An alternative for score prediction in rain interrupted cricket matches.

Abhiram Eswaran
Department of Computer Science
University of Massachusetts
Amherst, MA 01002
aeswaran@umass.edu
Akul Swamy
Department of Computer Science
University of Massachusetts
Amherst, MA 01002
aeswaran@umass.edu

May 3, 2017

Abstract

We propose an alternative to the state-of-the-art Duckworth-Lewis-Stern[1] method used to predict scores in a rain interrupted cricket match. Factors like average score at the venue and number of power-play balls remaining highly affect the performance of any team. The DLS method fails to take these factors into consideration. We use these along with DLS factors as our features. We then evaluate our model using different regression algorithms and compare the performance of our model and DLS Method with the ground-truth or the actual score of every match. We show that considering these additional features in the calculation of target score provides a better approximation of the predicted score.

1 Introduction

The DLS method was introduced as the standard way to predicting target scores in a rain interrupted cricket match in 1997, which was renamed to DLS in 2004. Over the years the game of cricket has evolved with higher scores being scored and chased down. While the game has evolved, the DLS method has been almost the same. The method heavily depends on 2 features, the wickets at hand and the overs left. The method has been criticized because wickets are a more heavily weighted resource than overs.[1] Another shortcoming of the method is that the DLS method does not account for changes in proportion of the innings for which field restrictions are in place compared to a completed match. In short, the DLS Method has failed to adapt to the evolved game of modern cricket.

In our project, we came up with a predictive model which considers features like wickets lost, overs left, power play balls left, target score, venue average and total overs of the game, thereby giving an unbiased prediction of the target score for a team. By considering the historical statistics of the venue and the field restrictions, we solve the existing shortcomings of the DLS method.

We built a model using the above mentioned features along with the ML regression algorithms like Linear Regression, Decision Trees, Logistic Regression, KNN Regressor and ensemble methods like Random Forests, Gradient Boosting and AdaBoost.

Since score prediction is different for 1st and 2nd innings, we consider 2 scenarios to evaluate our model. The first being, rain interrupting 1st innings and the innings being curtailed at the end. In the 2nd innings, we consider where rain interrupts and the match is abandoned. To measure our success, we propose two evaluation metrics,

- The RMSE of our model with baseline score and RMSE of DLS method with baseline score.
- The RMSE of the model when considering only DLS features and RMSE of the model when considering all features.

We found that the model outperformed the DLS-method for the first innings with Decision Trees performing the best. We also found that the model performed with the mentioned features rather than considering overs and wickets as done with the DLS-method. The findings underpin the fact that using additional features like average score of the venue and power play balls left, are critical in giving a better approximation of the target to be chased.

2 Related Work

Duckworth-Lewis method [2] is the one which is widely accepted in this scenario. It takes into account, wickets and overs remaining as the only resources of a team. The general formula used to compute the target score is given by,

$$Team2'sparscore = Team1'sscore \times \frac{Team2'sresources}{Team1'sresources}$$

This however does not seem correct and might not always result in a fair score adjustment. Also, with the advent of T20 cricket and power plays, larger totals have been chased down with ease over the recent past in International cricket. The D/L methods fails to take this acceleration into account. We address this by considering power play overs as a feature in our model. Also, the venue plays a key role in determining the final score and we wish to investigate this by employing ML algorithms to make use of these features. In order to improve Duckworth Lewis Stern method, Mankad et al[10] discusses using an alternative to Duckworth-Lewis table by uses a Gibbs sampling scheme related to isotonic regression to provide a nonparametric resource table. This approach helps to reflect the relative run scoring resources available to the two teams, that is overs and wickets in combination. But the approach fails at considering other important factors affecting score prediction.

