Cricket Match Outcome Prediction Using Sequence Models

Team Name: EnTrophy
Saikat Dutta (23D2031)[1], Samrat Mukherjee (23D1599)[1],
Naay Balodia (23M2166)[1] Shubhajeet Dey (23M0819)[2],
Vaibhav Bhatnagar (23M1062)[3], Chanchal Gupta (23M1069)[3]

[1]Center for Machine Intelligence and Data Science
[2]Computer Science and Engineering
[3]Electrical Engineering

November 28, 2023

1 Introduction

Cricket is a dynamic and unpredictable sport, and predicting match outcomes in real-time is a challenging task due to the nature of the game. Such types of predictive models enhance the experience of the viewers and simultaneously can provide feedback to the team staff to make better decisions during the game.

The primary goal of our project is to predict the ball-by-ball win percentage in the second innings of any given T20 international cricket match. In addition to basic match state features, we have incorporated features based on batter and bowler statistics. The outcome of the match heavily relies on the batsmen on the crease and the bowler who is balling the over. The field placement and the fielders in the respective positions are also important, but using them as features is difficult due to lack of availablity of such data.

Features corresponding to each ball in the second innings are arranged as a sequence and fed to an LSTM model for winner prediction.

We have also experiment with a new loss function based on Duckworth-Lewis method (DLS). It is a target score calculation algorithm which the governing body of International Cricket had developed for scenarios where the match is delayed due to rain or any other reasons. This score gives us a measure of the par score which, represents the target that a team would be expected to reach based on the resources available to them. This score is adjusted according to the state of the game when the interruption occurred.

2 Literature Review

Viswanandha et.al. (1) was one of the first works for predicting outcome of a T20 match. It dynamically updates the game context as the game progresses after each over. It also included the relative strength between the two teams playing the match by using the individual players recent and career statistics. They used traditional supervised learning algorithms like SVM, Decision Trees, KNN, and Random Forest algorithms to predict the winner of the match.

Chakwate. and Madan (2) used LSTM to predict the outcome of 50 over matches. They used ball by ball statistics of runs scored in that ball from the bat and as extra. Using this rudiementary form of input (and augmenting the wicket information and the target score), they achieved an accuracy of 98.4%.

3 Dataset Analysis

In our dataset, there are 101 different countries which are present, thus again highlighting the popularity of the T20 format. The teams which played the least number of matches are Iran and Mongolia, both having played a single game. The top-5 teams which have played the most number of matches are Pakistan(334 matches), India(329 matches), England(308 matches), Australia(299 matches), and New Zealand (290 matches).

In our dataset there are 2910 matches with winners at the end of the match, with the team that batted first emerged victorious in 1469 instances. Conversely, in the remaining subset of matches, comprising 62 instances, no decisive result was achieved due to weather conditions. These 62 matches were excluded from our analysis due to the inherent challenge in accommodating weather-related disruptions in our sequential modelling.

Furthermore, among the dataset entries, 30 matches concluded in a tie. Notably, 25 of these tied matches proceeded to a "Super-over," a special tie-breaker mechanism, where each team get one over to score as many runs as possible at the maximum expense of 2 wickets. In the context of Super-over matches, 14 instances witnessed the team batting first achieving success, while 11 instances saw the team bowling first emerging victorious in the Super-over format.

In two instances, a "Bowl-Out" format was employed to resolve the tie. Additionally, two ties were resolved using the Duckworth-Lewis (D/L) method, and one tie remained unresolved, with neither a Bowl-Out nor a Super-over conducted.

In summary, our dataset encompasses 1484 matches in which the team batting first emerged victorious, while 1453 matches were won by teams that successfully chased the target set by the opposing team. Notably, 65 matches concluded without a conclusive result due to various factors, primarily adverse weather conditions.

Out of 3002 matches, there were 2937 matches with a decisive winner out of which 1475 times the team who won the toss won the match.

4 Proposed Method

4.1 Feature Extraction:

For each ball in the second innings, we create a feature set consisting of the following features.

- **Basic features:** We used the following basic match state information as features:
  - **Over and ball information:** Over and ball information is encoded as a decimal value i.e. "<over_no>.<ball_no>"
  - **Runs to win:** This value is calculated by subtracting runs scored till now from target score.
- **Batter features:**
  - **Batter average:** The average number of runs scored by a batsman per dismissal, indicating their consistency and effectiveness in scoring runs.
  - **Batter strike rate:** The speed at which a batsman scores runs, measured as the number of runs scored per 100 balls faced, reflecting their ability to accelerate the scoring rate.

