innings, the task is to predict the match result. Existing algorithms for predicting the outcome of limited over cricket matches are simplistic and their performance leaves room for improvement. In this paper, we provide novel features including team strength indicators that capture the situation of the match more comprehensively and accurately. We use a collection of state-of-the-art supervised Machine Learning (ML) approaches for the prediction tasks. Further, we also present an approach based on Long-Short Term Memory (LSTM) Networks to incorporate the oft-mentioned concept of 'momentum' for predicting the outcomes. We show with real data that the mentioned ML models outperform the current state of art (WASP) in outcome prediction for cricket. Further, we show that incorporating the proposed features improves prediction accuracy. Finally, the LSTM model outperforms all other models with the same set of features, thereby confirming that 'momentum' indeed helps us in better prediction of outcomes.

CCS CONCEPTS

• **Computing methodologies → Supervised learning**.

KEYWORDS

Cricket, Neural Networks

ACM Reference Format:
Natwar Modani, Manoj Kilaru, Anjan Kaur *, Ritwik Sinha, and Harsh Khetan. 2020. Predicting Outcomes in Limited-Overs Cricket Matches. In *7th ACM IKDD CoDS and 25th COMAD (CoDS COMAD 2020), January 5–7, 2020, Hyderabad, India.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3371158.3371166

Anjan is currently at Zumper,Inc San Francisco, emailId:anjankaur01.mkr@gmail.com.

1 INTRODUCTION

Commercial sports are increasingly playing an important and ubiquitous role in our lives [2]. The financial and anthropological implications of the popularity of modern sports has led to significant interest in analyzing data from sports [3]. More specifically, predicting the outcome of a game has attracted a lot of attention in Football (Soccer) [8, 15], Tennis [17], and Squash [16].

Cricket [1] is a popular sport, particularly in the commonwealth countries. Two teams of eleven players each participate in a cricket match. The matches are divided in innings. During one innings, one team 'bats', and the other team 'bowls'. The roles are reversed in next innings. Hence, cricket is asymmetric in nature, in the sense that the two competing teams have different objectives at a given point of time. In the limited over formats of the game, each team gets to bat once, and for a certain number of 'overs', and the team scoring more runs is declared winner [1]. In this paper, we focus on limited over international men's cricket matches of 50 overs for each side, which are popularly called as ODIs (One Day International). A short introduction to ODI cricket is given in Section 2.

The natural prediction task in a game of cricket changes between the two innings. In the first inning, the task is to predict the final score that the team batting first will reach. In the second innings, the task is to predict the winner. Also, we want to predict the outcomes while the match is in progress, and not just before the start of the match/innings. There have been some attempts in the past to predict the scores and winners of the match. However, these methods fail to account for several key attributes of the match, and hence, their accuracy is not satisfactory.

The current state-of-art methods include WASP [7]. WASP has the same two goals as our method, namely predicting the score for the team batting first and predicting winning probability for team batting second. However, it only looks at the number of wickets and balls remaining for predicting the first innings score, and additionally, the first innings score for the win probability calculation for second innings. On the other hand, [5] takes (recent) past performances of the teams into account, however, those are used only in the beginning and the in-play updates are based on

[1]unless the game is interrupted, in which case, the target for the team batting second may be revised

Duckworth-Lewis-Stern (DLS) [9, 18] method. Also, given that actual team composition can (and often, does) change, it may not reflect the actual team quality that is playing.

In our work, we extend these in two primary ways. First, we use a more comprehensive set of features. This includes a collection of features that capture the current state of the match, and historical data of performances of the players playing on the day. A naïve way to incorporate the past performances of player is to create features on per player basis. However, given that there are 22 players involved in a match, and there are many facets for each player that need to be captured, the number of features can become too large. Also, the features will have interactions amongst themselves, and hence, the number of possibilities to consider grows very rapidly. As the number of ODI matches played till date is not very large, we may encounter the problem of overfitting to the data. Instead, we summarize the past performance related indicators with only 4 features per team, and we define them in such a way that the interaction with other features (for example, number of overs remaining or number of wickets left) is also captured in those feature values.

Second, we explore a collection of state of the art supervised ML methods. This includes Linear Regression, Regularized Regression, Generalized Additive Models, and Recurrent Neural Networks with LSTM units. Momentum is a concept often mentioned by experts and commentators. Momentum refers to the recent trend in events of the match, and hence, conjectures that for predicting the likely outcome, how the match came to current stage is also important, not just the current state. However, we are not aware of any work which attempts to quantify and incorporate the momentum in prediction tasks. By using LSTM, we naturally incorporate the concept of momentum in our modelling, and indeed, we see a significant performance improvement in prediction tasks.

Formally, the contributions of this paper are the following. First, we provide methods for in-play prediction of the final score for the first innings and the winner during the second innings of a limited over cricket match, respectively. Second, we propose metrics to capture the abilities of the players who will participate in the rest of the game appropriately to improve prediction accuracy. Third, we use LSTMs to capture the effect of momentum, our experiments show that incorporating momentum improves the prediction accuracy. Finally, we present several prediction algorithms using LSTM and regression based approaches that outperform the baselines (DLS and WASP) in our experiments with a sizeable number of recent ODI matches.