Sethuraman et al.[3] discusses using momentum, venue, player ratings and player history for predicting scores using a machine learning model. The project uses Correlation Based Subset Feature Selection to pick the relevant set of features and runs a linear regression model to predict the score of a cricket match. While momentum and venues have a positive impact, modeling on player history and player rating impacts the performance of the model negatively since form of a player is not quantifiable. We address these problems by considering just the average score in the venue as a feature. Another work on similar lines is from Kampakis et al. [4]. They perform ML based prediction of outcomes in T20 games in English County cricket. They consider relevant team and player based features in determining the performance. This is a classification problem where they predict the end result of the game, which can find relevant applications to betting. Ours on the other hand, is a regression problem where we try to predict the target for the second innings. Here, we avoid considering team specific features as our method is aimed at performing unbiased score adjustment in case of a rain interruption in a 50 over match.

Similar to score prediction, they has been related work in different sports that predict the outcome of the game. Albina Yezus et al. [5] is a paper which studies the methods of machine learning used in order to predict outcome of soccer matches. It uses features like match information, season information and table for each moment in the season and models it with machine learning algorithms like Random Forest and K nearest neighbors. The models achieves a prediction accuracy of 60%. On similar lines Liang et al. [6] predict scores in European football games by building a model that can give an accurate prediction on game results based on data of previous matches and relevant analysis. The model shows the ability of dealing with big data and beats bookmakers on game results using classification. The best performing model was found to be Logistic regression achieving an accuracy of 54.5%.

3 Dataset

The dataset was scraped off Cricinfo[7] using BeautifulSoup. The data contained the ball by ball details of all one day international matches starting from 2006. The number of test cases was around 30000 for 1500 matches for 1st innings and 2nd innings separately. The data scraped consisted of metadata of the match including the venue, umpires, result and day of the match. The ball by ball details contained the striker, non-striker, runs, extras, bowler for each ball. From the above data we picked relevant features like wickets, overs and venue. The average scores of the ground were scraped separately and linked with the venue. The following were the features considered from the scraped data,

- Balls played
- Wickets lost
- Cumulative runs scored till the ith ball
- Power play balls remaining
- Average score at the venue
- The total number of overs in the match

For the second innings we used the target the team has to chase as an additional feature. All the data was integer in type.

4 Methodology

Our project is an application project where we try to use ML algorithms in the domain of cricket to predict target scores in a rain interrupted game. As a part of the project, we built a pipeline which includes data collection and pre-processing, feature engineering, feature selection, hyperparameter optimization, model learning and regression.

4.1 Data collection and pre-processing

The data was scraped off Cricinfo[7] using Beautifulsoup as mentioned before. The data was obtained in a CSV format. The data contained some meta data for each match like ground and umpire information. Further, it contained information at the ball level. ie; The result of each ball played in the match was listed as one row which included details of runs scored, batsman, extras conceded, all of these just for that specific delivery. It was then processed using pandas[8] to extract only the necessary fields into a dataframe.

4.2 Feature Engineering

Once the necessary fields were extracted into a dataframe, multiple other features were engineered using these base features. ie; The raw feature was just the ball number and the runs conceded for the ball. Since, we need to capture the rate of scoring, we compute the cumulative result for each ball or the state of the game at the end of the ith ball. Further, the number of powerplay balls left at every state in the game is computed using the elapsed over information while processing each row. Additionally, we include the average ground score with the aforementioned features. Once all these are engineered, we obtain the final feature vector containing- balls, runs, wicket, groundaverage, ppballsleft and totalovers.

4.3 Feature Selection

On using the SelectKBest method from sklearn[9] to perform feature selection, the complete set of features was returned as the best set. This is not surprising as these indeed are features which heavily determine the scoring in a match and were hand engineered in the first place.

4.4 Hyperparameter Optimization

For hyperparameter optimization, we use sklearn's GridSearchCV method with default parameters. To this method, we pass the regressor object along with the parameters we wish to optimize. We then store the optimized object for later testing and hence avoid re-computation.

4.5 Learning

As a part of the learning process for this problem, we explored numerous different regressors. We further trained these models and evaluated the performance on the prediction task at hand. Below, we present to you the details of the models we employed in the process.

4.5.1 Linear Regression

Considering the fact that ours is a regression problem, we chose Linear Regression as our first model. We chose this model owing to the simplicity and straightforwardness. Additionally, we set this up as our simple baseline upon which set an upper bound on the RMSE. Mathematically speaking, Linear Regression tries to fit the best straight line through the set of data points in n dimensions. The equation for the model is as shown below,

$$y_i = \beta_0 1 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^\top \beta + \varepsilon_i, \qquad i = 1, \ldots, n$$

where β_i is the weight on the i^{th} feature.

