Predicting Players' Performance in the Game of Cricket Using Machine Learning

by

Niravkumar Pandey

A thesis submitted in partial fulfillment

of the requirements for the degree of

Master of Science (MSc) in Computational Sciences

The Faculty of Graduate Studies

Laurentian University

Sudbury, Ontario, Canada

THESIS DEFENCE COMMITTEE/COMITÉ DE SOUTENANCE DE THÈSE
Laurentian Université/Université Laurentienne
Faculty of Graduate Studies/Faculté des études supérieures

Title of Thesis Titre de la thèse	Predicting Players' Performance in the Game of Cricket Using Machine Learning	
Name of Candidate Nom du candidat	Pandey, Niravkumar	
Degree Diplôme	Master of Science	
Department/Program Département/Programme	Computational Sciences	Date of Defence Date de la soutenance April 12, 2018

APPROVED/APPROUVÉ

Thesis Examiners/Examinateurs de thèse:

Dr. Kalpdrum Passi
(Supervisor/Directeur de thèse)

Dr. Ratvinder Grewal
(Committee member/Membre du comité)

Dr. Claude Vincent
(Committee member/Membre du comité)

Dr. Bhaba Krishna Mohanty
(External Examiner/Examinateur externe)

Approved for the Faculty of Graduate Studies
Approuvé pour la Faculté des études supérieures
Dr. David Lesbarrères
Monsieur David Lesbarrères
Dean, Faculty of Graduate Studies
Doyen, Faculté des études supérieures

ACCESSIBILITY CLAUSE AND PERMISSION TO USE

Abstract

Player selection is one of the most important tasks for any sport and cricket is no exception. The performance of the players depends on various factors such as the opposition team, the venue, his current form etc. The team management, the coach and the captain select eleven players for each match from a squad of 15 to 20 players. They analyze different characteristics and the statistics of the players to select the best playing 11 for each match. Each batsman contributes by scoring maximum runs possible and each bowler contributes by taking maximum wickets and conceding minimum runs. This thesis attempts to predict the performance of players as how many runs each batsman will score and how many wickets each bowler will take for both teams in one-day international cricket matches. Both the problems are targeted as classification problems where number of runs and number of wickets are classified in different ranges. We used Naïve Bayes, Random Forest, multiclass SVM and Decision Tree classifiers to generate the prediction models for both the problems. Random Forest classifier was found to be the most accurate for both problems.

Keywords

Cricket, One Day International (ODI), supervised learning, Naïve Bayes, Random Forest, multiclass SVM, Decision Trees, Oversampling

Acknowledgements

I would like to acknowledge my supervisor Dr. Kalpdrum Passi. I completed my thesis with his help. I found the research area, topic, and problem with his suggestions. He guided me with my study, and supplied me many research papers and academic resources in this area. He is patient and responsible. When I had questions and needed his help, he would always find time to meet and discuss with me no matter how busy he was. Also, he arranged a lot of meetings with Dr. Claude from School of Sports Administration who provided ideas.

In addition, I would like to acknowledge Dr. Claude Vincent, from School of Sports Administration, Laurentian University. He pointed me in the right direction whenever I needed any kind of help. He also helped me write my thesis in a more meaningful way.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1 Introduction

Selecting the best players for a particular match in any sport involves predicting the players' performance. Players' performance varies with the team they play against and the ground on which they play the match. Player selection is particularly more important in the game of cricket as the 11 players selected at the beginning of the match are fixed unless in case of injury. Moreover, the substituted players in such cases have limited privileges. Players' performance can be predicted by analyzing their past statistics and characteristics. Cricket players' abilities and performance can be measured in terms of different stats. Batsmen's statistics include batting average, batting strike rate, number of centuries etc. Whereas bowlers' statistics are measured by bowling average, bowling strike rate, economy rate etc. Other characteristics of batsmen include, batting hand of the batsman, the position at which the batsman bats etc. and those of bowlers include, the type of bowler, bowling hand of the bowler etc. Moreover, recent performances of the batsman/bowler, the performance of the batsman/bowler against a particular team and the performance of the batsman/bowler at a given venue are also taken into account for predicting his performance in the upcoming match. The team management, the coach and the captain utilize these facts and their own experience to select the team for a given match.

In this study, we used machine learning and data mining techniques to predict batsmen and bowlers' performances in a given day's match. We predict how many runs a batsman will score and how many wickets a bowler will take in the upcoming match. We targeted both the problems as classification problems where we classified runs and wickets into different ranges. We experimented with four supervised machine learning algorithms and compared their performance. The models generated by these algorithms can be used to predict the players' performance in future matches.

1.1 The Game of Cricket

Cricket is a sport played by two teams with each side having eleven players. One team bats and the other team bowls (fields) at a time and one such session is called an innings. In the center of the field, there is a 22-yard long pitch where most of the action takes place. Both ends of the pitch will have a wicket which has three wooden stumps and two cross pieces called the bails.

Figure 1 Cricket Pitch [1]

Each team consists of batsmen, bowlers and a wicket-keeper. All the players from the bowling team are on the field; one of them is behind the wickets, one of them bowls (throws the ball) from one end of the pitch and the other players are fielding, arranged in a particular fashion decided by the captain of the team. Two players from the batting team are on the field, alternating batting at a time. One of them bats from one end while the other one waits at the other end where the bowler is bowling from. The batsmen can be dismissed in many different ways with each ball bowled and this is called a wicket. As at a given time, there need to be exactly two batsmen on the field, the batting team has 10 wickets at the beginning of their innings. The batting team has to defend their wickets and score maximum runs possible and the bowling team has to get wickets as soon as possible and restrict the batting team from scoring runs. The team scoring the most runs wins at the end of the match.

The batsman on the opposite end of the bowler is called the striker. The striker takes guard on the popping crease which is four feet away in front of the wickets. The striker has to prevent the wickets from being hit by the ball by striking the ball hard with his bat. He tires to hit the ball well enough to score maximum runs on each delivery. Runs can be scored in two different ways. One way is to hit the ball hard enough for it to cross the boundary. If the batsman hits the ball into the air and the ball crosses the boundary before dropping on the ground, the batting team gets six runs, which is the maximum number of runs that can be scored on a legal delivery; otherwise the batting team gets four runs if the ball drops before crossing the boundary. Another way to score runs is by the two batsmen swapping ends running the length of the pitch in opposite directions while the fielders retrieve the ball.

The fielding team's role is to prevent the batsmen from scoring runs and dismiss them as soon as possible. A batsman can be dismissed in several ways. When the bowler hits the wickets directly with the ball and removes the bells from the stumps, the batsman is said to be bowled. When the batsman prevents the ball from hitting the stumps with his body, he is said to be dismissed as leg before wicket (lbw). If the striker leaves the popping crease and misses the ball and the wicket keeper removes the bells by hitting wickets with the ball, the batsman is dismissed as stumped. If the batsman hits the ball into the air and the ball is caught by a fielder without dropping on the ground, the type of dismissal is called caught. If a fielder retrieves the ball and removes bells from the stumps by hitting them with the ball, before the batsman reaches the crease while swapping ends to get a run, the batsman running towards the end where the bells have been removed, is said to be dismissed by run out. Any type of wicket except run out is said to be taken by the bowler who bowled the ball.

The bowling ends are swapped at the end of each over. An over consists of six balls bowled by a bowler. A different bowler comes in to bowl the next over. The number of balls may increase with illegal deliveries as wide balls or no balls which act as penalties against the bowling team in the form of an extra run and an extra ball for the batting team in that over. There are several ways in which a delivery can be declared a no ball or a wide ball. The umpires declare a delivery as a no ball or a wide ball according to those rules.

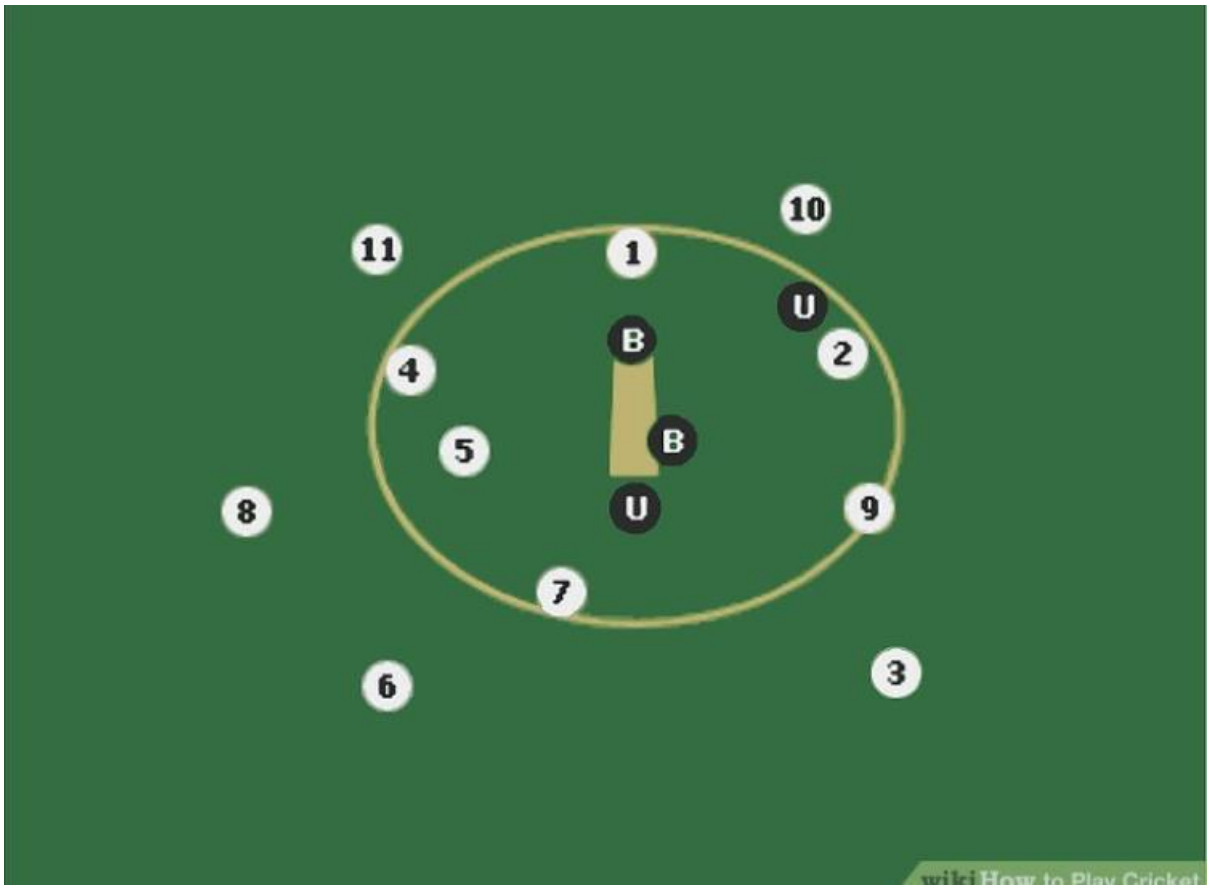

Figure 2 Player Positions in Cricket, B – Batsmen, U – Umpires, 7 – Bowler, 1 – 6 & 8 – 11 – Fielders [2]

1.2 Formats of the Game

Cricket is played in three different formats: one-day matches, T20 matches and test matches. One-day matches, also known as ODIs (One Day International) and T20 (twenty-twenty) matches are also known as limited overs cricket. In these formats there are two innings, so each team gets one chance to bat and one chance to bowl. In ODIs, a maximum of 50 overs can be bowled in one innings and in T20s, as the name suggests, a maximum of 20 overs can be bowled in one innings. So, an innings ends if 50/20 overs have been bowled or the batting team has lost 10 wickets. At the end of the first innings the teams change roles. The bowling team now bats and tries to chase the target set by the other team within 50/20 overs or before losing their 10 wickets. Similarly, the batting team now bowls and tries to prevent the other team from chasing the target down within 50/20 overs or by taking 10 wickets. Test matches on the other hand are played over a maximum of five days and each team can play up to two innings in a match.