2 AN INTRODUCTION TO ODI CRICKET

Cricket is a game of 'bat' and 'ball', played between two teams of eleven players each. Generally it is played on a roughly oval ground, with a rectangular central strip, known as the 'pitch'. There are a set of three 'stumps' (also referred to as 'wickets') on each of the two ends of the pitch. At any point of time, one team 'bats' and the other team 'bowls' (and 'fields'). One such turn where a team bats is called as 'inning'. The decision of which team will bat first is based on the 'toss' of a coin.

Two 'batsmen' from the batting team bat at any point of time. A batsman tries to hit the ball bowled by a bowler to score 'runs' by either running back and forth across the length of the pitch, or by hitting the ball past the boundary. The bowlers attempt to limit the run scoring and also try to get the batsman 'out', also called as taking wicket, by one of many possible ways (including clean-bowled, caught, leg before wicket or run-out). When the batsman gets 'out', he is replaced with another batsman, who is not yet out in this inning. If ten batsman of a team get out, the inning is considered as complete. Bowlers bowl in units of 'overs', consisting of six legal deliveries, bowled from one end of the pitch. The next over is bowled from other end of the pitch, and cannot be bowled by the same bowler. Generally, both batting and bowling are specialized skills, and very few players are good exponents of both the skills, and are referred to as all-rounders.

The matches played between international teams, where each team gets to bat once and get a maximum of 50 overs are called as One Day Internationals (ODI). In such matches, each bowler is allowed to bowl a maximum of ten overs (and hence, a team would need a minimum of five bowlers to bowl in the innings, unless the opposition team's ten batsman get out before fifty overs). The team scoring more runs is declared as the winner (and once the winner is determined, the play stops).

There is no limit on number of balls a batsman can play. Given that the total number of players are eleven (and five of those have to be good enough bowlers), if a batting team loses early wickets, the attempt is to not lose further wickets, even if it means scoring runs slowly (so that the batting team can utilize as many of 300 balls available as possible).

The bowlers generally bowl in such a way that the ball hits the 'pitch' before reaching the batsman. Due to the varied nature of soil used for making the pitches, the behaviour of ball after hitting the pitch can be very different on different grounds. Also, the dimension of each ground can be different. In addition, the weather also affects the nature of the game. Accordingly, the expected number of runs to be scored on different grounds can be very different.

3 RELATED WORK

Prediction of outcomes for sports is well studied [15–17]. However, for cricket, the state of the art outcome prediction methods leave room for improvement. One of the early approaches to use statistical methods in limited overs cricket was Duckworth-Lewis [9], which attempted to provide a fair method to adjust the target for an interrupted match. While the intention in this work was to adjust the target, it defines the concept of resources. Resources are assumed to be linearly translatable into runs, and accordingly this approach can also used as a method for predicting the scores or winner of the matches. Stern extended this approach in [18] to account for modern scoring rates. This refined approach is now referred to as DLS in honour of the three creators.

A more direct work on predicting the scores for first innings and win probability during the second inning is presented in [7]. Here, the idea is to find the probability of an elementary event (scoring 0, 1, 2, 3, 4 or 6 runs or losing a wicket) on a given ball, conditioned on the current number of wickets lost and balls already bowled. These probabilities are used in a dynamic program to calculate the expected number of runs to be scored. While it uses very little information, it turns out to be a fairly stable predictor in our experiments. This method is used as viewer engagement tool for several ODI series by broadcasters.

In [5], a pre-match margin of victory is predicted using multiple linear regressions (factoring in recent form, experience, home ground average, etc.), which is then adjusted using the DLS resources. A dynamic logistic regression based approach is presented in [4]. It divided the covariates in pre-match (those that do not change during the match, for example, which team won the toss, which is the home team, and ranking difference) and in-play (that change during the match, in particular, number of runs, wickets and balls left). Neither of these approaches factor in the actual team composition and situation of the match, in terms of quality of the batsman still to bat and the bowlers who can bowl.

Our key contributions are, novel features to capture team strength (which are based on the actual composition of the team, rather than based on team ranking/past results, which may be based on a different team composition), and use of LSTM models for prediction tasks that out-perform all other methods.

4 FEATURES

In this section, we define some features that help us incorporate more information about the match and improve prediction accuracy. First, we will define features to capture the team level batting and bowling strengths.

For batsmen, batting average and batting strike rates are popular metrics to capture their value. Batting average is defined as the number of runs per dismissal and batting strike rate is defined as the number of runs scored per 100 balls. Given that the batting innings are limited to 10 wickets and 300 legal balls, both of these metrics are indeed useful indicators of a batsman's value. Similarly, for a bowler, bowling average (number of runs conceded per dismissal achieved) and bowling economy rate (number of runs conceded per over) are good indicators of their value.

