DATA MINING ON CRICKET DATA SET FOR PREDICTING THE RESULTS

by

Sushant Murdeshwar

A Project Report Submitted
in
Partial Fulfillment of the
Requirements for the Degree of
Master of Science
in
Computer Science

Supervised by
Dr. Carol Romanowski

Department of Computer Science

B. Thomas Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, New York

December, 2016

Acknowledgements

I would like to thank my advisor, Dr. Carol Romanowski for providing me with her valuable advice in this entire semester for implementing my project. I am very much thankful to her for all the guidance that she gave me whenever I had any doubts while working on this project. This project would not have been successful without the weekly feedback that she provided me on my progress. I would also like to take the opportunity to thank my colloquium advisor Dr. Joe Geigel for all the feedback that he provided me during the milestone presentations and the poster rehearsal. The feedback that he gave during the poster rehearsal definitely helped me in improving some of the aspects of my poster.

Abstract

Cricket is a sport that contains a lot of statistical data. There is data about batting records, bowling records, individual player records, scorecard of different matches played, etc. This data can be put to proper use to predict the results of games and so this problem has become an interesting problem in today's world. Most of viewers nowadays try to do some sort of prediction at some stage of the tournaments to see which team will eventually win the upcoming matches and thereby the tournament. This report aims at solving the problem of predicting the results of games by identifying the important attributes from the data set and using the data mining algorithms. I have limited my area of study to the domestic Twenty 20 tournament which is held in India every year during the summer i.e. the Indian Premier League. The previous work that I read and used as a reference were either predicting game results for all sports in general or sports like Basketball, Soccer, etc. My report describes in detail the different attribute selection techniques as well as the data mining algorithms used to solve this problem of result prediction in cricket. I have also used accuracy as the evaluation criteria to evaluate how well the prediction performs. Some future work is also suggested in this report in case some other student in the future is interested in continuing the study of this problem and improving upon my results.

Table of Contents

1. Introduction

Cricket is being played in many countries all around the world. There are a lot of domestic and international tournaments being held in many countries which play cricket. Cricket is a game played between two teams comprising of 11 players in each team. The result is either a win, loss or a tie. However, sometimes due to bad weather conditions the game is also washed out as Cricket is a game which cannot be played in rain. Moreover, this game is also extremely unpredictable because at every stage of the game the momentum shifts to one of the teams between the two. A lot of times the result gets decided on the last ball of the match where the game gets really close. Considering all these unpredictable scenarios of this unpredictable game, there is a huge interest among the spectators to do some prediction either at the start of the game or during the game. Many spectators also play betting games to win money. So, keeping in mind all these possibilities, this report aims at studying the problem of predicting the game results before the game has started based on the statistics and data available from the data set. The study uses the Indian Premier League data set of all 8 seasons played till now i.e. from 2008 to 2016.

There are different ways to do the prediction. The prediction can be done taking into consideration the player's performance as well as the team performance. There are many unpredictable things that happen in a cricket game like matches being washed out due to rain, a key player getting injured before the game, players changing their teams, etc. Sometimes a key player also gets inured during the game and hence is not able to take further part in the game. All these factors do affect the prediction to some extent. The report discusses a methodology that I followed for the game result prediction. The methodology consists of first the attribute selection algorithms which trim down the list of attributes to only important ones and then the data mining algorithms which can be applied on those attributes. The game prediction problem that I am studying does not take into consideration the player's performance but it does take into consideration the team's past performance at a high level extent along with the other factors like toss winner, toss decision, home support, etc. The attribute selection techniques consist of the wrapper method and the ranker method. The data mining algorithms that are used are Decision Tree (J48), Random Forest, Naïve Bayes, K-Nearest Neighbor. The data mining tool used in the project is WEKA and it is a freely available data mining tool which has good support for a number of different data mining algorithms.

