

# Predicting the Outcome of ODI Cricket Matches using Decision Trees and MLP Networks

Rajiv Kumar\*, Jalaz Kumar<sup>†</sup>

\*Assistant Professor, <sup>†</sup>Student

Department of Computer Science and Engineering

National Institute of Technology Hamirpur, India

Email: \*rajiv@nith.ac.in, <sup>†</sup>jalazkumar1208@gmail.com

**Abstract**—Applying Data Mining & Machine Learning in Sports Analytics is a blooming sector in the field of Computer Science. After Football, Cricket is the second most popular sports with a fan base of around 2.5 billion and mostly popular in South Asia, Australia, The Caribbeans and UK. It has tremendous spectator support and the masses show great interest in predicting the outcome of games. The result of a cricket match depends on lots of in-game and pre-game attributes. Pre-game attributes like venue, past track-records, Innings(First/Second), team strength etc. and in-game attributes like Toss, run rate, wickets in hand, strike rate etc. influence a match result predominantly. In this, we have used 2 different approaches, Decision Tree & MultiLayer Perceptron Network, to predict how these factors affect the outcome of an ODI cricket match. Based on the emerged results, We have designed CricAI: Cricket Match Outcome Prediction System. Our designed tool takes into consideration the pre-game attributes like venue(home,away,neutral),innings(first/second) & ground to & predict the outcome of given match.

**Keywords:** Decision Tree Classifier, MLP Classifier, Neural Networks

## I. INTRODUCTION

Cricket which is the world's second most popular sport after soccer, is basically a bat and ball game played between two teams of eleven players each. Each teams comes to bat and has a single inning in which it seeks to score as many runs as possible, while the other team fields. The innings ends when the total quota of deliveries, which depends on game format have turned up, or the 10 batsman has been dismissed, whichever comes first. The prime objective is to score more runs & thus Runs are the decisive factor.

Game of Cricket is a highly unpredictable in nature. Till the very last moment, it is difficult to make accurate predictions about the game. Various natural factors affecting the game output, huge betting market and enormous media coverage have given strong incentives to model this gentlemen's game from the Machine Learning perspective.

Rules of Cricket are determined by the International Cricket Council(ICC).

There are three internationally recognized formats of Cricket matches - Test match, ODI match(One Day International) and T20 match. The main difference between these three formats is the scheduled duration of the game which directly modifies the number of deliveries each team got to play in their respective innings.

Test cricket format is the longest one and is considered as the highest standard of game. Match duration is five days in which each team get to play 2 innings each. A standard test cricket day consist of 3 sessions of 2 hours each.

One Day International i.e. ODI format is of limited overs, where each team faces 300 deliveries(50 overs). ODI match is scheduled to complete in a Day or a Day/Night combination.

T20 is the shortest internationally recognized format of this game, where each team innings consist of 20 overs. This is more of an "explosive" and more "athletic" than the other two formats.

We focus our research on One Day Internationals, the most popular format of the game. Outcome of ODI match is influenced by a large no. of factors and can be predicted like all other games. We need to find the best attributes or factors that influence the match outcome. For our study we considered the factors analyzed by and ,which are proven to have a significant impact on outcome of ODI match. The factors considered for analysis include:

- **Teams Past Performance:** This factor captures the historic outcomes of all the matches played between them.
- **Ground:** This plays a vital role as teams have great track records on grounds and carry psychological superiority over the other.
- **Innings:** This factor determines which team batted first & which batted second.
- **Home Game Advantage:** This is achieved by using Venue feature, which determines whether a particular ground is home/away/neutral for each of the playing teams.

Both of our classification models are built using these factors. To predict the outcome of ODI matches we have applied two classification techniques - Decision Trees and Multi Layer Perceptron Networks. We have conducted comparative studies among various classifiers and summarized the results in this paper.

We then built a software tool called CricAI based on emerged results, which can be used to predict the outcome of any ODI match given the concerned factors as inputs. This software of ours can be of real value to the cricketers, support staff of teams and cricket analysts in terms of analyzing the future game in advance and working towards maximizing their chances of victory.

Clustering couldn't have made any contribution to our research as we dealt with multiple independent attributes, therefore placing them in clusters after finding similarity did not seem feasible.

The rest of this paper is organised as follows. Section 2 explains the approach we have used for conducting the analysis. Section 3 presents a comparative study of the classifiers used. Section 4 presents the related work in the area and the future scope associated with this approach. Section 5 gives the conclusions.

## II. APPROACH FOR ANALYSIS

### A. Data Collection

Data was extracted from [www.stats.cricinfo.com](http://www.stats.cricinfo.com) by running a scraping script in a justified manner, sending 1 request per second.