However, these metrics capture the strengths of individuals, and are difficult to use in a prediction task. We need to account for which players are already out, likely batting order, number of overs remaining, etc. for such metrics to be used, and it is not obvious how to model all these factors and their interactions. Hence, we propose creating team level strength features, accounting for all such factors, so that for the prediction algorithm, the modeling is simple. Also, we want the team strength features to be *specific to the current match situation* (since we want to do in-play predictions). A naïve way can be to take the average across all players in the team. However, such an approach would have several drawbacks. First, given that the batsmen arrive in a particular order, and there is no limit on how many balls a batsman can play, the batsmen coming earlier are more important than the ones who are scheduled to bat later (as they may not be required to bat for long, or, at all). Also, the weightage scheme should not be static as the innings progresses. Consider a situation after 10 overs when 3 wickets have fallen. It is likely that the team will lose a lot of wickets and almost all the batsmen will have to bat, and hence, while computing team batting strength, strengths of all the batsmen should play a significant role. However, if the team has lost the same 3 wickets after 47 overs, it is unlikely that batsmen at numbers 8 and below would be required to bat. Hence, their strengths should not be very important to calculate the team batting strength.

While the strength features are useful indicators, if they are based on very few samples, they may not be reliable. For example, while several players have scored more than 100 runs in their first match, but no one who has been playing for any significant number of matches has been able to keep that batting average. Similarly, on the bowling side, some people may start with a very good performance, but it is not likely that they would be able to sustain the same level of performance. Hence, for robustness purpose, we need to discount performance over a small number of events. Also, since we want to use these features for real life prediction tasks, we take player statistics into account till the start of the match. With these insights, we are now ready to define team batting and bowling strength features.

Team Batting Strength (at a given point of time): To define the team batting strength for batting average, we need to know the order of the batsmen to arrive at the crease, since the batsman coming earlier have a larger impact on team strength as explained earlier. However, we do not know the order in advance, and hence we define the order in data driven manner as follows. First, we keep the two batsmen on strike in first two places. A good metric for the batsman's quality is the product of their batting average and batting strike rate, which represents both their ability to score more runs and quickly. Accordingly, we order all the batsmen, who are not yet out and currently not batting, sorted by the product of their batting average and batting strike rate. The batsmen who are out are not included in calculating team batting strength features, as they cannot influence the match by their batting anymore.

Further, as mentioned earlier, for robustness, we discount the performance of a player over a shorter period as follows. Let $r_i(r)$ represents the runs scored till date and $r_i^*(a)$ be the 'raw' batting average for batsman i. Then we define adjusted batting average $r_i(a)$ for that batsman i as

$$\begin{aligned} r_i(a) &= r_i^*(a) && \text{if } r_i(r) \geq 1000 \\ &= r_i^*(a) \times r_i(r)/1000 && \text{otherwise} \end{aligned} \tag{1}$$

and the adjusted strike rate strength $r_i(s)$ of player is defined in terms of 'raw' batting strike rate strength $r_i^*(s)$ as

$$\begin{aligned} r_i(s) &= r_i^*(s) && \text{if } r_i(r) \geq 1000 \\ &= r_i^*(s) \times r_i(r)/1000 && \text{otherwise} \end{aligned} \tag{2}$$

Let the number of overs already bowled be represented by b_o. We define the positional importance of a batsman as

$$\begin{aligned} p_k &= 1 && \text{for } k = 1, 2 \\ &= (0.99 - 0.015 * b_o)^{(k-2)} && \text{otherwise, i.e., } k \geq 3 \end{aligned} \tag{3}$$

that is, the positional value of batsmen currently on strike (positions 1 and 2) is taken as 1, and for all subsequent batsman, it reduces, and the reduction is sharper, if fewer overs are remaining. For example, a batsman, who is ranked number 2 amongst the not-out batsman, who are not currently batting, and if the number of overs already bowled is 10, then the positional importance of this player would be $(0.99 - 0.015 * 10)^2 = 0.7056$, and if the number of overs already bowled is 30, then the positional importance of the same player is $(0.99 - 0.015 * 30)^2 = 0.2916$.

Table 1: Features used in prediction tasks

Grp	Feature Name	Description
1	CurOvers	Current number of overs bowled
1	CurRuns	Current team score
1	CurWickets	Number of wickets lost at this stage
2	B1RunsMade	Runs scored in this match by batsman 1 batting currently
2	B1StrikeRate	Runs in 100 balls in match for batsman 1 batting currently
2	B2RunsMade	Runs scored in this match by batsman 2 batting currently
2	B2StrikeRate	Runs in 100 balls in match for batsman 2 batting currently
2	DLSResLeft	Resources remaining according to DLS
2	Venue Type	Is it a home, away or neutral venue for batting team
2	WGrndAvgScore	Weighted mean score on this ground
2	Toss	Did the team batting first won the toss
3	TeamBatStrAvg	Team batting strength in terms of average (Ref. Eq.[4])
3	TeamBatStrStrRate	Team batting strength in terms of strike rate (Ref. Eq.[5])
3	TeamBowlStrAvg	Team bowling strength in terms of average (Ref. Eq. [6])
3	TeamBowlStrEconRate	Team bowling strength in terms of economy rate (Ref. Eq.[7])

We have to make some parameter choices to define the positional importance (the values 0.99 and 0.015 for positional importance in Equation 3). This choice was based on our domain knowledge and alignment between DLS value of resources and value of the wickets based on positional importance. However, due to lack of space, we will not explain this alignment in detail.