Limited overs cricket is more challenging for both batsmen and bowlers. Batsmen need to score runs as fast as possible and the bowlers need to restrict them by conceding the least runs and taking wickets. The focus of this thesis is the ODI format which is the most popular format in international cricket nowadays. In this study, we are trying to predict how many runs a batsman will score and how many wickets a bowler will take in a given day's match.

1.3 Contributions

The principal contributions of this thesis are:

- We introduce a model that can quantify the performance of batsmen using their past statistics.
- We introduce a model that can quantify the performance of bowlers using their past statistics.
- We introduce four new measures based on raw attributes, that represent different aspects of both batsmen and bowlers' performance
- We compare the accuracies of different multiclass classification algorithms on our cricket dataset. This comparison can be used as a reference by selecting data classification algorithms for predicting players' performance.

1.4 Outline

The rest of the thesis is organized as follows:

Chapter 2 highlights some work related to the game of cricket.

Chapter 3 describes the data and the preprocessing that we did on the data. We describe the statistics and the attributes that are used to measure the players' performance. We also introduce some new attributes that we used in this study.

Chapter 4 gives a brief description of the machine learning algorithms that we used to create the prediction models.

Chapter 5 reveals the results of the experiments that we carried out on our data. We also discuss and compare the results and performances of different machine learning algorithms on our data.

Chapter 6 concludes the thesis and gives some directions for future work in the field.

Chapter 2

Literature Review

2 Related Work

An extensive online search produced very few articles related to players' performance prediction in the game of cricket. A very small number of researchers have tried to predict the performance of cricket players. Muthuswamy, S. and Lam, S. **[3]** predicted the performance of Indian bowlers against seven international teams against which the Indian cricket team plays most frequently. They used backpropagation network and radial basis function network to predict how many runs a bowler is likely to concede and how many wickets a bowler is likely to take in a given ODI match. Wikramasinghe, I. **[4]** predicted the performance of batsmen in a test series using a hierarchical linear model. Iyer, S. R. and Sharda, R. **[5]** used neural networks to predict the performance of players where they classify batsmen and bowlers separately in three categories – performer, moderate and failure. Then based on the number of times a player has received different ratings, they recommend if the player should be included in the team to play World Cup 2007. Saikia, H. and Bhattacharjee, D. **[6]** classified all-rounders in four categories using Naïve Bayes classification: Performer, Batting All-rounder, Bowling-All-rounder and Under-performer. They used the data of 35 all-rounders who played in first three seasons of IPL to generate the classification model and used the model to predict the expected classes of six new all-rounders. Saikia et al. **[7]** predicted the performance of bowlers in IPL IV using artificial neural networks. They used the bowlers' performance measures from

ODI and T20I (T20 international) matches and the Combined Bowling Rate measure introduced by Lemmer, H. H. **[8]**.

A lot of work has been done to measure players' performance and rank them. These rankings can then be used to select players for matches and tournaments. Barr, G. D. I. and Kantor, B. S. **[5]** defined a criterion for comparing and selecting batsmen in limited overs cricket. They defined a new measure P(out) i.e. probability of getting out and used a two-dimensional graphical representation with Strike Rate on one axis and P(out) on another. Then they define a selection criterion based on P(out), strike rate and batting average of the batsmen. Lemmer, H.H. **[8]** defined a new measure called Combined Bowling Rate to measure the performance of bowlers. The Combined Bowling Rate is a combination of three traditional bowling measures: bowling average, strike rate and economy. Bhattacharjee, D. and Pahinkar, D. **[9]** used this Combined Bowling Rate to analyze the performance of bowlers in the Indian Premier League(IPL). They also determined other factors that affect the performance of bowlers and applied multiple regression model to identify the factors that are empirically responsible for the performance of bowlers. Mukharjee, S. **[10]** applied Social Network Analysis to rate batsmen and bowlers in a team performance. He generated a directed and weighted network of batsmen-bowlers using player-vs-player information available for test and ODI cricket. He also generated a network of batsmen and bowlers using the dismissal record of batsmen in the history of cricket. Shah, P. **[11]** also defined new measures to measure players' performance. The new measure for batsmen takes into account the quality of each bowler he is facing and the new measure for bowlers considers the quality of each batsman he is bowling to. The aggregate of individual performance of a batsman against each bowler is the total performance index of the batsman. Similarly, the aggregate of individual performance of a bowler against each batsman is the total performance index of the bowler. Parker, D., Burns, P. and Natarajan,

H. **[12]** defined a model for valuation of players for IPL auction. Their model considered factors like previous bidding price of the player, experience of the player, strike rate etc. Sharp et al. **[13]** used integer optimization to select a T20 team. They described methods for quantifying batsmen's performance based on their scoring abilities and bowlers' performance based on their wicket taking abilities. These measures were then used in an integer program that would select an optimal team of 11 players. Ahmed et al. **[14]** used evolutionary multi-objective optimization for cricket team selection. They used batting average and bowling average as a measure of performance for batsmen and bowlers. They redefined team selection as a bi-objective optimization problem and then used non-dominated sorting genetic algorithm for multi-objective genetic optimization over the team. Omkar, S.N. and Verma, R. **[15]** used genetic algorithms for selecting a team. They defined the fitness of a team by considering the individual fitness of each player on the team. The fitness of a player is calculated based on his performance in batting, bowling, wicket-keeping, fielding, his physical fitness and his experience in the game. They also considered the team's performance against a particular team, on a particular pitch and the recent performance of the team. Then they used the genetic algorithm by representing the team as a string where each string bit represented a player. Lewis, A. J. **[16]** defined new measures of players' performance in ODIs using the Duckworth-Lewis method. Kimber, A. and Hansford, A. **[17]** carried out a statistical analysis of batting in cricket. They investigated the properties of batting average and stated that the traditional formula of batting average depends on unrealistic parameters. They defined an alternative parameter-free formula to calculate the batting average.

Another application of predictive analytics in cricket is to predict the winning team for a match or tournament. There are different approaches to achieve this. One approach would be to rank and compare players of different teams. Another approach would be to use other match-related

factors that affect the players' performance as the entire team. Jhanwar, M. and Paudi, V. **[7]** predict the outcome of a cricket match by comparing the strengths of the two teams. For this, they measured the performances of individual players of each team. They developed algorithms to model the performances of batsmen and bowlers where they determine the potential of a player by examining his career performance and then his recent performances. Prakash C. D., Patvardhan, C. and Lakshmi, C. V. **[13]** defined batting index and bowling index to rank players' performance for their models to predict outcomes of IPL matches. Ovens M. and Bukiet B. **[14]** applied a mathematical approach to suggest optimal batting orders for ODI matches. The Duckworth-Lewis method was introduced by Duckworth and Lewis **[22]** as a fair method to reset the target in interrupted ODI matches. Sankaranarayanan et al. **[19]** used data mining techniques to model and predict ODI matches. They used historical match data such as average runs scored by the team in an innings, average number of wickets lost by the team in an innings etc. and instantaneous match data such as whether the batting team is playing at the home ground or away or at a neutral venue, performance features of the two batsmen playing at the moment etc. to model the state of the match. Then they predict the outcome of the match by using machine learning algorithms such as linear regression and nearest-neighbors clustering algorithms. Swartz et al. **[22]** modelled and simulated ODI matches to predict the outcome of each ball that is bowled. They used historic data from past ODI matches to estimate the probability of each possible outcome. The probabilities depend on the batsman, the bowler, the number of wickets lost, the number of balls bowled and the innings.

Our work is probably the first generalized approach to predict how many runs will a batsman score and how many wickets will a player take on a particular match day. Muthuswamy and Lam [3] carried out a similar study predicting how many wickets will a bowler take using neural networks but their work was limited to eight Indian bowlers and is difficult to generalize

for all bowlers in the world. We did a more detailed study where we derived our own measures to capture different aspects of players' performance. No article in the literature describes such attributes. The literature guided us in selecting some of the input attributes that affect players' performance. We used some supervised machine learning algorithms to build prediction models that can be used to predict the performance of any player in a given match.

Chapter 3

Data and Preprocessing

3 The Data

We obtained all our data from www.cricinfo.com using scraping tools, parsehub **[30]** and import.io **[31]**. For batting, we considered matches played from 14 January 2005 to 10 July 2017. The senior most player during this span was SR Tendulkar, so we collected innings by innings list of the performance of all the batsmen from 18 December 1989 when he played his first ODI match. For bowling, we considered matches played form 2 January 2000 to 10 July 2017. The senior most player during this span was PA de Silva, so we collected innings by innings list of the performance of all the batsmen from 31 March 1984 when he played his first ODI match. Since the past stats of the players such as average, strike rate etc. are not available directly online for each match they played, we calculated from the innings by innings list for each match. We considered only those players who had played at least 10 innings till the match day. We had 25927 records for batsmen's data and 36230 records for bowlers' data. We imported all the data in MySQL tables and used php to manipulate them.

For predictive analytics, we used Weka **[32]** and Dataiku **[33]**. Both these tools are a collection of machine learning algorithms for data mining and also provide some preprocessing functionalities. All the results in this study have been obtained from Weka **[32]** 3-9-1-oracle-jvm and Dataiku Data Science Studio **[33]** on Mac OS 10.11.6 and Windows 10.

3.1 Player Statistics

Players' performance is measured in terms of several measures. The traditional measures that we used for measuring players' performance in this study are explained below in section 3.1.1 and 3.1.2. We derived four other measures Consistency, Form, Opposition and Venue using the traditional measures as will be explained in section 3.2.

3.1.1 Batting Measures

Innings: The number of innings in which the batsman has batted till the day of the match. This attribute measures the experience of the batsman. The more innings the batsman has played, the more experienced the player is.

Batting Average: Batting average commonly referred to as average is the average number of runs scored per innings. This attribute indicates the run scoring capability of the player.

$$Average = \frac{Runs\ Scored}{Number\ of\ Innings\ Played}$$

Strike Rate (SR): Strike rate is the average number of runs scored per 100 balls faced. In limited overs cricket, it is important to score runs at a fast pace. More runs scored at a slow pace is rather harmful to the team as they have a limited number of overs. This attribute indicates how quickly the batsman can score runs.

$$Strike\ Rate = \frac{Runs\ Scored}{Number\ of\ Balls\ Faced} \times 100$$

Centuries: Number of innings in which the batsman scored more than 100 runs. This attribute indicates the capability of the player to play longer innings and score more runs.

Fifties: Number of innings in which the batsman scored more than 50 runs (but less than 100). This attribute indicates the capability of the player to play longer innings and score more runs.

Zeros: Number of innings in which the batsman was dismissed without scoring a single run.

Highest Score (HS): The highest runs scored by a batsman in any (single) innings throughout his career. This attribute is used in the formula for calculating the venue attribute. This attribute shows the run scoring capability of the batsman at the venue. If a player has s very high score at a venue in past, he is more likely to score more runs at that venue.

3.1.2 Bowling Measures

Innings: The number of innings in which the bowler bowled at least one ball. It represents the bowling experience of a player. The more innings the player has played, the more experienced the player is.

Overs: The number of overs bowled by a bowler. This attribute also indicates the experience of the bowler. The more overs the bowler has bowled, the more experienced the bowler is.

Bowling Average: Bowling average is the number of runs conceded by a bowler per wicket taken. This attribute indicates the capabilities of the bowler to restrict the batsmen from scoring runs and taking wickets at the same time. Lower values of bowling average indicate more capabilities.

$$\text{Bowling Average} = \frac{\text{Number of Runs Conceded}}{\text{Number of Wickets Taken}}$$

Bowling Strike Rate: Bowling strike rate is the number of balls bowled per wicket taken. This attribute indicates the wicket taking capability of the bowler. Lower values mean that the bowler is capable of taking wickets quickly.

$$\text{Strike Rate} = \frac{\text{Number of Balls Bowled}}{\text{Number of Wickets Taken}}$$

Four/Five Wicket Haul: Number of innings in which the bowler has taken more than four wickets. This attribute indicates the capability of the bowler to take more wickets in an innings. Higher the value, more capable the player.