TABLE I  
SCRAPPED DATASET FORMAT

| Match Id | Team 1    | Team 2    | Winner    | Margin    | Ground     |
|----------|-----------|-----------|-----------|-----------|------------|
| ODI #1   | Australia | England   | Australia | 5 wickets | Melbourne  |
| ODI #2   | England   | Australia | England   | 6 wickets | Manchester |
| ODI #3   | England   | Australia | Australia | 5 wickets | Lord's     |

Dataset comprises of all the ODI matches from Jan 5,1971 to Oct 29,2017. A total of 3933 ODI match results was scrapped. The collected data was subjected to cleaning process where some of the matches were deleted from the analysis. Since the impact of the nature on the game of cricket cannot be foreseen, matches which were either interrupted by rain or ended up in a draw/tie, were being removed from the dataset. Matches of Special teams like World XI, Asia XI & Africa XI were also removed.

We also further replicated our dataset two times by swapping the team positions i.e. A game between Team 1: India and Team 2: Sri Lanka was also replicated as Team 1: Sri Lanka and Team 2: India. For further making the dataset suitable for Input to the various Machine Learning Classifier Models, We converted the continous dataset into a categorial dataset, using dummy variables.

**Innings Feature** was determined by first translating Column: *Margin* into Column: *Winner Innings* using:

- Win by Balls  $\implies$  Winner Innings: 2
- Win by Runs  $\implies$  Winner Innings: 1

Further, Using Column: *Winner* and the generated Column: *Winner Innings*, we aquired the innings of each team per match.

**Venue Feature** was determined by using Column: *Winner* and Scrapped dataframe from [www.stats.cricinfo.com](http://www.stats.cricinfo.com) giving the names of cricket grounds in all countries. Combining both of these, Column: *Host Country* was generated, which was used to get venue of a match with respect to both the teams.

The data set was saved in comma separated format. A total of 7494 match records were used in analysis. Finally, we

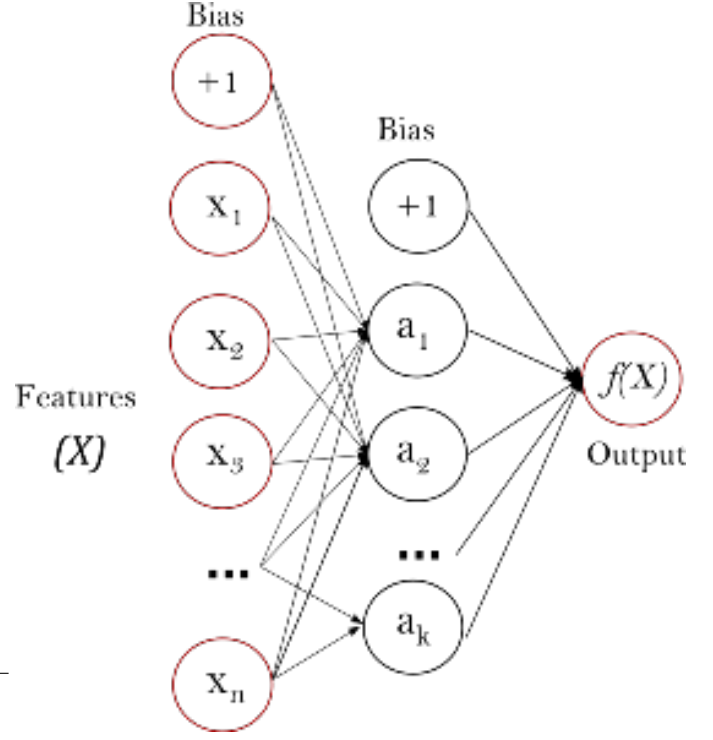


Fig. 1. Multilayer Perceptron Network

divided the dataset into two parts, namely, the test data and the training data.

★ Training Dataset Size: 5620

★ Testing Dataset Size: 1874

### B. MultiLayer Perceptron Networks

Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function

$$f(\cdot) : R^m \rightarrow R^o$$

by training on a dataset, where  $m$  is the number of dimensions for input and  $o$  is the number of dimensions for output. Given a set of features  $X = x_1, x_2, \dots, x_m$  and a target  $y$ , it can learn a non-linear function approximator for either classification or regression. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers. Fig 1. shows a Perceptron Network with 1 hidden layer.

The leftmost layer, known as the input layer, consists of a set of neurons

$$x_i | x_1, x_2, \dots, x_m$$

representing the input features. Each neuron in the hidden layer transforms the values from the previous layer with a weighted linear summation

$$w_1x_1 + w_2x_2 + \dots + w_mx_m$$

, followed by a non-linear activation function. The output layer receives the values from the last hidden layer and transforms them into output values.

MLPClassifier implements a multi-layer perceptron (MLP) algorithm that trains using Backpropagation. More precisely, it trains using some form of gradient descent and the gradients are calculated using Backpropagation.