2. Related Work

There has been a lot of related study to this problem in various different sports. The papers I have used as references are all related work that had been done on this problem. The paper by Trawinski [2] described the prediction of results using a fuzzy classification system. This paper was predicting the results for basketball games. I had used the attribute selection techniques mentioned in this paper for my project. The attribute selection technique proposed in this paper was done using WEKA so it was a good reference point for me too. The wrapper method algorithms and the ranker method algorithms implemented in this paper were also used in my project. But the prediction part was done using the fuzzy classification system and I did not use that system for my prediction part.

The paper by Haghighat [3] also described the prediction of results in sports using the data mining techniques. But this paper was not specific to any particular sport, rather it was for in general all sports. The attribute selection algorithm that it used was more of an elimination approach where the attributes were eliminated one by one and the classification accuracy is computed. Once a good subset of attributes is achieved, then the eliminated attributes are again added one by one to see if the accuracy improves. But, in my approach I did not use this elimination approach for the attribute selection part. Then the paper used various data mining algorithms to perform the classification. I used the Naïve Bayes and the decision tree algorithms from the paper in my project and compared my accuracy with that of the paper.

The paper by Zdravevski [4] takes into consideration some of the external factors like the number of players injured before a particular game, the winning streak of a team before a particular game, the fatigue factor of the players, etc. The approach that I used in the project does not take into account the external features like player injury, fatigue or the winning streak. My data set contains more data about the matches and the events happening in the match like toss and player of match rather than the data about external factors.

3. Methodology

I have followed the following methodology in the course of my project. The methodology consists of 5 different phases as shown in Figure 1 i.e. Data Set Generation, Data Cleaning, Attribute Selection, Data Mining and Analysis of Results. Each of these phases was part of my project milestones submitted and I had created a Gantt chart to keep track of the timeline. I will be discussing each phase in the following sections in detail.