3.2 Data Preprocessing

3.2.1 Calculating the Weights

As we saw, different measures highlight different aspects of a player's abilities and hence some

measures have more importance than others, e.g. batting average is an important factor for all the formats of the game as it reflects the run scoring abilities of a batsman in general. Similarly, strike rate would be an important factor for limited over matches as it is important to score more runs in limited overs. So, we weighted each measure of performance according to its relative importance over other measures. We determined the weights using analytic hierarchy process(AHP) [34] [35]. AHP is an effective tool for complex decision making. It aids in setting priorities and making the best decision. AHP reduces complex decisions into a series of pairwise comparisons. AHP captures both subjective and objective aspects of a decision. The AHP generates a weight for each evaluation criterion according to the decision maker's pairwise comparisons of the criteria. The higher the weight, the more important the corresponding criterion. Next, for a fixed criterion, the AHP assigns a score to each option according to the decision maker's pairwise comparisons of the options based on that criterion. The higher the score, the better the performance of the option with respect to the considered criterion. Finally, the AHP combines the criteria weights and the options scores, thus determining a global score for each option, and a consequent ranking. The global score for a given option is a weighted sum of the scores it obtained with respect to all the criteria.

The analytic hierarchy process decomposes the decision making process in following steps: [35]

1. Define the problem and the knowledge sought.
2. Structure the decision hierarchy with the goal at the top level, objectives/attributes at the intermediate levels and alternatives at the lowest level.
3. Construct a set of pairwise comparison matrices where each element in an upper level is compared to the elements in the level immediately below it. These comparisons are made using a scale of numbers which indicates how many times more important is one element over another. This scale is tabulated in table 1 below.

4. The priorities obtained from the comparisons are used to weigh the priorities in the level immediately below. This is done for every element. The overall or global priority for every element in the level below is obtained by adding its weighted values. This process is continued until priorities of the alternatives in the lowest level obtained. The weights are calculated using some mathematical operations.

Table 1 Relative importance levels of objectives/attributes [35]

Level of Importance	Meaning	Description
1	Equal Importance	Two activities contribute equally to the objective.
2	Weak or Slight	
3	Moderate Importance	Experience and judgement slightly favor one activity over another.
4	Moderate Plus	
5	Strong Importance	Experience and judgement strongly favor one activity over another
6	Strong Plus	
7	Very strong or demonstrated importance	An activity is favored very strongly over another; its dominance demonstrated in practice
8	Very, very strong	
9	Extreme importance	The evidence favoring one activity over another

		is of the highest possible order of affirmation
Reciprocals of the above	If activity *i* has one of the above non-zero numbers assigned to it when compared with activity *j*, then *j* has the reciprocal value when compared with *i*	A reasonable assumption
1.1 – 1.9	If the activities are very close	May be difficult to assign the best value but when compared with other contrasting activities the size of the small numbers would not be too noticeable, yet they can still indicate the relative importance of the activities.

Following is an example of how AHP can be used to determine weights of the attributes in our study. Here we determine weights of the traditional batting performance measures to calculate the new attributes.

First, using our knowledge of cricket statistics and experience, we arrange the attributes in their decreasing order of importance as:

Average > Innings > Strike Rate > Centuries > Fifties > Zeros

Next, we create a matrix to compare their importance using table 1.

Table 2 Pairwise comparison of the attributes

	Average	Innings	Strike Rate	Centuries	Fifties	Zeros
Average	1	3	4	5	6	7
Innings	$\frac{1}{3}$	1	3	4	5	6
Strike Rate	$\frac{1}{4}$	$\frac{1}{3}$	1	3	4	5
Centuries	$\frac{1}{5}$	$\frac{1}{4}$	$\frac{1}{3}$	1	2	3
Fifties	$\frac{1}{6}$	$\frac{1}{5}$	$\frac{1}{4}$	$\frac{1}{2}$	1	3
Zeros	$\frac{1}{7}$	$\frac{1}{6}$	$\frac{1}{5}$	$\frac{1}{3}$	$\frac{1}{3}$	1

Next, we calculate the weight of each attribute. First, we calculate the priority of each attribute using the formula:

$$P_j = \left(\prod_{j=1}^{N} p_{ij} \right)^{\frac{1}{N}}$$

where; P_j is the priority of attribute j, N is the number of attributes and p_{ij} is the level of importance of attribute j over attribute i. Next, we normalize each attribute's priority using the following formula:

$$W_j = \frac{P_j}{\sum_{i=1}^{N} P_i}$$

Finally, we get following weights for the attributes:

Average: 0.4262

Innings: 0.2566

Strike Rate: 0.1510

Centuries: 0.0787

Fifties: 0.0566

Zeros: 0.0328

3.2.2 New Attributes

To predict a player's performance, his past performances need to be analyzed in terms of how much experience he has, how consistent he has been in his performance, how well he has been performing in recent matches, how well can he tackle the bowlers/batsmen of different teams, how well does he play at different venues, etc. Traditional measures of players' performance cannot reflect these factors directly. So, we tried to reflect and quantify them by deriving four new measures from the traditional measures. These attributes are weighted averages of the traditional attributes. These attributes are explained as follows:

Consistency: This attribute represents how experienced and consistent the player is. It is the weighted average of the traditional attributes calculated over the player's entire career. Its formula is as follows:

For batting,

$$\boldsymbol{Consistency} = \mathbf{0.4262} \times \boldsymbol{Average} + \mathbf{0.2566} \times \boldsymbol{Innings} + \mathbf{0.1510} \times \boldsymbol{Strike\ Rate} + \mathbf{0.0787} \times \boldsymbol{Centuries} + \mathbf{0.0556} \times \boldsymbol{Fifties} - \mathbf{0.0328} \times \boldsymbol{Zeros}$$

For bowling,

$$\boldsymbol{Consistency} = \mathbf{0.4174} \times \boldsymbol{Overs} + \mathbf{0.2634} \times \boldsymbol{Innings} + \mathbf{0.1602} \times \boldsymbol{Strike\ Rate} + \mathbf{0.0975} \times \boldsymbol{Bowling\ Average} + \mathbf{0.0615} \times \boldsymbol{Four/Five\ Wickets\ Haul}$$

Form: This attribute represents the player's current form. It quantifies the player's performance over past twelve months.

For batting,

$$Form = 0.4262 \times Average + 0.2566 \times Innings + 0.1510 \times Strike\ Rate + 0.0787 \times Centuries + 0.0556 \times Fifties - 0.0328 \times Zeros$$

For bowling,

$$Form = 0.3269 \times Overs + 0.2846 \times Innings + 0.1877 \times Strike\ Rate + 0.1270 \times Bowling\ Average + 0.0789 \times Four/Five\ Wickets\ Haul$$

Opposition: This attribute represents the player's performance against the team with which the match is being played.

For batting,

$$Opposition = 0.4262 \times Average + 0.2566 \times Innings + 0.1510 \times Strike\ Rate + 0.0787 \times Centuries + 0.0556 \times Fifties - 0.0328 \times Zeros$$

For bowling,

$$Opposition = 0.3177 \times Overs + 0.3177 \times Innings + 0.1933 \times Strike\ Rate + 0.1465 \times Bowling\ Average + 0.0943 \times Four/Five\ Wickets\ Haul$$

Venue: This attribute represents the player's performance at the ground at which the match is being played.

For batting,

$$Venue = 0.4262 \times Average + 0.2566 \times Innings + 0.1510 \times Strike\ Rate + 0.0787 \times Centuries + 0.0556 \times Fifties - 0.0328 \times Zeros$$

For bowling,

$$Venue = 0.3018 \times Overs + 0.2783 \times Innings + 0.1836 \times Strike\ Rate + 0.1391 \times Bowling\ Average + 0.0972 \times Four/Five\ Wickets\ Haul$$

3.2.3 Rating the Traditional Measures

The values of the traditional attributes fall in very wide ranges and small differences in these values do not discriminate different players, e.g. batsmen having batting averages of 32.00, 35.50 and 38.60 are considered to be of same quality. So, we rated each traditional measure from 1 to 5 based on the range in which its value falls, to calculate the derived attributes, with 1 being the minimum and 5 being the maximum. We looked at the values of these attributes for different players and applied our knowledge to rate the measures, e.g. some of the best batsmen of the world have had batting averages greater than or equal to 40 for most of the time during their career and generally, averages greater than or equal to 40 are considered excellent, so we rated such batsmen 5 for averages greater than 39.99. We used these ratings instead of actual values of the measures, in the formulae of derived attributes. The measures are rated as follows:

No. of Innings:

For Consistency:

1 - 49 : 1

50 - 99 : 2

100 - 124 : 3

125 - 149 : 4

>=150 : 5

For Form:

1 - 4 : 1

5 - 9 : 2

10 - 11 : 3

12 - 14 : 4

>=15 : 5

For Opposition:

1 - 2 : 1

3 - 4 : 2

5 - 6 : 3

7 - 9 : 4

>=10 : 5

For Venue:

1 : 1

2 : 2

3 : 3

4 : 4

>=5 : 5

Batting Average (for all derived attributes):

0.0 - 9.99 : 1

10.00 - 19.99 : 2

20.00 - 29.99 : 3

30.00 - 39.99 : 4

>=40 : 5

Batting Strike Rate (for all derived attributes):

0.0 - 49.99 : 1

50.00 - 59.99 : 2

60.00 - 79.99 : 3

80.00 - 100.00 : 4

>=100.00 : 5

Centuries:

For Consistency:

1 - 4 : 1

5 - 9 : 2

10 - 14 : 3

15 - 19 : 4

>=20 : 5

For Form:

1 : 1

2 : 2

3 : 3

4 : 4

>=5 : 5

For Opposition:

1 : 3

2 : 4

>=3 : 5

For Venue:

1 : 4

>=2 : 5

Fifties:

For Consistency:

1 - 9 : 1

10 - 19 : 2

20 - 29 : 3

30 - 39 : 4

>=40 : 5

For Form & Opposition:

1 - 2 : 1

3 - 4 : 2

5 - 6 : 3

7 – 9 : 4

>=10 : 5

For Venue:

1 : 4

>=2 : 5

Zeros:

For Consistency:

1 - 4 : 1

5 - 9 : 2

10 - 14 : 3

15 – 19 : 4

>=20 : 5

For Form & Opposition:

1 : 1

2 : 2

3 : 3

4 : 4

>=5 : 5

Highest Score (For Venue Only):

1 - 24 : 1

25 - 49 : 2

50 - 99 : 3

100 - 150 : 4

>=150 : 5

Overs:

For Consistency:

1 - 99 : 1

100 - 249 : 2

250 - 499 : 3

500 - 1000 : 4

>=1000 : 5

For Form & Opposition:

1 - 9 : 1

10 - 24 : 2

25 - 49 : 3

50 - 100 : 4

>=100 : 5

For Venue:

1 - 9 : 1

10 - 19 : 2

20 - 29 : 3

30 - 39 : 4

>=40 : 5

Bowling Average (for all derived attributes):

0.00 - 24.99 : 5

25.00 - 29.99 : 4

30.00 - 34.99 : 3

35.00 - 49.99 : 2

>=50.00 : 1

Bowling Strike Rate (for all derived attributes):

0.00 - 29.99 : 5

30.00 -39.99 : 4

40.00 -49.99 : 3

50.00 -59.99 : 2

>=60.00 : 1

Four/Five Wicket Haul:

For Consistency:

1 - 2 : 3

3 - 4 : 4

>=5 : 5

For Form, Opposition & Venue:

1 - 2 : 4

>=3 : 5

3.2.4 Other Input Attributes

Our experiments showed that the derived attributes themselves are sufficient to accurately predict players' performance. Also, there are some other factors apart from past performances that affect players' performances, e.g. depending on the types of bowlers the opposition team has, it would be better to include more left-handed batsmen than right-handed batsmen in the team or vice versa. So, we incorporated additional attributes which indicate the players', the opponents' and the venues' characteristics, in our experiments. These attributes are explained below:

Batting Hand: The dominant hand of the batsman while batting. It has two possible values: Left or Right. Depending on the characteristics of the bowlers of the opposition team, left-handed batsmen might perform better than the right-handed batsmen or vice versa.

Bowling Hand: The dominant hand of the batsman while bowling. It has two possible values: Left or Right. Depending on the characteristics of the opposition team's batsmen, left-handed bowlers might perform better than the right-handed bowlers or vice-versa.