  We use the above features for both on-strike and non-strike batters. These features are calculated for each batter in the training dataset. During test time, if a new batter is encountered, then the following method is used to approximate the batter's statistics:

  → Precompute average batter statistics of batters who bat at positions 1 to 6 *(Group-1)*, positions 7 to 9 *(Group-2)*, and positions 10-11 *(Group-3)*.

  → Determine the batting position of the unknown batter in the given 'test' match.

  → Based on the batting position, we can use the average statistics of the corresponding group. For example, if the unknown batter walks out to bat at 7^{th} position, we can use average statistics of Group-2.
- **Bowler features:**
  - **Bowler average:** The average number of runs conceded by a bowler per wicket taken, serving as a measure of a bowler's effectiveness in dismissing batsmen while minimizing runs conceded.
  - **Bowler strike rate:** The frequency at which a bowler takes wickets, calculated as the number of balls bowled per wicket taken, illustrating the bowler's ability to make breakthroughs in terms of dismissals.

– **Bowler economy:** The average number of runs conceded by a bowler per over bowled, providing insight into a bowler's ability to control the run flow and maintain pressure on the batsmen.

The above features are calculated for each bowler in the training dataset. During test time, if a new bowler is encountered, we use the average of these statistics and are used to approximate features for that bowler.

4.2 Model details

We have used an LSTM model (Long Short Term Memory model) for our sequence-level prediction. LSTM architecture designed to overcome the limitations of traditional RNNs in learning and remembering long-term dependencies in sequential data. The key problem LSTM addresses is the vanishing gradient problem, which can occur in standard RNNs. The vanishing gradient problem makes it difficult for the network to learn and retain information from earlier time steps in a sequence, especially when the sequence is long.

In our problem, the prediction needs to take into account of how the game has proceeded so far, in order to perform better at test time, thus motivating such architectural choice.

We use a single LSTM cell with hidden feature dimension of 64 and output feature dimension of 2.

4.3 Loss function

In our work, we create soft labels (pseudo winning probability) for each ball based on DLS par score and compute Cross Entropy Loss. In the upcoming sub sections 4.3.1, 4.3.2, 4.3.3 we are going to explain the workflow for this computation.

4.3.1 DuckWorth Lewis par score (DLS)

DLS method was designed to calculate the target score (number of runs needed to win) for the team batting second in a limited-overs cricket match interrupted by weather or other circumstances. It was adopted officially by the ICC in 1999.

The DLS method works using the notion that teams have two resources with which to make as many runs as they can - these are the number of overs they have still to receive and the number of wickets they have in hand. From any stage in their innings, their further run-scoring capability depends on both these two resources in combination.

Now, this method has also been extended for the calculation of par score, which examines the team's performance during play time. It helps in understanding whether the chasing side is doing better or worse than they would need to do on average to reach the target score.

The DLS par score for team two is calculated as follows:

$$DLS\ par\ score = (Target\ score) * \frac{Resource\ utilized\ by\ team\ two}{Total\ resource\ utilized\ by\ team\ one}$$

The resources mentioned here are precalculated and stored in a table format, for all combinations of wickets lost and whole overs left. The table also known as D/L Standard Edition table for T20 is available to public by ICC.

4.3.2 Calculation of Soft labels

Now, using the DLS par score at each ball, we try to calculate soft labels for each ball in the second innings. Here we moved to soft labels from hard labels (final win/loss) as using hard labels seemed unintuitive for initial stages of the play. Like for example, on using hard labels if a team starts the play at a good pace, even if it looses the match in the end, the model will be penalize for predicting high win probability in the early stages of the game which is unintuitive. In our work, we try to work with soft labels which are calculated using current score and DLS par score which give a better understand of team's performance and win probability. These labels are calculated for ball by ball basis.