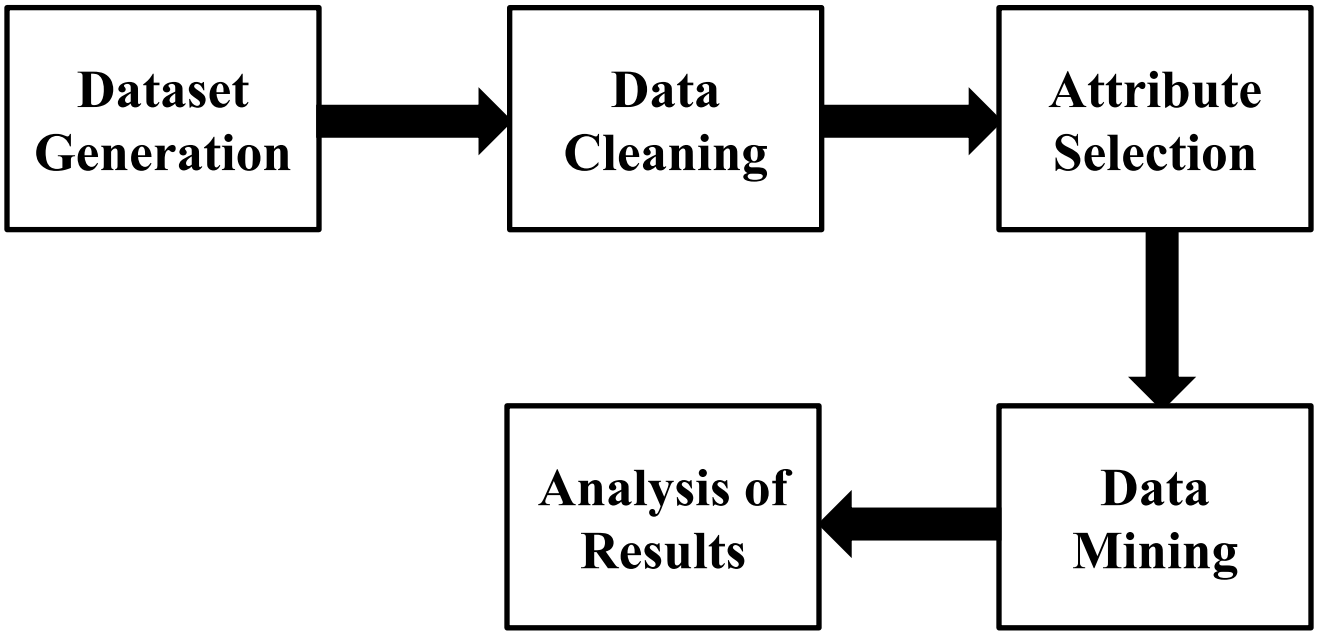


Figure 1: Methodology Diagram

3.1 Data Set Generation

The data was collected from the http://cricsheet.org [1] website. The website has data about all 8 seasons (from 2008 to 2016) of domestic tournaments held in India i.e. the Indian Premiere League. It is a 20-20 format of tournament. It means that each team bats or bowls for maximum 20 overs each and the result of the game is decided at the end of the total 40 overs played. The data set downloaded from this website was of two formats; one was the ball by ball detail of each match in the .csv format and the other data was the high level data about the match in the .json format. I used the Java classes File and the FileWriter to read the .csv files and write the contents to a new file. The .csv files had data about 577 different matches and the Java code basically copied the important data from all the 577 files and combined them into a single file. The data from the .csv file consisted of many irrelevant information like gender, date, umpire information, etc. which were all discarded and the java code copied only data about the team, season, venue, toss, toss_winner, player_of_match and the ball by ball data. The .csv file had data as shown in following figure.

```
version,1
info,team,Royal Challengers Bangalore
info,team,Kolkata Knight Riders
info,gender,male
info,season,2007/08
info,date,2008/04/18
info,competition,Indian Premier League
info,match_number,1
info,venue,M Chinnaswamy Stadium
info,city,Bangalore
info,toss_winner,Royal Challengers Bangalore
info,toss_decision,field
info,player_of_match,BB McCullum
info,umpire,Asad Rauf
info,umpire,RE Koertzen
info,reserve_umpire,VN Kulkarni
info,tv_umpire,AM Saheba
info,match_referee,J Srinath
info,winner,Kolkata Knight Riders
info,winner_runs,140
ball,1,0.1,Kolkata Knight Riders,SC Ganguly,BB McCullum,P Kumar,0,1,"",""
ball,1,0.2,Kolkata Knight Riders,BB McCullum,SC Ganguly,P Kumar,0,0,"",""
```

Figure 2: The .csv file

The .json data contained high level information about the 584 different matches in the Indian Premier League. The .json files contained data about 7 more matches which were missing in the first .csv file. This data helped me in figuring out the missing matches. Here I made use of the JSONParser and JSONArray classes of the org.json.simple java package. This package contains various classes that can be used to parse and read the .json files. I copied the data from the .json file to a new .csv file which had some of the additional information that were not present in the first .csv file. The .json files had data as shown in the following image. So, in the end I combined the two .csv files to generate a new data set with 21 attributes and 584 different instances. This combined data set is the one I have used in my project for predicting the results.


```
[
  {
    "date": "2008-04-18",
    "firstInnings": {
      "batting": {
        "extras": {
          "details": {
            "b": "4",
            "lb": "4",
            "w": "9"
          },
          "total": "17"
        },
        "overs": "20",
        "players": [
          {
            "balls": "12",
            "fours": "2",
            "playerDismissal": "c Jacques Kallis b Zaheer Khan",
            "playerName": "Sourav Ganguly",
            "runs": "10",
            "sixes": "0",
            "strikeRate": "83.33"
          },
```


Figure 3: The .json file

Data Set Description

The new combined data set that I generated had the data from both the .csv files and the .json files. The data set consists of 21 different attributes and 584 different instances. The data set spans across all the seasons of the Indian Premier League from 2008 to 2016 season. The Winning Team is the classifier in the data set and this project is to predict the winning team in the match. The attributes in the data set are as follows:

1) **Season:** The season in which the match was played
2) **Match_Number:** The match number in the current season
3) **Team1:** The Playing team 1
4) **Team2:** The Playing team 2
5) **Venue:** Playing venue
6) **Home_Team:** The team playing at the home venue
7) **Toss_Winner:** The team winning the toss
8) **Toss_Decision:** The team winning the toss will decide to bat or field first
9) **Player_of_Match:** The best player in the match in terms of performance
10) **Team_Batting_First:** The team which bats first
11) **Team_Batting_Second:** The team which bats second
12) **First_Innings_Score:** The total score of the team batting in 1^{st} innings
13) **Overs_Played_In_First_Innings:** The total number of overs played in 1^{st} innings
14) **Wickets_Lost_In_First_Innings:** The total number of wickets lost in 1^{st} innings
15) **First_Innings_Run_Rate:** The rate at which the batting team scores in 1^{st} innings
16) **Second_Innings_Score:** The total score of the team batting in 2^{nd} innings
17) **Overs_Played_In_Second_Innings:** The total number of overs played in 2^{nd} innings
18) **Wickets_Lost_In_Second_Innings:** The total number of wickets lost in 2^{nd} innings
19) **Second_Innings_Run_Rate:** The rate at which the batting team scores in 2^{nd} innings
20) **Winning_Margin:** The margin of runs by which the winning team won the match
21) **Winning_Team:** This is the class attribute i.e. the winning team

3.2 Data Cleaning

The data obtained from the http://cricsheet.org [1] website was already cleaned. So, I did not have to do any sort of cleaning on the data. However, I had to tackle the missing values data and the data for the matches which were washed out due to rain. There were 7 matches whose data were missing in the .csv files. Those matches' data were present in the .json file. So, after combining the two data into a single data set, I had to manually fill in the data about the 7 missing teams from the http://espncricinfo.com/ [5] website. Moreover there were 10 matches which were washed out so those instances were filled with Null values and they were discarded from the data set. So, the final data set had 574 instances with 21 attributes.

3.3 Attribute Selection

The data set obtained after handling the missing values had to be filtered with the help of the attribute selection algorithm. Since, there were 21 attributes it was necessary to identify all the important attributes which would be useful for the data mining tasks. The paper by Haghighat [3] explained the attribute elimination process. Here, first a data set with some number of attributes is selected and then each attribute is eliminated one by one from the set of the attributes. This elimination is based upon the result of running the classification algorithms on the set of attributes. An attribute is completely eliminated if the accuracy improves after its removal or else the attribute is kept in the data set. So, at the end of this elimination process we get a set of attributes using which we get the highest accuracy of prediction for the classification algorithms. The paper by Trawinski [2] described the two types of attribute selection algorithms i.e. the wrapper method and the ranker method that were used by the authors in their study. I used this paper as my reference for the attribute selection phase. I wanted to evaluate the worth of each attribute and rank each attribute to identify their importance before using them for making the predictions during the data mining phase. So I decided to choose these two types of attribute selection algorithms from the paper by Trawinski [2]. These two types of attribute selection methods are available to use in WEKA. After the attribute selection phase, the list of attributes gets minimized to 10 i.e. Team1, Team2, Venue, Home_ Team, Toss_Winner, Toss_Decision, Player_of_Match, Team_Batting_First, Team_Batting_Second and Winning_Team. Here the Winnning_Team is the classifier. The next two sections describe the wrapper method and the ranker method in detail.

3.3.1 Wrapper Method

The wrapper method uses a subset evaluator algorithm like CFSSubsetEval and WrapperSubsetEval in combination with a searching technique such as Best First search or Greedy search to evaluate the subset of attributes. Behind the scenes, the wrapper method first creates all the possible subsets from the set of available attributes. Then it makes use of a classification algorithm like J48, Naïve Bayes, etc. to predict the classifier from the features available in each subset. Then it calculates the accuracy of prediction for each of the subsets. After comparing the accuracy of each subset, this method then chooses the best subset of attributes whose accuracy is the highest. The output of the wrapper method of attribute selection return us with a set of 4 attributes as seen below.