Batting Position: The number at which the batsman bats in the batting order. Different batsmen tend to play better at certain numbers. So, sending a batsman at a particular number will make him more comfortable at play, e.g. M S Dhoni has been playing better at position 7 than other positions.

Match Type: The type of the match. This attribute has four possible values: Normal, quarter-final, semi-final or final. Different types of matches have different levels of importance which affects players' performance, e.g. final matches are more important than normal matches. Moreover, different players are more comfortable and have shown better performances in some types of matches, e.g. some players tend to play well in normal matches but fail in semi-finals and finals or vice versa.

Match Time: The time at which the match is played. There are two possible values: Day or Day-night. The time of the match also affects players' performance depending on different factors like weather, visibility, location etc.

Strength of opposition: This is the batting/bowling strength of the opposition team. It is the average of the consistency measure of the batsmen/bowlers of the opposition team. Players find it easy to score runs/take wickets against weaker teams than stronger teams.

Ven: The relative venue for the teams. It has three possible values: Home, Away or Neutral. The relative venue of the match is certainly a factor that affects players' performance. Some players perform better at home while some play better away from home.

Oppo: The opposition team. Players usually tend to perform better against some teams. This attribute also incorporates the characteristics of the opposition team's players in general.

Role: Playing role of the player. It can take following values:

Opening Batsman (OBT) – The two batsmen who usually bat at position one or two are called opening batsmen.

Top Order Batsman (TOB) – The batsmen who usually bat at position three or five are called top order batsmen.

Middle Order Batsman (MOB) - The batsmen who usually bat at position five to eight are called middle order batsmen.

Batsman – The batsmen who usually bat at different positions are categorized simply as batsmen here.

All-rounder – The players who are equally skilled at both batting and bowling are called all-rounders.

Batting All-rounder – The players who can both bat and bowl but are more skilled at batting than bowling, are called batting all-rounders.

Bowling All-rounder – The players who can both bat and bowl but are more skilled at bowling than batting, are called bowling all-rounders.

Bowler – The players who are expert bowlers but not so skilled at batting, are categorized as bowlers.

Captain: This is a binary attribute indicating whether a player is captain of the team. This attribute tries to indicate the control and responsibilities the player has. Some players perform well as captains while some perform worse.

WK: This is a binary attribute indicating whether a player is a wicketkeeper. Wicketkeepers are primarily batsmen. They are expected to score more runs as they specialize in batting and are less fatigued than other players as they are physically less active during fielding compared to other fielders.

Innings: This attribute indicates if it is the first or the second innings of the match. Depending on different factors like time of the match, the venue, the characteristics of the pitch, etc., sometimes it is more desirable to bat in the first innings while sometimes it is better to bowl in the first innings.

Tournament: The type of tournament in which the match is being played. Players feel different levels of pressure and go through psychological ups and downs during different types of tournaments. This attribute can take following values:

Two Team Tournament (TT)

Three-Four Team Tournament (TFT)

Five or more Team Tournament (FT)

Toss: Indicates whether the player's team won or lost the toss. Toss affects the mental state of the players as winning the toss gives them the power to decide whether to bat first or to bowl first and gives a strategic lead to the team.

Pressure: Indicates mental and psychological pressure on the player. It takes values from 1 to 5. Its value depends on the type of match being played and the teams that are playing the match. The values are defined as follows:

Normal matches: 1

Quarter Finals: 3

Semi Finals: 4

Finals: 5

Above values are incremented by 1 if the match is India vs Pakistan or Australia vs England as these countries are strong rivals of each other.

Host: The country in which the match is being played. Some players tend to perform better in certain countries as shown by their stats. This attribute also tries to incorporate the general nature of the pitches of different grounds in the country, e.g. Australian and South African venues are known to have bouncy pitches which are helpful to pace bowlers whereas pitches in India are usually dry and are more supportive to spin bowlers.

Ground: The ground on which the match is being played. The data about different pitches is not available at this time, so we tried to incorporate the general nature of the pitches at different grounds using this attribute. Also, players are more comfortable at some venues, e.g. a player who has had some world records at a particular ground, is more likely to perform better on that ground.

3.2.5 Data Cleaning

A large number of values of Opposition and Venue were zero. This is because a player has not played any match against a particular team or at a venue before the day of play. We treated such values as missing values and replaced them with the class average of corresponding attributes.

3.2.6 Oversampling

We observed that a majority of the records fall within class 1 in both batting and bowling. This created a major imbalance in the distribution of values and affected the performance of the learning algorithms. To solve this problem, we applied an oversampling technique Supervised Minority Oversampling Technique (SMOTE) **[36]** on minority classes to make all the classes equally distributed. SMOTE over-samples minority classes by creating synthetic example tuples. To create synthetic tuples of minority class, SMOTE takes each minority class sample and creates synthetic examples along the line segment joining any or all of its nearest neighbors. To generate a synthetic sample, the difference between the feature vector under consideration and its nearest neighbor is taken. This difference is then multiplied by a random number between zero and one and the product is added to the feature vector under consideration. This way, a random point along the line segment joining two specific features is selected. Neighbors from the k nearest neighbors are selected based on the amount of oversampling required. e.g. to oversample a minority class by 300%, three neighbors from a tuple's nearest neighbors are selected and one sample in the direction of each is generated.

3.2.7 Outputs

Both the problems are treated as classification problems.

Runs are predicted in five classes:

1 - 24: 1

25 - 49: 2

50 - 74: 3

75 - 99: 4

>=100: 5

Wickets are predicted in three classes:

0 - 1: 1

2 - 3: 2

>=4: 3

Chapter 4

Learning Algorithms

4 Supervised Learning

Supervised learning is a machine learning technique of deriving a function from a labeled training sample. A training sample is a set of training tuples. A training tuple consists of a set of input attributes and an associated output value. A supervised learning algorithm generates an inferred function by analyzing the training data. This function is then used to classify an unseen data. In predictive analytics, the generated function is called a predictive model. For our study, we used Naïve Bayes, Decision Tree, Random Forests and multiclass SVM to generate the prediction models.

4.1 Naïve Bayes Classifier

Bayesian classifiers are statistical classifiers that predict the probability with which a given tuple belongs to a particular class **[37]**. Naïve Bayes classifier assumes that each attribute has its own individual effect on the class label, independent of the values of other attributes. This is called class-conditional independence. Bayesian classifiers are based on Bayes' theorem.

4.1.1 Bayes' Theorem

Let X be a data tuple described by measurements made on a set of n attributes. Let H be a hypothesis such that X belongs to a specified class C. Bayesian classifiers calculate P(H|X), the probability with which the hypothesis H holds true for the observed attribute values of the data tuple X. P(H|X) is called the posterior probability or posteriori probability of H conditioned on X. Similarly, P(X|H) is the posterior probability or posteriori probability of X given H i.e. the probability with which the data tuple X exists, given the hypothesis H is true. P(H) is the prior probability or a priori probability of H which means that H holds true for a data tuple regardless of the values of its attributes. P(X) is the prior probability or a priori probability of X, which is the probability with which the data tuple X with given attribute values exists. Now, Bayes Theorem is defined as,

$$\boldsymbol{P(H|X)} = \frac{\boldsymbol{P(X|H)P(H)}}{\boldsymbol{P(X)}}$$

4.1.2 Naïve Bayes Classification

1. Let D be a training set of data tuples and their associated class labels, where each tuple is represented by an n-dimensional attribute vector, $X=(x_1, x_2, x_3,\ldots,x_n)$.

2. For a multiclass classification, suppose that there are m classes, $C_1, C_2, C_3,\ldots,C_m$. The Naïve Bayes classifier predicts that a given tuple X belongs to the class with the highest posterior probability conditioned on X. That is X belongs to class C_i if and only if

$$P(C_i|X) > P(C_j|X) \quad for\ 1 \leq j \leq m, j \neq i$$

Thus, we need to maximize $P(C_i|X)$. The class with maximum $P(C_i|X)$ is called the maximum posteriori hypothesis.

From Bayes theorem,

$$\boldsymbol{P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}}$$

As P(X) is constant for all classes, we need to maximize $\boldsymbol{P(X|C_i)P(C_i)}$. If the class prior probabilities are unknown, all the classes are assumed to be equally probable i.e. $P(C_1) = P(C_2) = P(C_3) = \ldots = P(C_m)$ and in that case, all we need to do is to maximize $\boldsymbol{P(X|C_i)}$. Otherwise, we maximize $\boldsymbol{P(X|C_i)P(C_i)}$.

3. For high dimension data, it would be very expensive computationally to calculate $\boldsymbol{P(X|C_i)}$. For this, the naïve assumption of class-conditional independence is made which assumes that the attribute values are conditionally independent of each other. Thus,

$$\boldsymbol{P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i)}$$

$$\boldsymbol{= P(x_1|C_i) \times P(x_2|C_i) \times P(x_3|C_i) \times \ldots \times P(x_n|C_i)}$$

4. Now it is easy to estimate the probabilities $\boldsymbol{P(x_1|C_i)}$, $\boldsymbol{P(x_2|C_i)}$, … , $\boldsymbol{P(x_n|C_i)}$ from the training tuples. Here, x_k refers to the value of the corresponding attribute A_k of tuple X. x_k is calculated based on the type of attribute i.e. whether the attribute is categorical or continuous valued. For different types x_k is calculated differently as follows:

    a. If A_k is categorical, then $P(x_k|C_i)$ is the number of tuples of class C_i in D having the value x_k for A_k, divided by $|C_i,D|$, the number of tuples of class C_i in D.
    b. A continuous-valued attribute is typically assumed to have a Gaussian distribution with a mean μ and standard deviation σ , defined by

$$\boldsymbol{g(x,\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}}$$

    so that,

$$\boldsymbol{P(x_k|C_i) = g(x_k,\mu_{C_i},\sigma_{C_i})}$$

    Here, $\boldsymbol{\mu_{C_i}}$ and $\boldsymbol{\sigma_{C_i}}$ are mean and standard deviation, respectively of the attribute values of A_k for tuples of class C_i.

5. $\boldsymbol{P(X|C_i)P(C_i)}$ are calculated for each class C_i. The Naïve Bayes classifier predicts that the tuple X belongs to class C_i if and only if

$$P(C_i|X) > P(C_j|X) \quad for\ 1 \le j \le m, j \ne i$$

4.2 Decision Tree Induction

Decision tree induction is the process of creating decision trees for class-labeled training tuples **[38]**. A decision tree is basically a tree structure like a flowchart **[37]**. Each internal node of the tree represents a test on an attribute and each branch is the outcome of the test. Each leaf node is a class label. The first node at the top of the tree is the root node. Figure 3 shows a typical decision tree. It is a sample tree describing prediction rules for predicting runs based on the four derived attributes explained on section 3. Internal nodes of the tree are denoted by rectangles and leaf nodes are represented by ovals.