4.5.2 Decision Trees

The next model we explored was Decision Trees. We chose to explore this model due to couple of strong reasons. Firstly, the intuition behind the model's approach to learning is easy to comprehend. Secondly, the prediction time in case of Decision Trees is quick owing to the structure of the model. On a high level, the way the learning proceeds is by choosing the best attribute and the best threshold for splitting. The best is the one which minimizes the residual sum of squares. How long this process continues can be controlled using the depth or the number of data points necessary at a node for it to be split.

4.5.3 Random forests

Moving ahead, we switched over to experimenting with ensemble models owing to their better fitting and generalization accuracy. Random forests are essentially multiple decision trees which are built by choosing a subset of samples every time and again inducing more de-correlation by using a random subset of features at every split. This way, the trees are built and the relationships are learned. The results of all the trees are then combined via averaging, majority voting etc;

4.5.4 Comparison

Comparing the above mentioned 3 models, the linear regression model, though simple and straightforward, fails to capture complex non-linear relationships and hence fails to perform well in complex scenarios like this one. However, decision tree is much better in such cases where it can capture complex decision boundaries and also deal with missing values in data, which we have in ours. Since, it is a model of high capacity, it might also model noise and lead to overfitting. To avoid this, we switch to ensembles which strike a good balance between variance and bias, thereby achieving greater accuracy than other models. Hence, the comparison between the above 3 models.

4.5.5 Other regressors

Apart from the above mentioned regressors, we explored other regressors including lasso, gradient boosting regressor and KNN. KNN is a non-parametric model which memorizes the vector space and derives results using the results of neighborhood data points. Lasso is a method where linear regression with l1 regularization is used which performs feature selection as well as learning. It tries to drive only some weights to zero thereby giving less importance to those features. The gradient boosting regressor is one which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. AdaBoost, Logistic regression and support vector regression were also tried in the modeling.

4.6 Testing

The testing procedure was pretty straightforward for all the models. We selected the best performing model parameters using GridSearchCV and then ran prediction on a held out test set. The RMSE of this prediction with the ground truth was taken and compared with the RMSE of the DL method with the ground truth. We also compared the RMSE of the model with all features and features of DLS-Method. More details regarding these experiments and results are discussed in the next two sections.

5 Experiments and Results

The dataset scraped was split into test and training set in 80:20 ratio. The test set was held out for final evaluation. The training set was used with GridSearch to tune the hyperparameters for individual models.

5.1 Layout of the pipeline

The layout of the pipeline started with Hyperparameter optimization with GridSearch and then using the best hyperparameters obtained for Model Learning. We fit the model with the best hyperparameters obtained, obtain the training error and save the fitted model as a pickle to avoid repeated training. We also eliminated feature selection from the pipeline since the best number of features returned with Statistical Dependence Filtering was the maximum number of features in the dataset.

5.2 Hyperparamter Optimization

We used GridsearchCV to optimize our hyperparameters. We present the list of hyperparameters tuned for each of the models below.

- Lasso- alpha
- Decision Tree- min_samples_split, max_depth, min_samples_leaf, max_leaf_nodes
- Logistic Regression- C
- SVR- kernel, C, gamma
- Random Forests- n_estimators, max_features, max_depth, min_samples_split, bootstrap
- Gradient Boosting- learning_rate, max_depth
- AdaBoost- n_estimators

5.3 Training

The main objective of the project was to prove that having a linear dependency, like the DLS-method between the predicted score and the features would give poorer results. So we set Linear Regression as our baseline method. Every other model trained yielded better results than Linear Regression.

Using the best hyperparameters obtained from GridSearch we fitted the model on the training set and obtained the training error on the validation set. We did this for the two scenarios, for rain interruption in 1st innings and rain interruption in the 2nd innings. The training errors for the scenarios are provided in Table 1. It was observed that Decision Trees and Gradient Boosting gave the minimum training errors for 1st innings and 2nd innings respectively. Figure 1 and Figure 2 shows the RMSE training errors for each of the regressors selected.