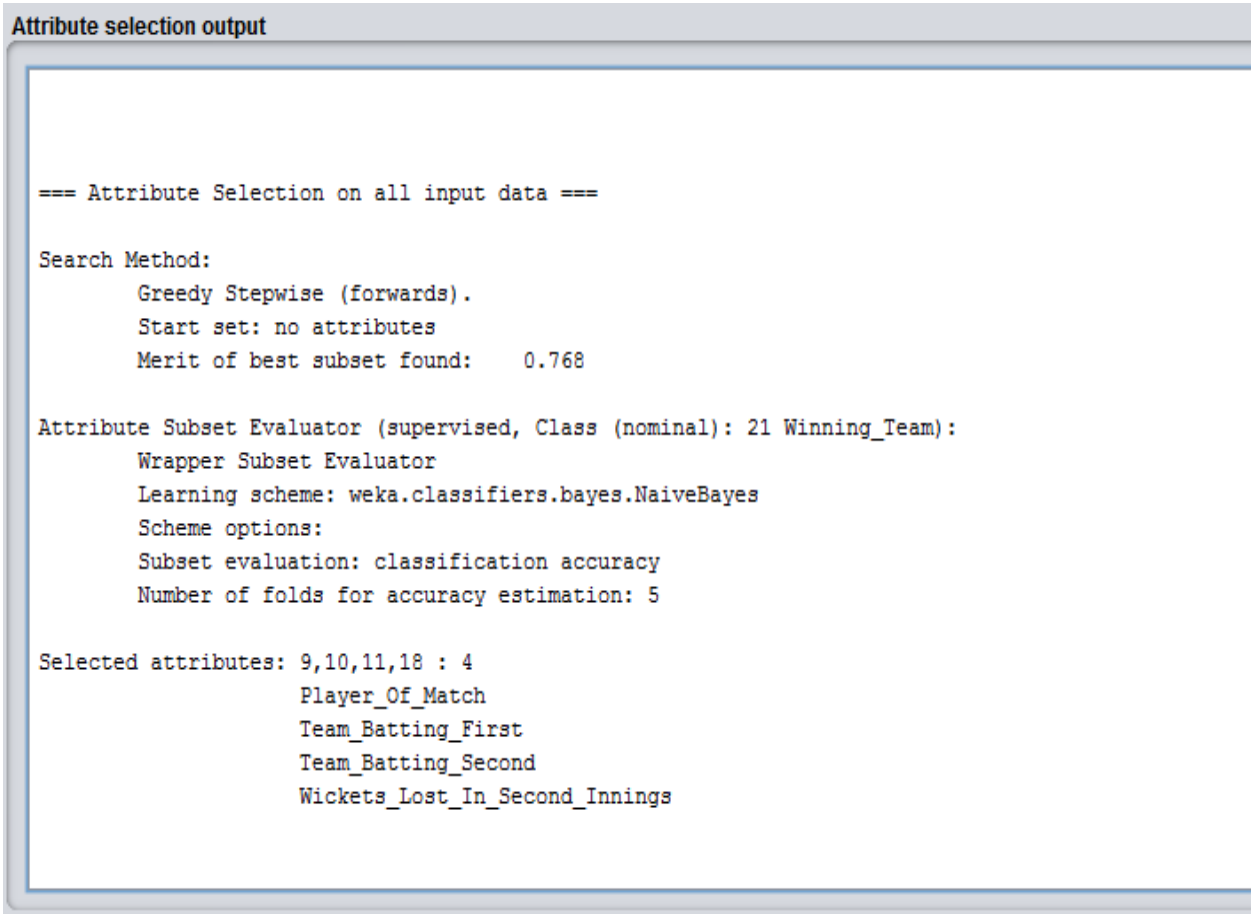


Figure 4: Attribute Selection by Wrapper Method

3.3.2 Ranker Method

The ranker method uses an attribute evaluator algorithm like InfoGainAttributeEval and GainRatioAttributEval and a ranker to rank all the available attributes in the set of attributes. We can set the number of attributes that we want to select from the ranked set of attributes. This helps us to automatically eliminate the lower ranked attributes. The output of the ranker method of attribute selection returns us with a ranked set of 6 attributes as seen below.