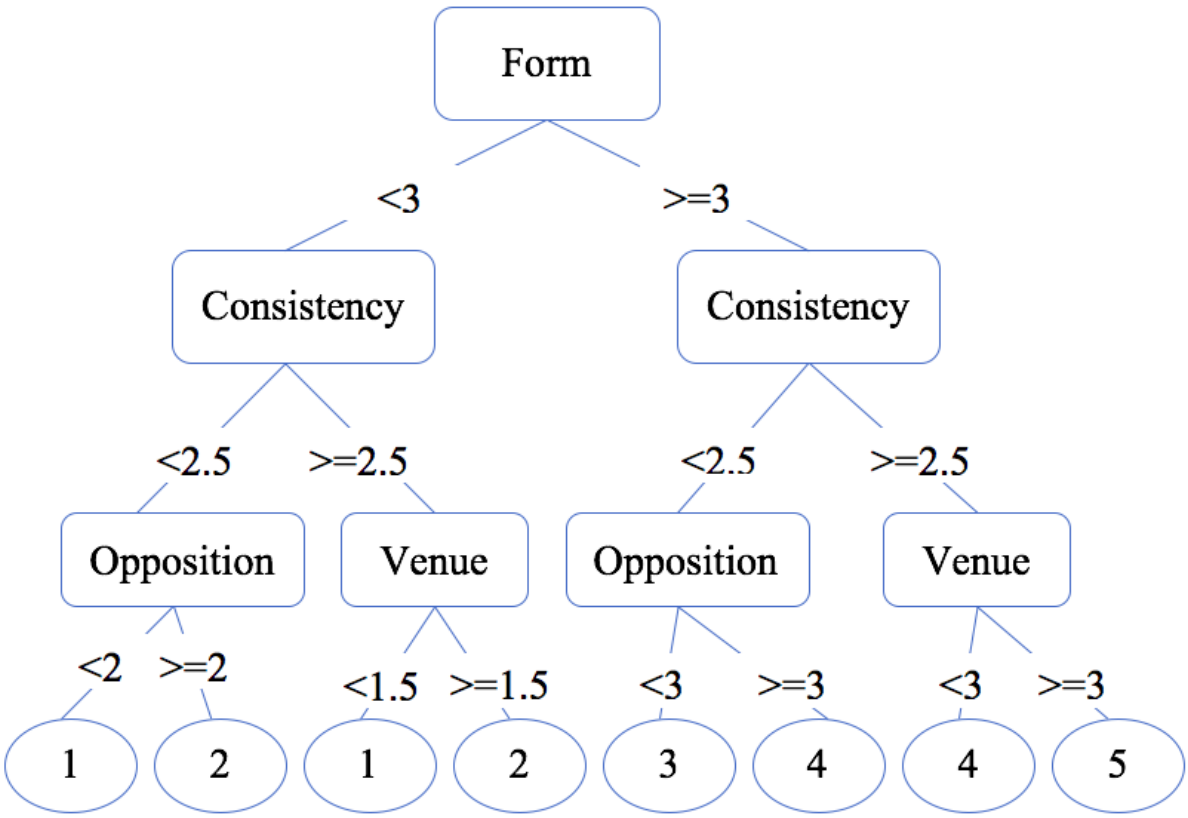

Figure 3 A decision tree describing prediction rules for predicting runs based on the derived attributes

To classify a given tuple X, the attributes of the tuple are tested against the decision tree starting from the root node to the leaf node which holds the class prediction of the tuple. The construction of decision trees is easy as it does not require any domain knowledge or parameter setting. Decision trees can easily handle multidimensional data. The representation of the classification rules in a tree form is intuitive and easy to understand by humans. Decision tree classifiers are fast at learning and classification and have a good accuracy in general.

4.2.1 Decision Tree Classification Algorithm

J. Ross Quinlan introduced a decision tree algorithm called ID3 in his paper **[38]**. Later he introduced a successor of ID3 called C4.5 in **[39]** to overcome some shortcomings such as over-fitting. Later on, L. Breiman, J. Friedman, R. Olshen and C. Stone described the generation of binary decision trees in their book Classification and Regression trees (CART) **[40]**. ID3 and CART follow a similar approach to learn decision trees from training data. ID3, C4.5 and CART are greedy algorithms which construct decision trees from top to down in a recursive divide-and-conquer manner. They start with a training set with tuples and their associated class labels. The training set is then recursively partitioned into smaller subsets as the tree is being built. The general strategy of the decision tree algorithms is described as follows:

- The algorithm starts with a data-partition D, an attribute list and an attribute selection method. The data partition D is the entire training set at the beginning. Attribute list is the list of attributes describing the data tuples. Attribute selection method is a procedure that determines the best attribute that discriminates the data tuples according to their

class. This procedure uses an attribute selection measure such as information gain or Gini index. The attribute selection measure determines if the decision tree is binary or non-binary.

- The tree starts at a single node N which contains all the tuples from D. If all the tuples in D belong to the same class, N becomes a leaf and the algorithm stops. Otherwise, the attribute selection method is called which determines the splitting criterion. The splitting criterion determines the best way to partition the training tuples into individual classes and returns the attribute that should be tested at node N. The splitting criterion also tells us which branches to grow from node N with respect to the outcomes of the chosen test. Ideally, the splitting criterion is determined so that the tuples in the same partition belong to the same class.

- The node N is labeled with the splitting criterion. Each branch from the node N represents the outcome of the splitting criterion. The tuples in D are then partitioned according to the test determined by the splitting criterion. There are three possible scenarios as shown in figure 4. Let A be the attribute determined by the splitting criterion, having v different values $\{a_1, a_2, \ldots, a_v\}$.

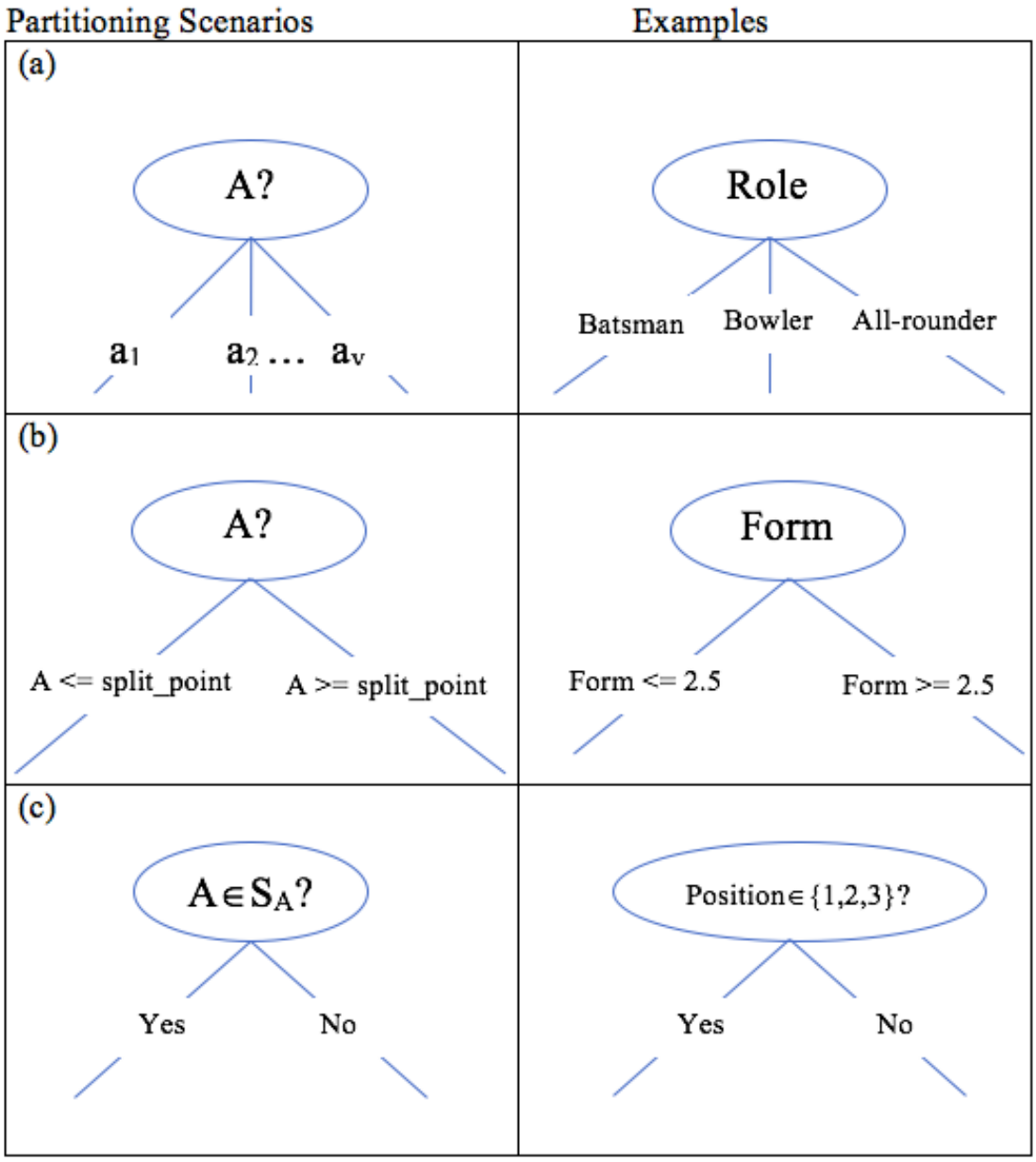

Figure 4 Possibilities of partitioning tuples based on splitting criterions

a. A is discrete valued: In this case, the outcomes of the test at node N are simply the known discrete values of A. A branch is created for each value and is labeled with that value. Partition D_j is the subset of class-labeled tuples in D having value a_j of A. As all the tuples in a given partition have the same value of A, A need not be considered in any future partitioning of the tuples.

b. A is continuous values: In this case, there are two possible outcomes of the test at node N based on the spilt point determined by the splitting criterion. Let a be the split point. The two possible outcomes are $A \leq a \; and \; A \geq a$. Two branches are created from N corresponding to the two outcomes and the tuples are partitioned accordingly.

c. A is discrete valued and a binary tree must be produced: The test at node N is of the form "$A \in S_A$?," where S_A is the splitting subset for A, returned by Attribute selection method as part of the splitting criterion. It is a subset of the known values of A. If a given tuple has value aj of A and if $a_j \in S_A$, then the test at node N is satisfied. Two branches are grown from N. By convention, the left branch out of N is labeled yes so that D_1 corresponds to the subset of class-labeled tuples in D that satisfy the test. The right branch out of N is labeled no so that D_2 corresponds to the subset of class-labeled tuples from D that do not satisfy the test.

- The above procedure is called recursively to form a decision tree. The recursive partitioning stops when one of the following conditions is met:
  a. All the tuples in the partition D belong to the same class.
  b. There are no remaining attributes on which the tuples can be partitioned. In this case, node N is converted to a leaf and labeled with the most common class in D.
  c. There are no more tuples to be partitioned. In this case, a leaf node is created with majority class in D.

4.2.2 Attribute Selection Methods

ID3 uses the attribute selection measure called information gain, which is simply the difference of the information needed to classify a tuple and the information needed after the split. These two can be formularized as follows:

Expected information needed to classify a tuple in the training set D

$$\boldsymbol{Info(D) = -\sum_{i=1}^{m} p_i \, log_2(p_i)}$$

where; pi is the nonzero probability that a tuple in D belongs to class Ci.

Information needed after the splitting (to arrive at the exact classification)

$$\boldsymbol{Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)}$$

where A is the attribute on which the tuples are to be partitioned.

Then, information gain

$$\boldsymbol{Gain(A) = Info(D) - Info_A(D)}$$

The attribute with highest information gain is selected as the splitting attribute.

C4.5 uses gain ratio as the attribute selection measure. Gain ratio is an extension to information gain in a sense because it normalizes information gain by using a split information value;

$$\boldsymbol{SplitInfo_A(D)} = -\sum_{\boldsymbol{j=1}}^{\boldsymbol{v}} \frac{\boldsymbol{|D_j|}}{\boldsymbol{|D|}} \times \boldsymbol{log_2}\left(\frac{\boldsymbol{|D_j|}}{\boldsymbol{|D|}}\right)$$

Then,

$$\boldsymbol{GainRatio(A)} = \frac{\boldsymbol{Gain(A)}}{\boldsymbol{SplitInfo_A(D)}}$$

The attribute with the highest gain ratio is selected as the splitting attribute.

4.3 Random Forest

Random Forest is an ensemble method for classification and regression [37]. Random forests are a set of decision trees where each tree is dependent on a random vector sampled independently and with the same distribution of all the trees in the forest [41]. The algorithm generates a number of decision trees creating a forest. Each decision tree is generated by selecting random attributes at each node to determine the split [41]. Tim Kam Ho introduced the first method for random forests using random subspace method in his paper [42]. Later, Breiman Leo extended the algorithm in his paper [41] and this method was official known as Random Forests. The general procedure to generate decision trees for random forests starts with a dataset D of d tuples. To generate k decision trees from the dataset, for each iteration k, a training set D_i of d tuples is sampled with replacement form the dataset D. To construct a decision tree classifier, at each node, a small number of attributes from the available attributes are selected randomly as candidates for the split at the node. Then Classification And

Regression Trees(CART) [40] method is used to grow the trees. The trees are then grown to maximum size and are not pruned. CART is a non-parametric decision tree induction technique that can generate classification and regression trees. CART recursively selects rules based on variables' values to get the best split. It stops splitting when it detects that no further gain can be made or some pre-determined stopping conditions are met.