In terms of speed, Linear Regression and Decision Trees took the minimum to train. Due to the number of hyper-parameters involved SVR took the highest time in hyperparameter optimization.

Method	RMSE for 1st innings	RMSE for 2nd innings
Baseline(Linear Regression)	24.503	21.715
Lasso	24.406	21.506
Logistic Regression	23.945	19.301
SVR	23.633	18.768
Decision Trees	**22.699**	5.911
Random Forests	22.763	5.166
Gradient Boosting	22.913	**4.519**
Adaboost	23.168	16.847

Table 1: Training Errors for 1st innings

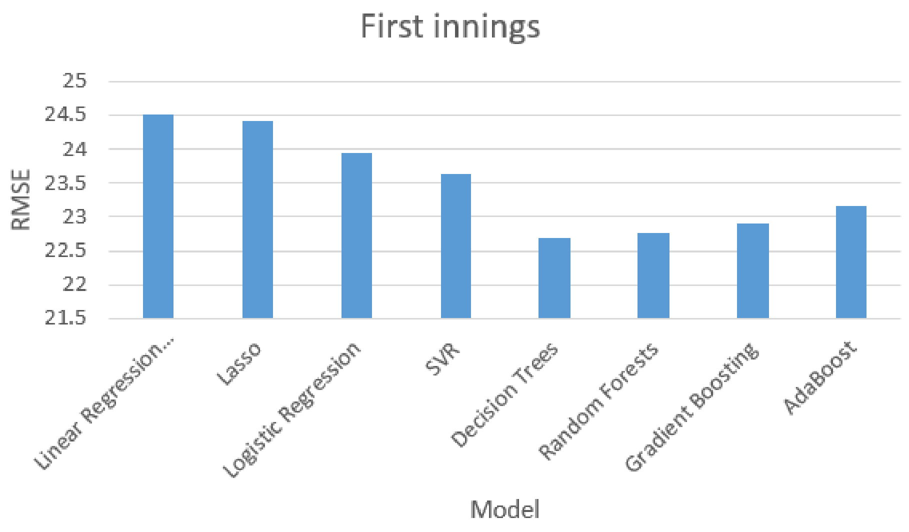

Figure 1: Training RMSE for 1st innings

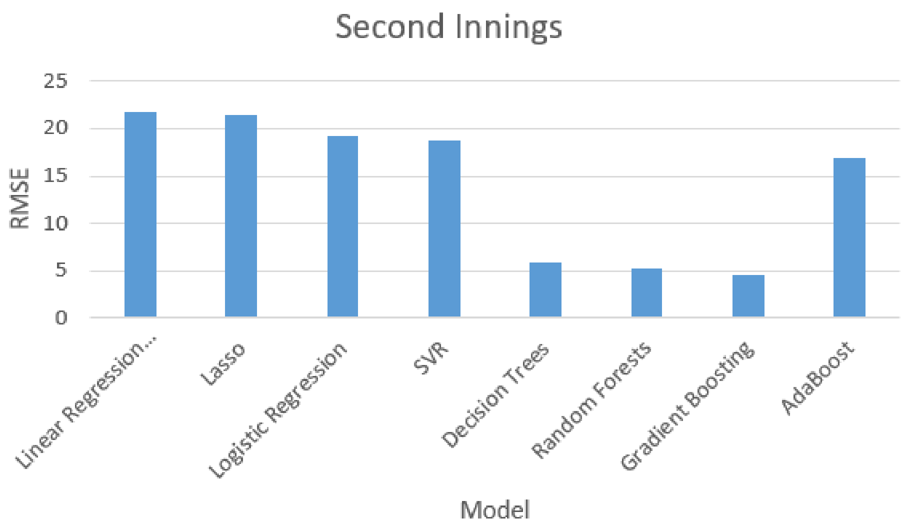

Figure 2: Training RMSE for 2nd innings

Innings	RMSE for ML Prediction	RMSE for DLS Method
1st innings	**34.335**	36.073
2nd innings	36.273	**22.392**