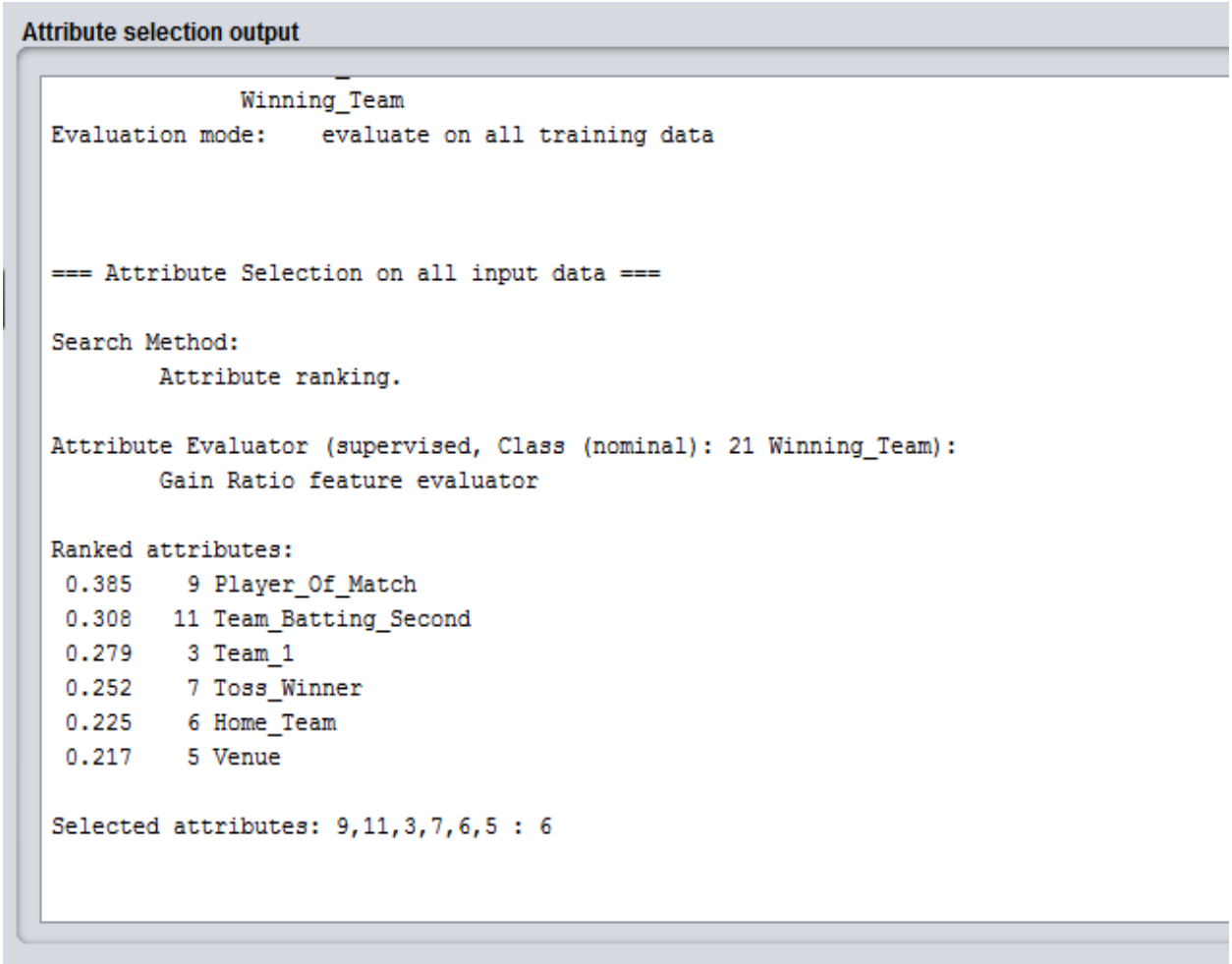


Figure 5: Attribute Selection by Ranker Method

3.4 Data Mining

The prediction part is done using the data mining algorithms Decision Tree (J48), Random Forest, Naïve Bayes and K-Nearest Neighbor in WEKA. I have used two methods in WEKA here i.e. the Percentage Split and the K-Fold Cross Validation methods to perform the prediction on the results of the games. The Percentage Split method divides the data set into two different sets i.e. Training Set and the Test Set. The model is built using the Training data set and the model is then tested on the Test data set. The accuracy of the model is calculated based on the correctly classified instances in the Test data set. The Percentage Split that I have used in all the data mining algorithms is 70% Training data and 30% Test data.

The K-Fold Cross Validation method divides the entire data set into K different subsets and each subset has a corresponding training data set and a test data set which is created internally. The classification algorithm is then executed K times on the different training data of the K subsets to build a model. The model is then tested for accuracy K times on the different test data of the K subsets. Thus, this method gives the best accuracy on the data set for a particular algorithm. I have used the K as 10 in all the data mining algorithms. The following sections describe the data mining algorithms with their results.

3.4.1 Decision Tree

The Decision Tree algorithm is the most common data mining algorithm that is used in predicting the classifier. WEKA uses J48 algorithm as the decision tree algorithm. When we provide the J48 algorithm with a set of attributes, it builds a tree-like model based on the various rules or decisions that it makes on the data set. Here, the leaf nodes represent the class attribute, the internal nodes represent the remaining attributes used for predicting the classifier and the branches represent the value of those attributes. The J48 algorithm with a Percentage Split method divides the data set into 70% Training data and 30% Test data and gives an accuracy rate of 31.07%. This accuracy improves drastically when we use the 10-Fold Cross Validation method. The results of both the methods can be seen below.