4.3.1 Classification and Regression Trees – CART

L. Breiman, J. Friedman, R. Olshen and C. Stone introduced a decision tree algorithm in their book Classification and Regression Trees [40]. The CART algorithm grows trees by choosing a split among all possible splits at each node so that the resulting child nodes are the purest. CART considers only univariate splits i.e. each split depends on the value of only one predictor variable. All possible splits consist of possible splits of each predictor. If X is a nominal categorical variable of n categories, there are $2^n - 1$ possible splits of this predictor. If X is an ordinal categorical or continuous variable with m different values, there are m - 1 different splits on X. A tree is grown starting from the root node by repeatedly using the following steps on each node. Following are the steps of tree growing process of CART algorithm:

1. Find each predictor's best split.
    a. For each continuous and ordinal predictor, sort its values from the smallest to the largest. For the sorted predictor, go through each value from top to examine each candidate split point (call it v, if $x \leq v$, the case goes to the left child node, otherwise, goes to the right) to determine the best. The best split point is the one that maximizes the splitting criterion the most when the node is split according to it.

b. For each nominal predictor, examine each possible subset of categories (call it A, if x ∈ A, the case goes to the left child node, otherwise, goes to the right) to find the best split.

2. Find the node's best split.

   Among the best splits found in step 1, choose the one that maximizes the splitting criterion.

3. Split the node using its best split found in step 2 if the stopping rules are not satisfied.

Stopping Rules:

Stopping rules control if the tree growing process should be stopped or not. The following stopping rules are used:

- If a node becomes pure; that is, all cases in a node have identical values of the dependent variable, the node will not be split.
- If all cases in a node have identical values for each predictor, the node will not be split.
- If the current tree depth reaches the user-specified maximum tree depth limit value, the
- tree growing process will stop.
- If the size of a node is less than the user-specified minimum node size value, the node will not be split.
- If the split of a node results in a child node whose node size is less than the user-specified minimum child node size value, the node will not be split.

4.4 Support Vector Machine

Vladimir Vapnik, Bernhard Boser and Isabell Guyon introduced the concept of support vector machine in their paper [43]. SVMs are highly accurate and less prone to overfitting. SVMs can be used for both numeric prediction and classification. SVM transforms the original data into a higher dimension using a nonlinear mapping. It then searches for a linear optimal hyperplane in this new dimension separating the tuples of one class from another. With an appropriate mapping to a sufficiently high dimension, tuples from two classes can always be separated by a hyperplane. The algorithm finds this hyperplane using support vectors and margins defined by the support vectors. The support vectors found by the algorithm provide a compact description of the learned prediction model. SVM takes different approaches to classify linearly separable and linearly non-separable data.

4.4.1 When Data are Linearly Separable

Let D be a data set given as (X_1, y_1), (X_2, y_2), (X_3, y_3), … , (X_n, y_n); where X_i is the set of training tuples and y_i are their corresponding class labels. Each y_i has two possible values, +1 or -1. Consider two input attributes A_1 and A_2 as shown in figure 5.

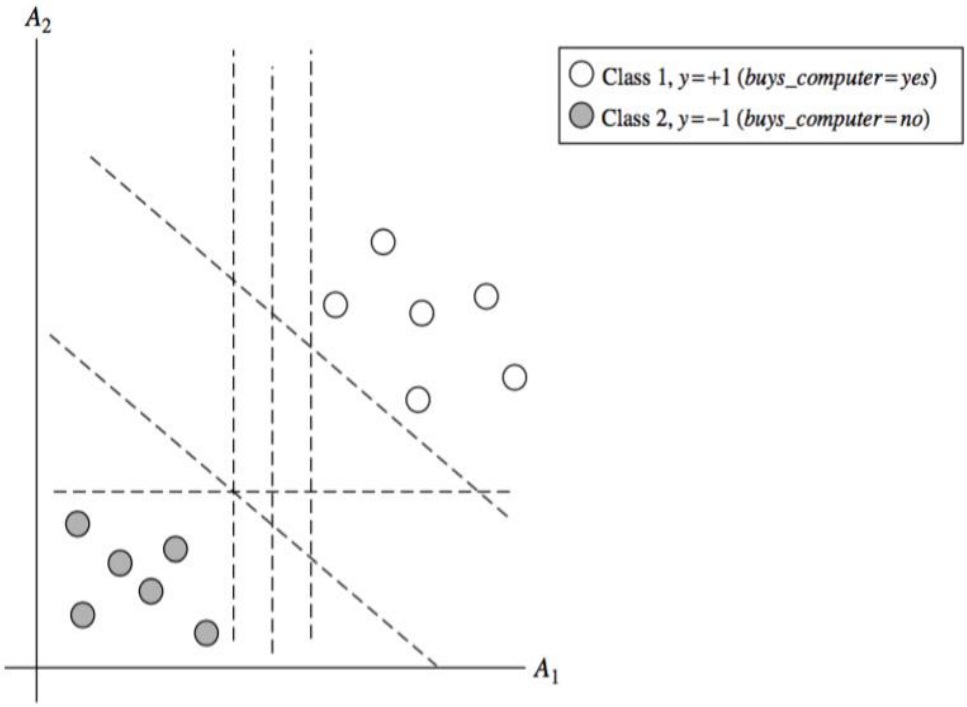

Figure 5 Linearly separable 2-D data [37]

From the figure, we can see that we can draw a straight line to separate the data points from class +1 with the data points form class -1. Thus, this data set is linearly separable. An infinite number of lines can be drawn to separate the class +1 and class -1 data points. The problem is to find the best one i.e. one that will have minimum classification error on new unseen tuples. If our data is 3 dimensional, we have to find a plane separating the data points. In general, for n-dimensional data, we need to find the best separating hyperplane to classify our data.

SVM tries to solve this problem by searching for the maximum marginal hyperplane. Figure 6 shows two possible hyperplanes for separating the data points and their associated margins.

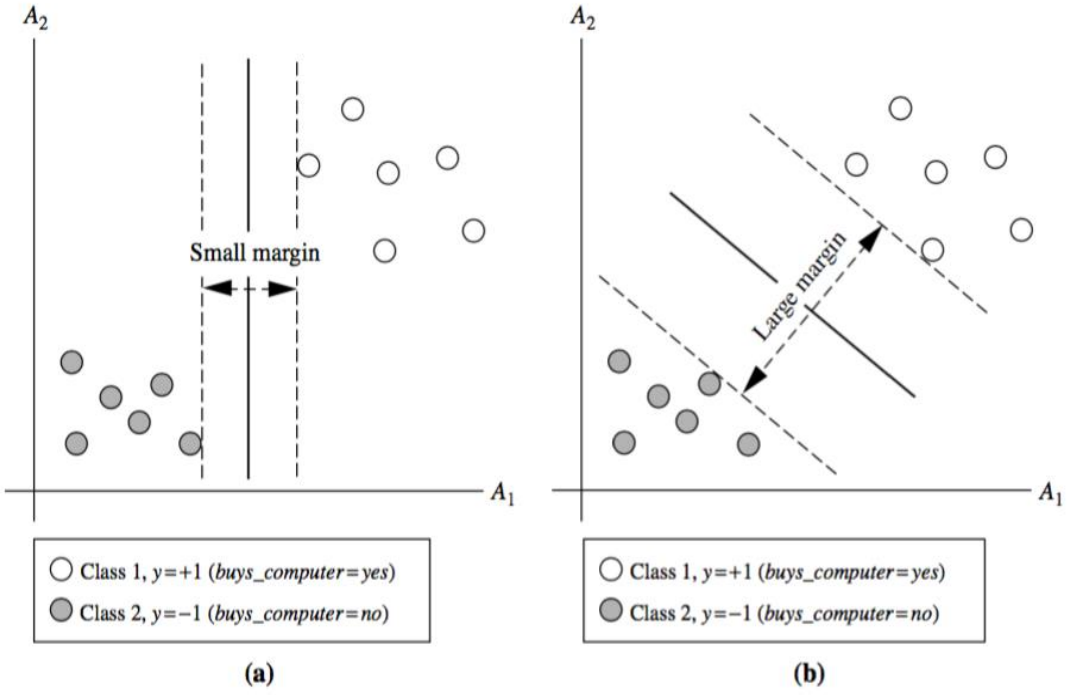

Figure 6 Possible hyperplanes for separating the data points [37]

As we can see, both the hyperplanes correctly separate the data points. But it is obvious that the one with the larger margin is expected to be more accurate for classifying unseen data tuples. SVM tries to find this hyperplane which is called maximum marginal hyperplane.

A separating hyperplane can be written as:

$$\boldsymbol{W} \cdot \boldsymbol{X} + \boldsymbol{b} = \mathbf{0}$$

where W is a weight vector, W = {w_1, w_2, w_3, … , w_n}, n is the number of attributes and b is a scalar often referred to as a bias. If we input two attributes A_1 and A_2, training tuples are 2-D, (e.g., X = (x_1, x_2)), where x_1 and x_2 are the values of attributes A_1 and A_2, respectively. Thus, any points above the separating hyperplane belong to class +1:

$$\boldsymbol{W} \cdot \boldsymbol{X} + \boldsymbol{b} > \mathbf{0}$$

and any points below the separating hyperplane belong to class -1:

$$\boldsymbol{W} \cdot \boldsymbol{X} + \boldsymbol{b} < \mathbf{0}$$

4.4.2 When Data are Linearly Inseparable

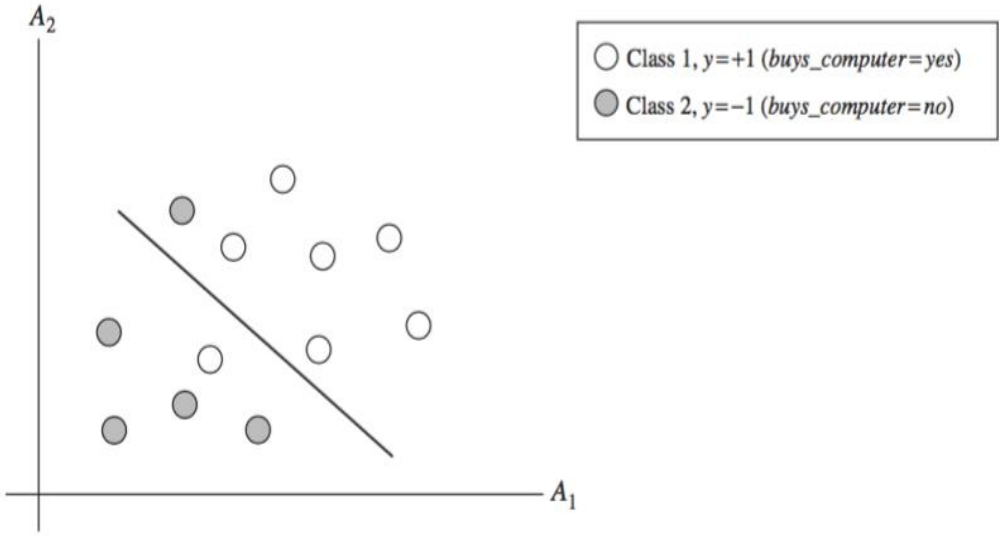

Figure 7 Linearly inseparable 2-D data [37]

Figure 7 shows a sample data where a straight line cannot be drawn to separate the data points. Such data are called linearly inseparable data. In this case, the strategy described above will not be able to classify the data tuples. But that approach can be extended to create nonlinear SVMs to classify such data.

In this approach, the original input data is first transformed into a higher dimensional space using a nonlinear mapping. Once the data is transformed into the new higher dimensional space, the second step is to search for a linear separating hyperplane in the new space. The maximum marginal hyperplane in this space corresponds to a nonlinear separating hypersurface in the original space.

4.4.3 Multiclass SVM

SVM is originally used for binary classification. However, several multiclass SVM algorithms have also been developed. In Weka **[32]**, we used LIBSVM package developed by Chih-Chung Chang and Chih-Jen Lin **[44]**. The package can be downloaded from http://www.csie.ntu.edu.tw/~cjlin/libsvm. LIBSVM is an easy to use package to apply multiclass SVM and has gained a wide popularity in machine learning.