We now define the team batting average strength $R_t(a)$ and team batting strike rate strength $R_t(s)$ as

$$R_t(a) = (1/T) \times \sum_i p_k \times r_k(a) \tag{4}$$

$$R_t(s) = (1/T) \times \sum_i p_k \times r_k(s) \tag{5}$$

where T is the sum of positional importance of all the remaining players, i.e., $T = \sum_k p_k$, and $r_k(a)$ and $r_k(s)$ are the adjusted batting average and strike rate strengths for the k^{th} best player.

Team Bowling Strength (at a given point of time): We now define team bowling strength for runs per wicket. This is the weighted average of best five bowlers' bowling average, where the weights are the number of overs they can bowl from this point on. The current ODI rule allows each bowler to be able to bowl up to 10 overs per innings. Also, once a bowler bowls an over, he cannot bowl the next over within the same innings. Hence, for example, if a bowler has bowled his fourth over in 45^{th} over of the opposition's batting inning, he can only bowl at most 2 more overs (47^{th} and 49^{th}). Hence, to compute the team bowling strength, we should take the weighted average of best bowlers' bowling strengths, weighted by the number of overs they can bowl from this point on.

$$W_t(a) = \sum_i b_i * w_i(a) \tag{6}$$

where $W_t(a)$ is the team bowling strength for bowling average, b_i is number of overs the i^{th} best bowler can bowl and $w_i(a)$ is bowling strength for average of the i^{th} best bowler. Similarly, team bowling strength for economy $W_t(e)$ is defined as

$$W_t(e) = \sum_i b_i * w_i(e) \tag{7}$$

where b_i is as defined above and $w_i(e)$ is bowling economy rate (runs conceded per over bowled) of the i^{th} best bowler.

The best bowlers for a team were decided by ordering them according to the multiplication of their bowling average and bowling economy rate (the lower the value, better it is). For robustness, we adjusted the bowling strengths for the bowlers who have not yet bowled minimum of 200 overs, as

$$w_p(a) = w_p^*(a) \times (200/bb(p)) \qquad \text{if } bb(p) \leq 200 \tag{8}$$

$$= w_p^*(a) \qquad \text{otherwise} \tag{9}$$

where $w_p(a)$ represents the bowling strength for average, $w_p^*(a)$ represents the raw bowling strength average and $bb(p)$ represents the number of overs bowled by the player p. Similarly, for bowling economy rate is adjusted for robustness as

$$w_p(e) = w_p^*(e) \times (200/bb(p)) \qquad \text{if } bb(p) \leq 200 \tag{10}$$

$$= w_p^*(e) \qquad \text{otherwise} \tag{11}$$

where $w_p(e)$ and $w_p^*(e)$ represent the bowling strength for economy and the raw bowling economy rate of player p, respectively.

We also considered the ground average score. It is the average first innings score for this ground. However, as cricket matches are played on grounds where no or very few matches may have been played in the past, we take a weighted average of average first innings score for the ground and all the matches played anywhere in the world. The weight of the ground specific first innings is taken as w, and the weight of anywhere in the world is taken as $(1 - w)$, so that as the number of matches played on the ground increase, we give more importance to the statistics corresponding to it.

We also add features for the number of balls played, number of runs made, and strike rates of the two batsman that are currently batting. The reason for adding these features is due to the widespread notion, that once a batsman is well-set (i.e., have played a few balls), they are more likely to be able to bat more freely. Also, these features capture their 'form' in the current match. We take the batsman who has scored more runs as 'batsman 1' amongst the two batsman currently batting. Finally, Table 1 gives all the features we used in our prediction tasks.

5 PREDICTION ALGORITHMS

Our prediction tasks are the following. During or at the start of the first innings, the task is to predict first innings final score. During or at the start of the second innings, the task is to predict the winner. As an intermediate step, we first predict the win margin for the second innings, and then use it to predict the winner. We define win margin W_m as:

$$W_m = ((S_s * 100/R_s) - S_f)/S_f \tag{12}$$

where S_s is the final second innings score, R_s is the resources used in second innings (based on DLS table, derived from balls played and wickets lost) and S_f is the first innings score. Basically, this is projecting the second innings score if the team batting second was to use all 100% resource, subtracting the first innings score (as all 100% resources are always used in first innings), and normalizing it with respect to the first innings score. We then convert the win margin into win probabilty using a sigmoid function.

We now briefly discuss the prediction algorithms used. These are a linear regression model (LR), a linear regression with the L1 regularization or Lasso, Lasso with interaction terms, and Generalized Additive Model. In addition to these, we also tried two models where we build separate regression for each over. The specifics of these models are described below. But first, we define some notation. Let y_{ij} denote the target variable, which is the final first innings score in case of first innings prediction task, and win margin (as defined above) for the second innings of match i. The index j stands for the over of the inning at which we are interested in making the prediction. While y_{ij} is constant with respect to j, this index is necessary because we would like to make predictions at each j ($j = 1, 2, \cdots, 50$). Next, let's define x_{ijk} as the value of the k^{th} feature in the i^{th} match at the end of the j^{th} over.