Table 2: Comparision of ML Prediction Model with the state of the art DLS-Method. ie; Test RMSE

Innings	RMSE with all features	RMSE with DLS Features
1st innings	**34.335**	36.616
2nd innings	**36.273**	37.2674

Table 3: Comparision of ML Prediction Model with all features and DLS features

5.4 Testing

From the above models we used the test set to predict scores for two scenarios,

- Rain interruption in 1st innings after the 1st team has played 40 overs and the innings was terminated.
- Rain interruption in 2nd innings after the 2nd team has played 40 overs and the match was abandoned.

To test the effectiveness of our predictions, we used the final scores scored by each team in the innings and scaled it down to 40 overs in both scenarios. This score would incorporate the ground truth of their actual score at 40 overs. We then use these scores, to compute the root mean square error of our predictions and root mean square of the DLS-method. Table 2 presents the performance of the best models against the DLS-Method for the 1st innings and 2nd innings. We discuss the importance of the results in the next section.

In Table 3 we also present the performance of the best models with all features and the features used by DLS Method in the computation of the predicted scores. It is clear from the results that using the suggested features gives better generalization performance. We discuss in depth in the next section.

In terms of speed decision trees had the best speed for prediction. In terms of generalization error Decision trees also gave the best performance for 1st innings scenario. Gradient Boosting performed the best for 2nd innings predictions.

6 Discussion and Conclusion

We set out with this project to prove the DLS-method of considering just 2 features for score prediction is archiac as the game as evolved. The results presented in Table 2 and Table 3 exactly reflect this.

Table 3 proves the hypotheses that it is important to consider more features, especially power play balls left and ground average into consideration to predict the score in a rain interrupted match. This would solve for the criticism DLS-method has perennially faced.

Table 2 validates the success of the Machine Learning model we used to predict scores and its comparison with the DLS Method. For 1st innings the best performing model was Decision Tree Regressor, this is attributed to the fact that Decision tree regressor uses the training data to pick variables and thresholds to optimize a local performance heuristic at each node in the tree. Further, since we are trying to predict scores at abrupt times of a match, there are chances that we would have not seen such values in training. This boils down to the problem of handling missing values and Decision Trees are pretty powerful in this regard. **Hence, our models gives a better generalization error than the state of the art DLS model for first innings.** The model for the 2nd innings fails because the 2nd innings models heavily on the data for the 2nd innings and just uses target as an additional feature to predict the score of the 2nd team. This is unlike what DLS method does. Hence the DLS is better for the 2nd innings.

In a nutshell, the project was a successful one. We achieved the core objective which showed the bias in DLS-Method with lesser features. The model we developed gave the best results than existing state of the art methods in the first innings. We got to test the performance of regression algorithms on the features we built and discovered decision trees and ensemble methods worked the best. We also found out that not having proper features would impact the model negatively as in the case of 2nd innings where the ML model failed compared to DLS Method. Future work for this project would include building a robust model for 2nd innings which would consider more features and give a better approximation of the score.

7 References

[1] The Duckworth Lewis factor *Srinivas Bhogle, March 06 2003*

[2] Duckworth-Lewis Method *Wikipedia*

[3] A Learning Algorithm for Prediction in the game of cricket *Sethuraman, Parameswaran Raman, Vijay Ramakrishna*

[4] Using Machine Learning to Predict the Outcome of English County twenty over Cricket Matches *Stylianos Kampakis, William Thomas, 2015*

[5] Predicting outcome of soccer matches using machine learning *Albina Yezus, Saint-Petersburg State University, 2014*

[6] Result Prediction for European Football Games *Xiaowei Liang, Zhuodi Liu and Rongqi Yan, 2015*

[7] ESPN Cricinfo *www.espncricinfo.com*

[8] Pandas *pandas.pydata.org*

[9] Sklearn *scikit-learn.org*

[10] Study and Analysis of Duckworth Lewis Method *Sapan H. Mankad, Anuj Chaudhary, Nikunj Dalsaniya, Vivek Mandir, Nirma University, 2014*