Chapter 5

Results and Discussion

5 Results and Discussion

5.1 Experiment Setup

We used different sizes of training and test sets to find the best combination that gives the most accuracy. We experimented by dividing the data set in four ways:

1. 60% training set – 40% test set
2. 70% training set – 30% test set
3. 80% training set – 20% test set
4. 90% training set – 10% test set

We analyzed and compared the performance of the algorithms in terms of several performance measures, which are described below in short:

Accuracy: The prediction accuracy of an algorithm is the ratio of the number of test instances correctly classified by the algorithm to the total number of test instances. The higher the accuracy, the better the performance.

$$Accuracy = \frac{Number\ of\ correctly\ classified\ instances}{Total\ number\ of\ instances}$$

Precision: Precision of a class is the ratio of the number of instances which were correctly predicted to be in that class(true-positive) to the total number of instances which were predicted to be in that class. Precision indicates how useful the model is, as it shows the how many instances were classified correctly from the ones that were classified. Let x be a class,

$$Precision = \frac{Number\ of\ instances\ of\ class\ x\ predicted\ correctly}{Total\ number\ of\ instances\ which\ were\ predicted\ to\ be\ in\ class\ x}$$

Recall: Recall of a class is the ratio of the number of instances which were correctly predicted to be in that class(true-positive) to the total number of instances of that class. Recall indicates how complete the model is, as it shows how many instances was the model able to find out correctly out of the total number of instances of a class. Let x be a class,

$$Recall = \frac{Number\ of\ instances\ of\ class\ x\ predicted\ correctly}{Total\ number\ of\ instances\ in\ class\ x}$$

F1 Score: F1 score of a class is the harmonic mean of precision and recall of the class. It captures the meaning of both precision and recall.

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Area under the ROC curve (AUROC): A Receiver Operating Characteristic curve is a graphical plot of true positive rate also known as sensitivity against false positive rate also

known as specificity of a binary classifier. True positive rate of a classifier is the ratio of the number of instances that were classified correctly as positives to the total number of positive instances. False positive rate is the ratio of the number of instances that were incorrectly classified as positives to the total number of negative instances. For multiclass problems, ROC curves are generated for each class by using one-vs-all approach. The area under the ROC curves is a measure of accuracy. Its value ranges from 0.5 to 1, with 0.5 meaning least accurate and 1 meaning most accurate.

The values of precision, recall, F1 score and AUROC in the tables in the following sections are weighted averages of the values of these measures for each class.

We used four machine learning algorithms: Naïve Bayes, Decision Trees, Random Forest and Support Vector Machine in our experiments. We simulated these algorithms in Weka **[32]** and Dataiku **[33]**. Following is a brief discussion on the performance of these algorithms and then we compare their performance based on prediction accuracy. All the results in this study have been obtained from Weka **[32]** 3-9-1-oracle-jvm and Dataiku Data Science Studio **[33]** on Mac OS 10.11.6 and Windows 10.

5.2 Naïve Bayes

Table 3 shows the prediction accuracy of Naïve Bayes for predicting runs with different sizes of training and test sets.

Table 3 Prediction accuracy of Naïve Bayes for predicting runs

Dataset Split	60% train – 40% test	70% train – 30% test	80% train – 20% test	90% train – 10% test
Accuracy(%)	43.08	42.95	42.47	42.50

For predicting runs, Naïve Bayes showed highest prediction accuracy of 43.08% with 60% training set and 40% test set. As it can be seen from the table, the prediction accuracy of the classifier decreases as we increase the size of the training set and decrease the test set. It showed an accuracy of 42.95% with 70% training set and 30% test set, 42.47% with 80% training set and 20% test set and the least accuracy of 42.50% with 90% training set and 10% test set. We simulated Naïve Bayes algorithms in Weka [32] as dataiku [33] does not have an implementation of the algorithm. Table 4 shows different performance measures of Naïve Bayes classifier with 60% training data and 40% test data for predicting runs.

Table 4 Performance of Naïve Bayes with 60% training data and 40% test data for predicting runs

Accuracy (%)	**Precision**	**Recall**	**F1 Score**	**AUROC**
43.08	0.424	0.431	0.418	0.740

Table 5 shows the prediction accuracy of Naïve Bayes for predicting wickets with different sizes of training and test sets.

Table 5 Prediction accuracy of Naïve Bayes for predicting wickets

Dataset Split	60% train – 40% test	70% train – 30% test	80% train – 20% test	90% train – 10% test
Accuracy(%)	57.05	57.18	57.48	58.12

For predicting wickets, Naïve Bayes had the highest accuracy of 58.12% with 90% training data and 10% test data and the least accuracy of 57.05% with 60% training data and 40% test data. Here the prediction accuracy increases as we increase the size of the training sample and decrease the test sample. We have an accuracy of 57.18% with 70% training data and 30% test data and 57.48% with 80% training data and 20% test data. Table 6 shows the performance of Naïve Bayes with 90% training data and 10% test data.

Table 6 Performance of Naïve Bayes with 90% training data and 10% test data for predicting wickets

Accuracy (%)	**Precision**	**Recall**	**F1 Score**	**ROC AUC**
58.12	0.577	0.581	0.575	0.765

5.3 Decision Trees

Table 7 shows the prediction accuracy of Decision Trees for predicting runs with different sizes of training and test sets.

Table 7 Prediction accuracy of Decision Trees for predicting runs

Dataset Split	60% train – 40% test	70% train – 30% test	80% train – 20% test	90% train – 10% test
Accuracy(%)	77.93	79.02	79.38	82.52

Decision Trees showed an accuracy of 80.46% with 90% training data and 10% test data for predicting runs. The accuracy decreased with decrease in training size and increase in test size. We had the least accuracy of 77.93% with 605 training data and 40% test data, 79.02% with 70% training data and 30% test data and 79.38% with 80% training data and 20% test data. We simulated Decision Trees in both Weka **[32]** and dataiku **[33]**. Table 8 shows detailed metrics of performance of Decision Trees in Weka **[32]** as we got the highest accuracy in Weka **[32]**.

Table 8 Performance of Decision Trees with 90% training data and 10% test data for predicting runs

Accuracy (%)	**Precision**	**Recall**	**F1 Score**	**ROC AUC**
82.52	0.824	0.825	0.824	0.923

Table 9 shows the prediction accuracy of Decision Trees for predicting wickets with different sizes of training and test sets.

Table 9 Prediction accuracy of Decision Trees for predicting wickets

Dataset Split	60% train – 40% test	70% train – 30% test	80% train – 20% test	90% train – 10% test
Accuracy(%)	84.40	85.12	85.99	86.50

For predicting wickets, the highest accuracy that decision trees could achieve was 86.50% with 90% training data and 10% test data. We had the least accuracy of 84.40% with 60% training data and 40% test data. As can be seen from the table, the prediction accuracy increases with increase in training data size and decrease in test data as we have an accuracy of 85.12% with 70% training data and 30% test data and 85.99% with 80% training data and 20% test data. Table 10 shows detailed metrics of performance of Decision Trees for predicting wickets. We got the highest accuracy with Decision Trees in Weka [32].

Table 10 Performance of Decision Trees with 90% training data and 10% test data for predicting wickets

Accuracy (%)	Precision	Recall	F1 Score	ROC AUC
86.50	0.865	0.865	0.865	0.921

5.4 Random Forest

Table 11 shows the prediction accuracy of Random Forest for predicting runs with different sizes of training and test sets.

Table 11 Prediction accuracy of Random Forest for predicting runs

Dataset Split	60% train – 40% test	70% train – 30% test	80% train – 20% test	90% train – 10% test
Accuracy(%)	89.92	90.27	90.67	90.88

Random Forest had the accuracy of 90.88% with 90% training set and 10% test set for predicting runs. As we decrease the size of the training set and increase the size of the test set,

the accuracy of the classifier decreases as shown in the table, where we have 90.67% accuracy with 80% training set and 20% test set, 90.27% with 70% training set and 30% test set and the least accuracy of 89.92% with 60% training set and 40% test set. Table 12 shows the detailed metrics of performance of Random Forest in dataiku **[33]** as we got the highest accuracy in dataiku **[33]**.

Table 12 Performance of Random Forest with 90% training data and 10% test data for predicting runs

Accuracy	Precision	Recall	F1 Score	ROC AUC
90.88	0.908	0.908	0.908	0.987

Table 13 shows the prediction accuracy of Random Forest for predicting wickets with different sizes of training and test sets.

Table 13 Prediction accuracy of Random Forest for predicting wickets

Dataset Split	60% train – 40% test	70% train – 30% test	80% train – 20% test	90% train – 10% test
Accuracy(%)	90.68	91.26	91.80	92.30

For predicting wickets, Random Forest achieved the highest accuracy of 92.30% with 90% training data and 10% test data. The accuracy of the classifier decreases as we decrease the size of the training set and increase the size of the test set. We have an accuracy of 91.80% with 80% training set and 20% test set, 91.26% with 70% training set and 30% test set and the least accuracy of 90.68% with 60% training set and 40% test set. Table 14 shows the performance metrics of Random Forest for predicting wickets in Weka **[32]**.

Table 14 Performance of Random Forest with 90% training data and 10% test data for predicting wickets

Accuracy (%)	Precision	Recall	F1 Score	ROC AUC
92.30	0.923	0.923	0.923	0.975

5.5 Support Vector Machine

Table 15 shows the prediction accuracy of Support Vector Machine for predicting runs with different sizes of training and test sets.

Table 15 Prediction accuracy of Support Vector machine for predicting runs

Dataset Split	60% train – 40% test	70% train – 30% test	80% train – 20% test	90% train – 10% test
Accuracy(%)	60.58	60.89	60.92	61.77

For predicting runs, support vector machine had the highest accuracy of 61.77% with 90% training data and 10% test data. The accuracy decreases with decrease in the size of training data and increase in the size of test data. With 80% training data and 20% test data, we have an accuracy of 60.92%, with 70% training data and 30% test data, we have an accuracy of 60.89% and with 60% training data and 40% test data, we see the least accuracy of 60.58%. Table 16 shows detailed metrics of performance of support vector machine for predicting runs in Dataiku **[33]**.

Table 16 Performance of Support Vector Machine with 90% training data and 10% test data for predicting runs

Accuracy (%)	Precision	Recall	F1 Score	ROC AUC
61.77	0.609	0.616	0.609	0.870

Table 17 shows the prediction accuracy of Support Vector Machine for predicting wickets with different sizes of training and test sets.

Table 17 Prediction accuracy of Support Vector machine for predicting wickets

Dataset Split	60% train – 40% test	70% train – 30% test	80% train – 20% test	90% train – 10% test
Accuracy(%)	69.45	69.53	70.43	70.95

Support vector machine has the highest accuracy of 70.95% with 90% training data and 10% test data. The accuracy of the classifier decreases with decrease in the size of training set and increase in the size of the test set. As can be seen from the table, we have an accuracy of 70.43% with 80% training set and 20% test set, 69.53% with 70% training set and 30% test set and the least accuracy of 69.45% with 60% training set and 40% test set. Table 18 shows the detailed performance of support vector machine for predicting wickets in Dataiku **[33]**.

Table 18 Performance of Support Vector Machine with 90% training data and 10% test data for predicting wickets

Accuracy (%)	Precision	Recall	F1 Score	ROC AUC
70.95	0.720	0.707	0.708	0.867

5.6 Summary

In this section, we give a summary and a comparison of the performance of the algorithms. Table 19 summarizes the accuracies of the algorithms for predicting runs and table 20 summarizes the accuracies of the algorithms for predicting wickets.