Linear Regression (LR): A linear regression [10] is given by the following model.

$$y_{ij} = \beta_0 + \beta_1 x_{ij1} + \beta_2 x_{ij2} + \cdots + \epsilon_{ij},$$

where $\epsilon_{ij} \sim N(0, 1)$ iid. The parameters β_i are estimated by minimizing the function $\sum(y_{ij} - \beta_0 - \sum \beta_k x_{ijk})^2/n$, n being the number of data points. The predictions for a particular i, j, k is given by using the parameter estimates from the above fit.

Linear Regression with Regularization: A linear regression with the L1 regularization (Lasso) [19] aims to increase prediction accuracy by shrinking the parameter estimates towards 0. It is also known to perform natural feature selection. The Lasso regression performs the following minimization $\sum(y_{ij} - \beta_0 - \sum \beta_i x_{ijk})^2/n + \lambda||\beta||_1$. The hyperparameter λ is selected using cross validation. More recent work has shown that it is possible to perform best-subset regression in a computationally feasible manner [13]. This approach minimizes the the following function $\sum(y_{ij} - \beta_0 - \sum \beta_i x_{ijk})^2/n + \gamma||\beta||_0 + \eta||\beta||_2^2$.

Regularized Regression with Interaction Terms: The above models make the assumption that there are no interaction effects. This is likely a naïve assumption, for instance, having 5 wickets left in the 5th over is likely to affect the batting team a lot more than having 5 wickets left in the 45th over. To model such interactions, we build the following model, which contains all the pair-wise interaction terms.

$$y_{ij} = \beta_0 + \sum_k \beta_k x_{ijk} + \sum_{k,l} \beta_{kl} x_{ijk} x_{ijl} + \epsilon_{ij}$$

we additionally perform regularization on this model, which helps us perform feature selection.

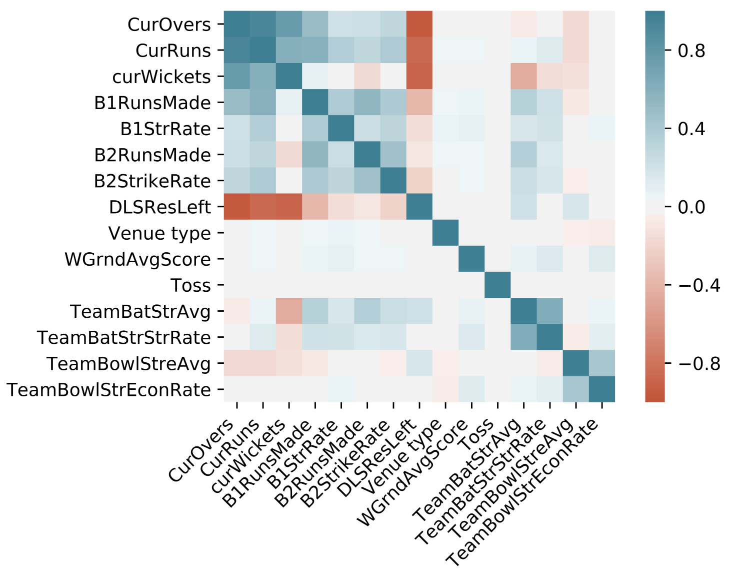
Generalized Additive Model (GAM): The above two regression models make the assumption that the final score is a linear function of the features. However, this may not be a good assumption for our problem. To enable us to model non-linearities in our model we explore GAM [12]. GAM makes the assumption that $E(y_{ij}|x_{ijk}) = \beta_0 + \sum_k g_k(x_{ijk})$, where $g_k(\cdot)$ are non-linear functions of the features, for example smoothing splines or local linear regression smoothers [11]. The optimization of this function is achieved using the backfitting algorithm [6]. Having non-linear terms in the regression requires more data, to reduce the number of non-linear terms in the regression, we selected only those features which are seen to have a non-linear relationships as seen from their univariate plots.

Over Level Models: The next natural question is to consider the possibility that the final score at the end of the innings is a function of the stage of the match. The difference in the relationships could be so distinct that we may model these as separate regression models. To this end, we explored the following model: for all j in $1, 2, \cdots, 50$. model $y_{ij} = \beta_{j0} + \sum_k \beta_{jk} x_{ijk} + \epsilon_{ij}$. A similar model with non-linear functional relationships, $g_k(\cdot)$, can lead to GAMs which model the final score based on features at each over separately. These two models are called **LR over-wise** and **GAM over-wise**, respectively.

LSTM: We use deep neural networks to model the non-linearities and interaction between features in our model. In addition, the concept of *momentum* says that not only the current state of match, but how this state was reached is also an important factor in determining how the rest of the match will unfold. For example, consider a team's score is 200 for loss of 3 wickets in 39 overs, and they lose 2 wickets for 4 runs in the next over. On the other hand, another team in another match is at 195 runs for loss of 5 wickets in 39 overs, and score 9 runs without losing a wicket in next over. In both cases, the score at the end of 40 overs is 204 for the loss of 5 wickets, but one would expect the approach of teams in the two cases to be different. LSTMs [14] are known to be able to remember important things about the input they received until this point, making them more precise in predicting what's coming next. We experimented with different architectures combining LSTMs and linear layers and found that 2 layers of LSTM stacked provided the best models for our problem.