Here we calculate χ^{DLS} (win probability using DLS) as follows:

$$\begin{aligned}
score_diff &= (DLS\ par\ score - current\ score) \\
(DLS\ par\ score <= current\ score) &\implies \chi^{DLS} = 1 \\
(cutoff\ score) <= (score_diff) &\implies \chi^{DLS} = 0 \\
0 < (score_diff) < (cutoff\ score) &\implies \chi^{DLS} = \frac{\ln\left(3 - \left(\frac{2 * score_diff}{(cutoff\ score)}\right)\right)}{\ln(3)}
\end{aligned}$$

It is quite intuitive that if the current score is more than the expected DLS par score, then the win probability of the team should be maximize (meaning 1.0). Here we have used a hyper-parameter named `cutoff score` which represents the maximum tolerable difference allowed between the DLS par score and the current score. If the score difference (DLS par score - current score) gets more than the cutoff score, the probability is clipped to minimum (meaning 0.0). Now, if the score difference is less than cutoff score, we follow a logarithm decay for probability, with slow decay for small values of difference (team gets begin for an over or so, can make up in next overs) and high decay for large values of difference (if approaching near cutoff score, the probability of winning should decay faster to zero). The logarithm function used here is chosen by going through multiple similar curves (sigmoid, exponential, etc.) and checking their curvature.

Considering that DLS par score is the ideal score the team should have to win, we use `cutoff score = (40 runs)` for our model. The χ^{DLS} curve for model is shown in Figure 1.

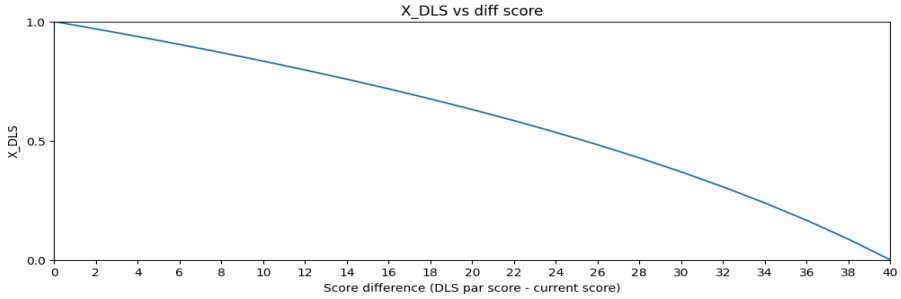

Figure 1: χ^{DLS} plot with `cutoff score = 40`

Now, for calculation of the final soft labels:

$$soft\ label = (\alpha * (hard\ label)) + ((1 - \alpha) * \chi^{DLS})$$

Here hard label means the final win/loss outcome and α is a hyper-parameter which decides the weightage to give to the hard label vs the DLS probability. High value of α means training the model with more focus on predicting the final label (hard label). Low value of α means training the model with more focus on predicting the DLS probability (soft label).

4.3.3 Inclusion of Variable Alpha (α)

In our work additionally to the use of DLS probability (χ^{DLS}), we are using a variable value of α to make model focus more on χ^{DLS} during initial stages of the play and as the play progresses we shift our focus from χ^{DLS} to the hard label (final win/loss).

The α value increases linearly till a certain number of remaining balls are left say `rem_balls_param`, at this point the value of α is equal to 1, meaning further this point, the soft label = hard label, which is important as we want the model to predict the final hard label (win/loss) during the final overs of play without the influence of χ^{DLS}.

So the variable alpha is calculated as follows:

$$\alpha = min\left(\frac{(120 - (balls\ remaining))}{(120 - rem\_balls\_param)},\ \ 1\right)$$

In our model `rem_balls_param = last 24 balls = last 4 overs`.

Now, the final loss is calculated using `nn.CrossEntropyLoss(predicted label, soft label)`.

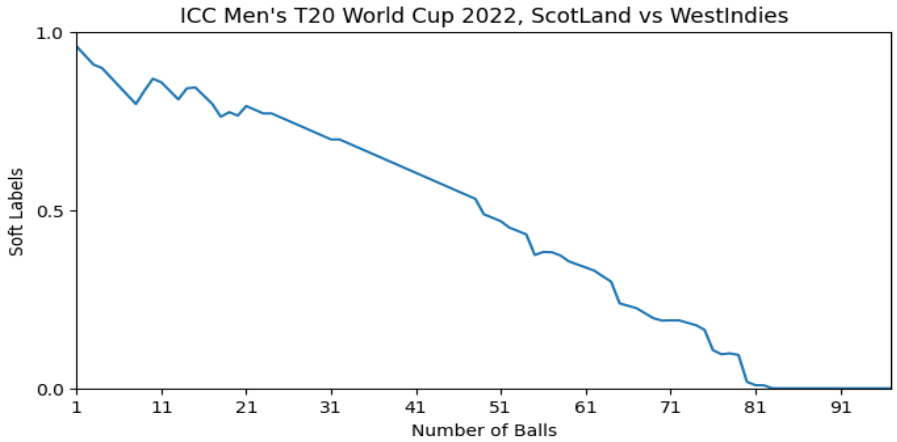

Figure 2: Soft Labels in Action: West Indies loses in 2^{nd} innings

5 Experiments and Results

Training details: As mentioned earlier, we have dataset of 2910 T20 international matches. We perform 80%-20% split of those matches to generate train and test splits respectively.