Table 19 Accuracies of the algorithms for predicting runs

Classifier	**Accuracy (%)**			
	60% train 40% test	70% train 30% test	80% train 20% test	90% train 10% test
Naïve Bayes	43.08	42.95	42.47	42.50
Decision Trees	77.93	79.02	79.38	80.46
Random Forest	89.92	90.27	90.67	90.88
SVM	60.58	60.89	60.92	61.77

Table 20 Accuracies of the algorithms for predicting wickets

Classifier	Accuracy (%)			
	60% train 40% test	70% train 30% test	80% train 20% test	90% train 10% test
Naïve Bayes	57.05	57.18	57.48	58.12
Decision Trees	84.40	85.12	85.99	86.50
Random Forest	90.68	91.26	91.80	92.30
SVM	69.45	69.53	70.43	70.95

As we can see, Random Forest builds the most accurate prediction models for predicting both runs and wickets in all the cases. Also, the accuracy of the models increases as we increase the size of the training dataset for all algorithms except in case of Naïve Bayes for predicting runs where the accuracy decreases as we increase the size of the training set. Random Forest predicts runs with the highest accuracy of 90.88% when we use 90% of the dataset for training. Similarly, Random Forest predicts wickets with highest accuracy of 92.30% when we use 90% of the dataset for training. On the other hand, Naïve Bayes predicts runs with the least accuracy of 42.5% when we use 90% of the dataset for training. Naïve Bayes predicts wickets too with the least accuracy of 57.05% when we use 60% of the dataset for training. Decision Trees performs reasonably well with the maximum accuracy of 80.46% and the minimum accuracy of 77.93% for predicting runs. It predicts wickets with the maximum accuracy of 86.5% and the minimum accuracy of 84.40%, which is again reasonably well against the performance of

Random Forest. The prediction models of SVM for predicting runs showed the maximum accuracy of 61.77% with 90% training data and the minimum accuracy of 60.58% with 60% training data. Also for wickets, SVM had the maximum accuracy of 70.95% with 90% training data and the minimum accuracy of 69.45% with 60% training data.

Table 21 summarizes the other performance measures of the algorithms with their best values for predicting runs and table 22 summarizes the other performance measures of the algorithms with their best values for predicting wickets.

Table 21 Performance measure of the algorithms for predicting runs

Classifier	Precision	Recall	F1 Score	AUROC
Naïve Bayes	0.424	0.431	0.418	0.740
Decision Trees	0.824	0.825	0.824	0.923
Random Forest	0.908	0.908	0.908	0.987
SVM	0.609	0.616	0.609	0.870

As can be seen from the table, Random Forest performs the best in terms of all the measures woth precision, recall and F1 Score of 0.908 and AUROC of 0.987 which are excellent values for a classifer. On the other hand, Naïve Bayes performs the worst with 0.424 precision, 0.431 recall, 0.418 F1 score and AUROC of 0.740. SVM also showed a poor performance with

precsion of 0.609, recall of 0.616 and F1 score of 0.609. However, it AUROC value is good, which is 0.870. Decision Trees has performed well with precision and F1 score of 0.824, recall of 0.825 and an excellent ROC value of 0.923.

Table 22 Performance measure of the algorithms for predicting wickets

Classifier	Precision	Recall	F1 Score	AUROC
Naïve Bayes	0.577	0.581	0.575	0.765
Decision Trees	0.865	0.865	0.865	0.921
Random Forest	0.923	0.923	0.923	0.975
SVM	0.720	0.707	0.708	0.867

Random Forest again performed the best for predicting wickets in terms of all the measures with precision, recall and F1 score of 0.923 and AUROC value of 0.975. Again, Naïve Bayes shows the worst performance with 0.577 precision, 0.581 recall, 0.575 F1 score and 0.765 AUROC. Decision Trees shows a good performance with precision, recall and F1 score of 0.865 and AUROC of 0.921. SVM performed reasonably well with precision of 0.720, recall of 0.707, F1 score of 0.708 and a good AUROC of 0.867.

Chapter 6

Conclusion and Future Work

6 Conclusion and Future Work

Selection of the right players for each match plays a significant role in a team's victory. An accurate prediction of how many runs a batsman is likely to score and how many wickets a bowler is likely to take in a match will help the team management select best players for each match. In this paper, we modeled batting and bowling datasets based on players' stats and characteristics. Some other features that affect players' performance such as weather or the nature of the wicket could not be included in this study due to unavailability of data. Four multiclass classification algorithms were used and compared. Random Forest turned out to be the most accurate classifier for both the datasets with an accuracy of 90.74% for predicting runs scored by a batsman and 92.25% for predicting wickets taken by a bowler. Results of SVM were surprising as it achieved an accuracy of just 51.45% for predicting runs and 70.95% for predicting wickets.

Similar studies can be carried out for other formats of the game i.e. test cricket and T20 matches. The models for these formats can be shaped to reflect required characteristics of the players; e.g. batsmen need to have patience and ability to play longer innings in test matches whereas score more runs in less overs in T20 matches. Similarly, bowlers need to have stronger wicket taking abilities in test matches and better economy rate i.e. conceding less runs in T20

matches. Moreover, attempts can be made to improve accuracies of the classifiers for ODI matches.

References

[1] [Online]. Available: www.visualdictionaryonline.com.

[2] "How to Play Cricket: 14 Steps (with Pictures) - wikiHow," [Online]. Available: https://www.wikihow.com/Play-Cricket.

[3] S. Muthuswamy and S. S. Lam, "Bowler Performance Prediction for One-day International Cricket Using Neural Networks," in *Industrial Engineering Research Conference*, 2008.

[4] I. P. Wickramasinghe, "Predicting the performance of batsmen in test cricket," *Journal of Human Sport & Excercise,* vol. 9, no. 4, pp. 744-751, May 2014.

[5] G. D. I. Barr and B. S. Kantor, "A Criterion for Comparing and Selecting Batsmen in Limited Overs Cricket," *Operational Research Society,* vol. 55, no. 12, pp. 1266-1274, December 2004.

[6] S. R. Iyer and R. Sharda, "Prediction of athletes performance using neural networks: An application in cricket team selection," *Expert Systems with Applications,* vol. 36, pp. 5510-5522, April 2009.

[7] M. G. Jhanwar and V. Pudi, "Predicting the Outcome of ODI Cricket Matches: A Team Composition Based Approach," in *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD 2016 2016)*, 2016.

[8] H. H. Lemmer, "The combined bowling rate as a measure of bowling performance in cricket," *South African Journal for Research in Sport, Physical Education and Recreation,* vol. 24, no. 2, pp. 37-44, January 2002.

[9] D. Bhattacharjee and D. G. Pahinkar, "Analysis of Performance of Bowlers using Combined Bowling Rate," *International Journal of Sports Science and Engineering,* vol. 6, no. 3, pp. 1750-9823, 2012.

[10] S. Mukherjee, "Quantifying individual performance in Cricket - A network analysis of batsmen and bowlers," *Physica A: Statistical Mechanics and its Applications*, vol. 393, pp. 624-637, 2014.

[11] P. Shah, "New performance measure in Cricket," *ISOR Journal of Sports and Physical Education,* vol. 4, no. 3, pp. 28-30, 2017.

[12] D. Parker, P. Burns and H. Natarajan, "Player valuations in the Indian Premier League," *Frontier Economics,* vol. 116, October 2008.

[13] C. D. Prakash, C. Patvardhan and C. V. Lakshmi, "Data Analytics based Deep Mayo Predictor for IPL-9," *International Journal of Computer Applications,* vol. 152, no. 6, pp. 6-10, October 2016.

[14] M. Ovens and B. Bukiet, "A Mathematical Modelling Approach to One-Day Cricket Batting Orders," *Journal of Sports Science and Medicine,* vol. 5, pp. 495-502, 15 December 2006.

[15] F. C. Duckworth and A. J. Lewis, "A Fair Method for Resetting the Target in Interrupted One-Day Cricket Matches," *The Journal of the Operational Research Society*, vol. 49, no. 3, pp. 220-227, March 1998.

[16] G. Sharp, W. J. Bretteny, J. Gonsalves and M. E. Lourens, "Integer optimisation for the selection of a Twenty20 cricket team," *Journal of the Operational Research Society,* pp. 1688-1694, September 2011.

[17] A. Faez, A. Jindal and K. Deb, "Cricket team selection using evolutionary multi-objective optimization," in *International Conference on Swarm, Evolutionary, and Memetic Computing*, Berlin, 2011.

[18] S. N. Omkar and R. Verma, " Cricket team selection using genetic algorithm," in *International congress on sports dynamics (ICSD2003)*, 2003.

[19] V. V. Sankaranarayanan, J. Sattar and L. V. Lakshmanan, "Auto-play: A Data Mining Approach to ODI Cricket Simulation and Prediction," in *2014 SIAM International Conference on Data Mining*, 2014.

[20] A. J. Lewis, "Towards Fairer Measures of Player Performance in One-Day Cricket," *The Journal of the Operational Research Society,* vol. 56, no. 7, pp. 804-815, July 2005.

[21] A. C. Kimber and A. R. Hansford, "A Statistical Analysis of Batting in Cricket," *Journal of the Royal Statistical Society,* vol. 156, no. 3, pp. 443-455, 1993.

[22] T. B. Swartz, P. Gill and S. Muthukumarana, "Modelling and simulation for one-day cricket," *The Canadian Journal of Statistics,* vol. 37, no. 2, pp. 143-160, June 2009.

[23] J. M. Norman and S. R. Clarke, "Optimal batting orders in cricket," *The Journal of the Operational Research Society,* vol. 61, no. 6, pp. 980-986, June 2010.

[24] P. Kalgotra, R. Sharda and G. Chakraborty, "Predictive modeling in sports leagues: an application in Indian Premier League," 2013.

[25] H. H. Lemmer, "An Analysis Of Players' Performances In The First Cricket Twenty20 World Cup Series," *South African Journal for Research in Sport, Physical Education and Recreation,* vol. 30, no. 2, pp. 71-77, 2008.

[26] H. Saikia, D. Bhattacharjee and H. Lemmer, "Predicting the Performance of Bowlers in IPL: An Application of Artificial Neural Network," *nternational Journal of Performance Analysis in Sport ,* vol. 12, pp. 75-89, April 2012.

[27] H. Saikia and D. Bhattacharjee, "On Classification of All-rounders of the Indian Premier League (IPL): A Bayesian Approach," *Vikalpa,* vol. 36, no. 4, pp. 51-66, 2011.

[28] T. Gweshe and I. Durbach, "An analysis of the efficiency of player performance at the 2011 Cricket World Cup," *Orion,* vol. 29, no. 2, pp. 137-153, 27 January 2013.

[29] D. Prakash, C. Patwardhan and S. Singh, "A new Category based Deep Performance Index using Machine Learning for ranking IPL Cricketers," *International Journal of Electronics, Electrical and Computational System,* vol. 5, no. 2, pp. 37-47, February 2016.

[30] " Free web scraping - Download the most powerful web scraper | ParseHub," [Online]. Available: https://www.parsehub.com.

[31] "Import.IO | Extract Data From The Web," [Online]. Available: https://www.import.io.

[32] "Weka 3 - Data Mining with Open Source Machine Learning Software in Java," University of Waikato, [Online]. Available: https://www.cs.waikato.ac.nz/ml/weka/.

[33] "Dataiku | Collaborative Data Science Platform," Dataiku, [Online]. Available: https://www.dataiku.com.

[34] T. L. Saaty, The Analytic Hierarchy Process, New York: McGrow Hill, 1980.

[35] T. L. Saaty, "A scaling method for priorities in a hierarchichal structure," *Mathematical Psychology,* vol. 15, 1977.

[36] N. V. Chavla, K. W. Bowyer, L. O. Hall and P. W. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research,* vol. 16, pp. 321-357, June 2002.

[37] J. Han, M. Kamber and J. Pei, Data Mining Concepts and Techniques, 3rd Edition ed., Waltham: Elsevier, 2012.

[38] J. R. Quinlan, "Induction of Decision Trees," *Machine learning,* vol. 1, no. 1, pp. 81-106, 1986.

[39] J. R. Quinlan, C4.5: Programs for Machine Learning, Elsevier, 2015.

[40] L. Breiman, J. Friedman, C. J. Stone and R. A. Olshen, Classification and regression trees, CRC Press, 1984.

[41] L. Breiman, "Random Forests," *Machine Learning,* vol. 45, no. 1, pp. 5-32, 2001.

[42] T. K. Ho, "The Random Subspace Method for Constructing Decision Forests," *IEEE transactions on pattern analysis and machine intelligence,* vol. 20, no. 8, pp. 832-844, August 1998.

[43] B. E. Boser, I. M. Guyon and V. N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers," in *Fifth Annual Workshop on Computational Learning Theory*, Pittsburgh, 1992.

[44] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology,* vol. 2, no. 3, April 2011.