6 EXPERIMENTAL RESULTS

Dataset: We collected the data for all men's ODI matches held from 13 June 2006 to 21 May 2017, which covers 1465 matches. Of these, we discarded matches that were interrupted (for example, due to rain) and the number of allocated overs were reduced as the dynamics of such matches is very different. This resulted in 961 matches available for our experiments. The collected data consisted of ball by ball score of the individual players (balls played, runs scored, already out or currently on strike for batsman, and balls bowled for the bowlers). We also collected information about the



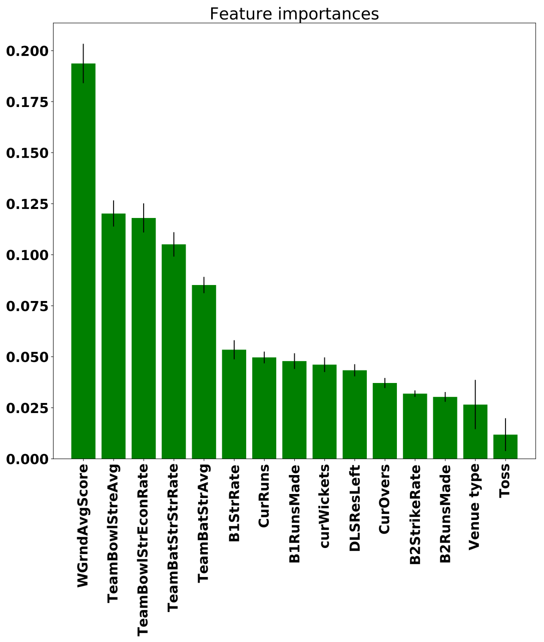


Figure 1: Correlation between feature is generally low (left). Proposed features have high importance (right). The black lines represent the confidence intervals

toss and venue type (home, away or neutral for team batting first). We also collected historical statistics about the players and grounds as of the date of match in order to compute the proposed features.

We split the data set temporally in such a way that first 90% of the matches are used for training and last 10% of the matches are used for test. This is the scenario in real world, where one cannot expect to know the data for the matches that are yet to take place. For academic purposes, we also did 10-fold cross validation, and the results were very similar.

Baselines: The most obvious baseline for our method is WASP. We also adapted the DLS (Duckworth-Lewis-Stern) method for predicting the outcomes in the following manner. We use the resource table to find out the amount of resources left at the completion of the given ball in percentage terms. We take the current score, and scale it to 100%. Let the number of runs scored after completion of ball b be r and number of wickets lost be w. We translate it to resources left, denoted by $\mu(b, w)$, using the DLS table. We then project the current number of runs scored r into the final score F as a linear extrapolation

$$F = r * 100/(100 - \mu(b, w))$$

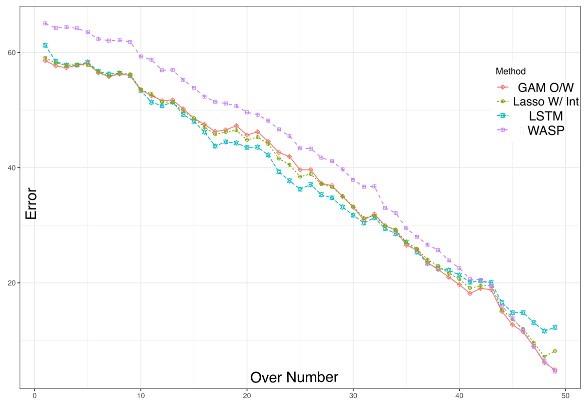
The denominator $(100-\mu(b, w))$ represents the percentage resources already consumed and r represents the number of runs already scored.

If this is the first innings, the prediction task is complete. If it is the second innings, we check if the extrapolated score is greater than the first innings actual score, then we predict that team batting second would win, else, we predict that team batting first would win. If the extrapolated score is equal to the first innings score, we predict a tie.

Feature Significance: First we investigate the quality of features proposed by us. For this purpose, we plot the feature correlations on left side of Figure 1, and feature importance on the right side. As we can see, there are a few variables which are highly correlated. But there seems to be sufficient orthogonal information that is likely to help us build good models. The features that are strongly correlated are curOvers and curRuns, with curWicket showing slightly less strong correlation with these. All of these are anti-correlated to DLSResLeft feature, which is expected. Since DRSResLeft is a non-linear function of remaining overs and remaining wickets, but is assumed to be linearly translatable to runs, hence we include it in our feature set. In terms of importance of features, while ground average unsurprisingly turns out to be the most important feature, the next set of important features are the team strength features proposed by us. This shows that the features we have proposed are useful.

Results: We measure the performance of the algorithms in terms of RMSE (Root Mean Square Error) and MAE (Mean Absolute Error). As the trends for both RMSE and MAE are very similar, we only report the results for RMSE.