We train our model for 120 epochs using Adam optimizer with a learning rate of 0.001 and batch size of 16.

5.1 Model training comparison

We compare the training of the model based on the features given as input to the model.

We perform the experiments by changing the features provided as input to the model. At first we train our models by including the batter features along with the basic features. We include both the batter and the non-striker's features as both of them have contributions towards the final results. Following that, we performed check the model performance by adding the bowler's feature along with the match state information. The bowler has contribution towards having an effect of the bowling side to win the match. We take bowling average, strike rate and economy of the bowler.

These experiments model how the batters and bowlers individually have an effect towards deciding the result of the match. To model the entire game, we incorporate the features of both the batters and bowlers together and train the LSTM model.

We see that the performance of the model for predicting the result of the match is best while including the batter statistics into the model. When we include the features of the bowler along with the basic features, the performance decreases to a great extent. When we combine the features of both the batsmen

Feature set	Accuracy after 60 deliveries	Accuracy after 90 deliveries	Accuracy after 120 deliveries
Basic features + Batter features	86.29	91.73	96.37
Basic features + Bowler features	68.40	71.18	82.98
Basic features + Batter features + Bowler features	85.59	88.88	94.44

Table 1: Accuracy comparison with increasing number of training epochs and progress of the match

and the bowler, we obtain better score compared to that of model trained with bowler's score but less than that of batter's score individually. The results are described in Table **??**.

5.2 Effect of threshold on test accuracy

We have varied the threshold on predicted output values to determine the winning team. Accuracy vs. threshold curves are plotted for model trained with basic and batter features in Figure-3. Best results achieved are reported in Table-2.

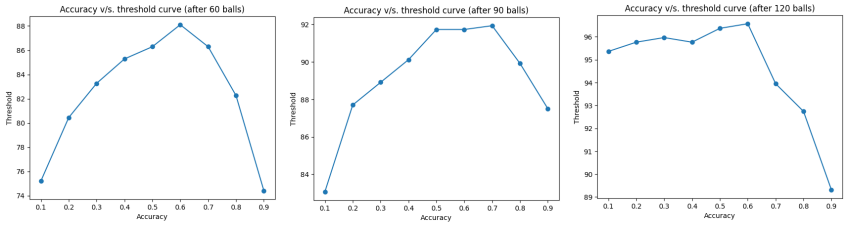

Figure 3: Accuracy vs. threshold curves.

	Best threshold	Accuracy (%)
After 60 deliveries	0.6	88.10
After 90 deliveries	0.7	91.93
After 120 deliveries	0.6	96.57

Table 2: Effect of threshold choice on test accuracy

5.3 Soft labels using DLS par score

Since we obtained highest score with the combination of batter statistics and the basic features, we apply the soft-labelling on top of it to make model robust. The DLS par score allows higher initial win percentages for teams which played

well in the initial stages of the play even if they lost the match, basically it doesn't penalize the model for it, which is correct, as in the end of the day, we are trying to predict the ball by ball win percentage.

	Accuracy (%)
After 60 deliveries	74.59
After 90 deliveries	85.08
After 120 deliveries	92.74

Table 3: DLS based soft-labeling along with Batter features

6 Conclusion

In this project, we trained a model for dynamic winner prediction. We explored different features e.g. match state-based basic features, batter features, and bowler features. We have also experimented with a novel Duckworth-Lewis method based loss function.

Our code is available in this github repository: `https://github.com/saikatdutta/CS725_Project_Team_Entrophy` .

References

[1] Viswanadha, S., Sivalenka, K., Jhawar, M. & Pudi, V. Dynamic Winner Prediction in Twenty20 Cricket: Based on Relative Team Strengths. *MLSA@PKDD/ECML*. (2017), https://api.semanticscholar.org/CorpusID:12828063

[2] Chakwate, R. & A, M. Analysing Long Short Term Memory Models for Cricket Match Outcome Prediction. *CoRR*. **abs/2011.02122** (2020), https://arxiv.org/abs/2011.02122

[3] CricSheet Downloads https://cricsheet.org/downloads/