MLP trains on two arrays: array  $\mathbf{X}$  of size (n\_samples, n\_features) which holds the training samples represented as floating point feature vectors; and array  $\mathbf{y}$  of size (n\_samples) which holds the target values (class labels) for the training samples.

Currently, MLPClassifier supports only the Cross-Entropy loss function, which allows probability estimates by running the predict\_proba method. Further, the model supports multi-label classification in which a sample can belong to more than one class which makes it quite suitable for our approach. For each class, the raw output passes through the logistic function. Values larger or equal to 0.5 are rounded to 1, otherwise to 0.

#### Advantages:

- \* It is capable to run non-linear models.
- \* Since, MLPClassifier uses Backpropagation so, it continuously learns and improvize itself.
- \* Capability to learn models in real-time using partial fitting.

#### Disadvantages:

- \* Highly sensitive to feature scaling.
- \* It uses a black box model, results may be more difficult to interpret.
- \* MLP requires tuning a number of hyperparameters such as the number of hidden neurons, layers, and iterations.

### C. Decision Trees

Decision Trees are a type of Supervised Machine Learning where the data is continuously split according to a certain parameter. The tree can be explained by two entities, namely decision nodes and leaves. The leaves are the decisions or the final outcomes and the decision nodes are where the data is split. Decision Trees (DTs) are used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

Given training vectors  $x_i \in R^n$ ,  $i=1, \dots, l$  and a label vector  $y \in R^l$ , a decision tree recursively partitions the space such that the samples with the same labels are grouped together. Let the data at node  $m$  be represented by  $Q$ . For each candidate split  $\theta (j, t_m)$  consisting of a feature  $j$  and threshold  $t_m$ , partition the data into  $Q_{left}(\theta)$  and  $Q_{right}(\theta)$  subsets as,

$$Q_{left}(\theta) = (x, y) | x_j \leq t_m$$

$$Q_{right}(\theta) = Q / Q_{left}(\theta)$$

The impurity at  $m$  is computed using an impurity function  $H()$ , the choice of which depends on the task being solved (classification or regression).

$$G(Q, \theta) = \frac{n_{left}}{N_m} H(Q_{left}(\theta)) + \frac{n_{right}}{N_m} H(Q_{right}(\theta))$$

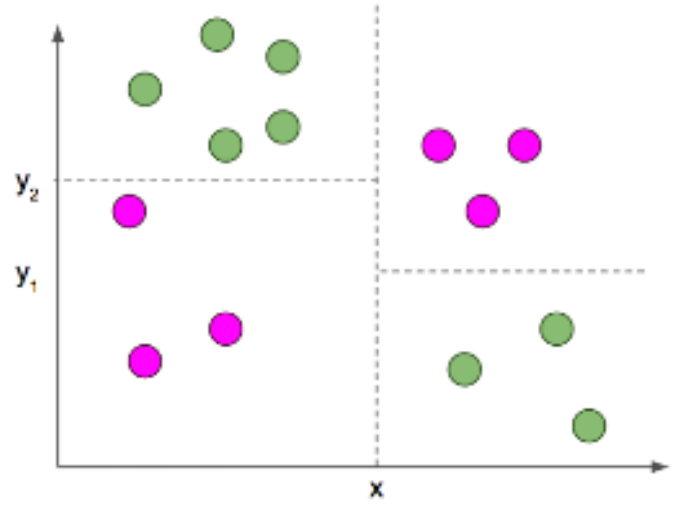


Fig. 2. Decision Tree

Select the parameters that minimises the impurity

$$\theta^* = \operatorname{argmin}_{\theta} G(Q, \theta)$$

Recurse for subsets  $Q_{left}(\theta^*)$  and  $Q_{right}(\theta^*)$  until the maximum allowable depth is reached,  $N_m < \min_{samples}$  or  $N_m=1$ .

#### Advantages:

- \* Quite Simple to understand, interpret and visualize.
- \* Able to handle both numeric as well as categorical data and also multi-output problems.
- \* Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by boolean logic.

#### Disadvantages:

- \* Creation of over-complex trees that do not generalize the data well. Overfitting is a problem in Decision Tree.
- \* Decision trees can be unstable because small variations in the data might result in a completely different tree being generated.
- \* For the cases, where some classes dominate creation of biased Decision Tree takes place.

## III. RESULTS AND OBSERVATIONS

### A. Performance Measures

### B. Comparative Analysis

## IV. RELATED WORK

## V. CONCLUSION

## VI. ACKNOWLEDGMENT

The preferred spelling of the word acknowledgment in America is without an e after the g. Avoid the stilted expression, One of us (R. B. G.) thanks . . . Instead, try R. B. G. thanks. Put sponsor acknowledgments in the unnumbered footnote on the first page.

References are important to the reader; therefore, each citation must be complete and correct. If at all possible, references should be commonly available publications.