First Innings Prediction: The first experiment reported in Table 2 compares the performance of the baseline methods with the proposed prediction algorithms for predicting first innings score in terms of RMSE. The performance of DLS is very poor as expected (mainly due to very early overs, where the estimates can be very unreliable). The LSTM using all the features outperform WASP by 19%, and best non-LSTM method by 7.47% (Lasso with Interaction terms using all the features). The results show that the team strength variables improve the prediction accuracy by 5-6% compared to



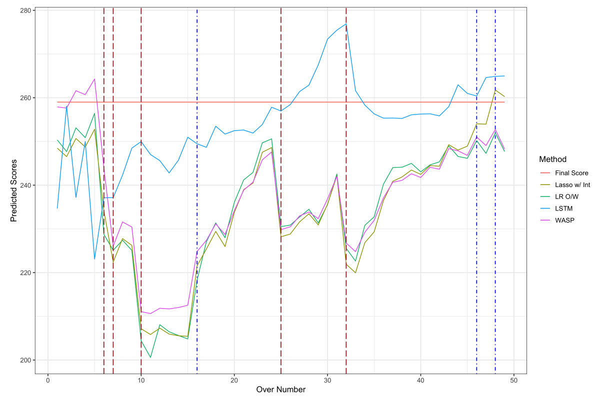


Figure 2: Over-wise error rate (RMSE) for various prediction algorithms for first innings score prediction (left) and an example match showing how the predictions change (right). The red dashed vertical lines represent the times when wickets fall, and purple vertical lines represent overs where ≥ 10 runs are scored. We are not plotting DLS error rates for first innings as those are much higher, and the readability of graph is affected.

Table 2: First Innings Score Prediction Performance in terms of RMSE

Features	LR	LASSO	GAM	LR O/W	Lasso w/Int	GAM O/W	LSTM	WASP	DLS
G1	44.76	45.14	43.85	43.62	44.03	43.58	40.98	45.58	121.13
G1, G2	43.60	43.81	43.31	43.19	43.25	43.20	40.33	-	-
G1 - G3	42.73	42.91	42.27	41.19	41.15	41.20	38.30	-	-

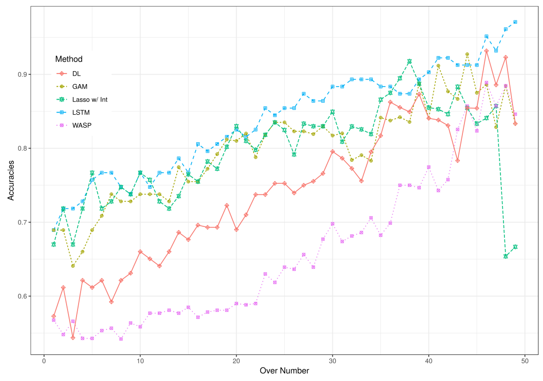
the same methods without using these features. The difference between LSTM and other regression methods (of about 7.5%) can be attributed to momentum and better ability to model non-linearities. We also note that WASP's simple approach is quite competitive if we only use the same features in other prediction methods also, but its inability to readily extend when more information is available is a major shortcoming.

To further investigate the behavior of the prediction algorithms, we plot the RMSE on a per over basis in Figure 2 left sub-figure. As one can see, initially, our proposed prediction methods (especially, GAM over level and Lasso with interaction terms) outperform WASP, which implies that additional information is useful in order for predicting the eventual score. However, as we get close to the end of innings, WASP catches up with the best model (GAM over level). Basically, this implies that in last 3 or 4 overs, only number of wickets remaining matters, as the teams attempt to hit as many runs as possible. On the other hand, LSTM performs really well during the middle overs, but it does not perform very well in first 2-3 overs and last 4-5 overs. The right hand side subfigure of Figure 2 shows an example match, and how the predictions change. The red dashed vertical lines represent the times when wickets fall, and purple vertical lines represent overs where ≥ 10 runs are scored.

Second Innings Prediction: Now let's look at results of the prediction task for second innings, where we also include the final score of team batting first in addition to all the features mentioned earlier. First, we present the results for the margin of victory for the team batting second in Table 3 for DLS (as baseline) and our proposed methods. Please note that WASP does not give us a win margin. Again, we see that our proposed method outperforms the baselines. Also, the LSTM achieves the best win margin prediction accuracy, about 8% higher than any other method, and about 56% better than DLS. While the team strength features help improve the prediction accuracy for LSTM, for other methods, there is no conclusive improvement by using these features. Curiously, in this case, with only G1 set of features, LSTM does not perform very well.

If we take a simple method of predicting the winner as, if the win margin is positive, the team batting second is predicted to win, and if it is negative the team batting second is predicted to lose (with zero denoting a tie). Figure 3 left side sub-figure presents the second inning win prediction results on an over-wise basis using this methods. We note that LSTM performs the best for the entire range, unlike in the first innings.

However, given that a small +ve win margin may not be indicative of definite win for team batting second, we learn a sigmoid to convert the win margin to a win probability. As the game progresses, the certainty about the result also increases, even with the same predicted win margin. Hence, we learn 1 sigmoid for each over. The right side subfigure in Figure 3 shows the sigmoid functions we obtained for overs 5, 25, and 45. As one can see, the sigmoid changes steeply for the later overs. We use the sigmoids to convert the win margin into win probability and compute the RMSE for win probability prediction. These are presented in Table 4. As we can see, the prediction accuracy of our methods for win probability is significantly better than DLS and WASP.



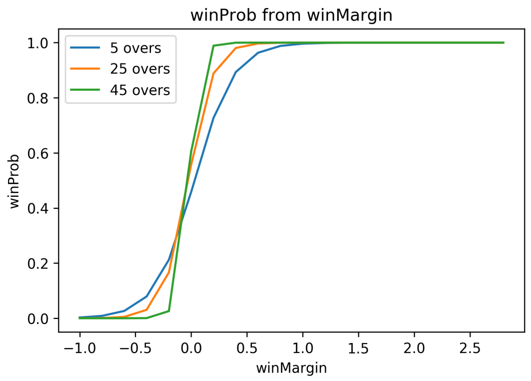


Figure 3: Second innings win margin prediction using all the features (left) and Converting Win Margin to Win Probability (right)

Table 3: Accuracy for second innings Win Margin Prediction Task in terms of RMSE (lower is better). WASP does not predict win margin.

Features	LR	LASSO	GAM	LR O/W	Lasso w/Int	GAM O/W	LSTM	DLS
G1	0.3068	0.3143	0.2797	0.3112	0.2956	0.2863	0.4129	0.5864
G1, G2	0.3049	0.3183	0.2789	0.3137	0.2999	0.2880	0.2651	-
G1 - G3	0.3062	0.3209	0.2777	0.3138	0.2964	0.2874	0.2555	-

Table 4: Second innings win probability prediction accuracy in terms of RMSE (lower is better).

Features	LSTM	GAM	LR O/W	Lasso w/Int	GAM O/W	DLS	WASP
G1	0.3191	0.1534	0.1626	0.1658	0.1589	0.4946	0.3045
G1, G2	0.1472	0.1572	0.1622	0.1598	0.1581	-	-
G1 - G3	0.1459	0.1573	0.1613	0.1576	0.1585	-	-

7 CONCLUSIONS

We present an in-play prediction method for the first inning score and a win probability scoring method for the team batting second. We use novel features and state-of-art techniques for this purpose and show that our method outperforms previously proposed methods. All the methods we are aware of, that attempt to incorporate player characteristics tend to not treat the new players correctly. While we present a robust mechanism to factor in the lack of data, incorporating List A records for new players, can improve results.

REFERENCES

[1] [n.d.]. Cricket - Wikipedia. https://en.wikipedia.org/wiki/Cricket. (Accessed on 10/17/2018).

[2] [n.d.]. Deloitte's Sports Industry Trends for 2018 | Deloitte US. https://www2.deloitte.com/us/en/pages/consumer-business/articles/sports-business-trends-disruption.html. (Accessed on 10/17/2018).

[3] [n.d.]. MIT Sloan Sports Analytics Conference. http://www.sloansportsconference.com/. (Accessed on 10/17/2018).

[4] Muhammad Asif and Ian G. McHale. 2016. In-play forecasting of win probability in One-Day International cricket: A dynamic logistic regression model. *International Journal of Forecasting* 32, 1 (2016), 34–43.

[5] Michael. Bailey and Stephen R. Clarke. 2006. Predicting the Match Outcome in One Day International Cricket Matches, while the Game is in Progress. *Journal of Sports Science Medicine* 5, 4 (2006), 480–487.

[6] Leo Breiman and Jerome H Friedman. 1985. Estimating optimal transformations for multiple regression and correlation. *Journal of the American statistical Association* 80, 391 (1985), 580–598.

[7] Eric Crampton and Seamus Hogan. 2012. Cricket and the Wasp: Shameless self promotion (Wonkish). http://offsettingbehaviour.blogspot.co.nz/2012/11/cricket-and-wasp-shameless-self.html. [Online; accessed 2-October-2018].

[8] Stephen Dobson, John A Goddard, and Stephen Dobson. 2001. *The economics of football*. Cambridge University Press Cambridge.

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

[10] Julian J Faraway. 2002. Practical regression and ANOVA using R.

[11] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. 2001. *The elements of statistical learning*. Vol. 1. Springer series in statistics New York, NY, USA:.

[12] Trevor Hastie and Robert Tibshirani. 1987. Generalized additive models: some applications. *J. Amer. Statist. Assoc.* 82, 398 (1987), 371–386.

[13] Hussein Hazimeh and Rahul Mazumder. 2018. Fast Best Subset Selection: Coordinate Descent and Local Combinatorial Optimization Algorithms. *arXiv preprint arXiv:1803.01454* (2018).

[14] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[15] Lars Magnus Hvattum and Halvard Arntzen. 2010. Using ELO ratings for match result prediction in association football. *International Journal of forecasting* 26, 3 (2010), 460–470.

[16] Tim McGarry and Ian M Franks. 1994. A stochastic approach to predicting competition squash match-play. *Journal of sports sciences* 12, 6 (1994), 573–584.

[17] Ian McHale and Alex Morton. 2011. A Bradley-Terry type model for forecasting tennis match results. *International Journal of Forecasting* 27, 2 (2011), 619–630.

[18] Steven E Stern. 2016. The Duckworth-Lewis-Stern method: extending the Duckworth-Lewis methodology to deal with modern scoring rates. *Journal of the Operational Research Society* 67, 12 (2016), 1469–1480.

[19] Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* (1996), 267–288.