Classifier output

```
=== Summary ===

Correctly Classified Instances          55               31.0734 %
Incorrectly Classified Instances       122               68.9266 %
Kappa statistic                          0.227
Mean absolute error                      0.1253
Root mean squared error                  0.2991
Relative absolute error                 77.4137 %
Root relative squared error            104.8115 %
Total Number of Instances              177

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.727    0.052    0.667      0.727    0.696      0.651    0.842     0.485     Team Bangalore
               0.500    0.427    0.130      0.500    0.206      0.047    0.623     0.333     Team Chennai
               0.000    0.123    0.000      0.000    0.000      -0.105   0.527     0.091     Team Delhi
               0.000    0.000    0.000      0.000    0.000      0.000    0.500     0.051     Team Gujarat
               0.042    0.026    0.200      0.042    0.069      0.032    0.566     0.160     Team Hyderabad
               0.000    0.011    0.000      0.000    0.000      0.000    ?         ?         Team Kochi
               0.222    0.000    1.000      0.222    0.364      0.442    0.635     0.390     Team Kolkata
               0.458    0.052    0.579      0.458    0.512      0.449    0.778     0.454     Team Mumbai
               0.000    0.029    0.000      0.000    0.000      -0.029   0.501     0.029     Team Pune
               0.158    0.019    0.500      0.158    0.240      0.238    0.741     0.242     Team Punjab
               0.615    0.030    0.615      0.615    0.615      0.585    0.874     0.434     Team Rajasthan
Weighted Avg.  0.311    0.080    0.455      0.311    0.315      0.278    0.679     0.309
```

Figure 6: J48 Result with Percentage Split

Here, the J48 algorithm could only classify 55 instances correctly out of the total 177 Test instances. So, the accuracy rate achieved in the Percentage Split method was very low at 31.07%.

Classifier output

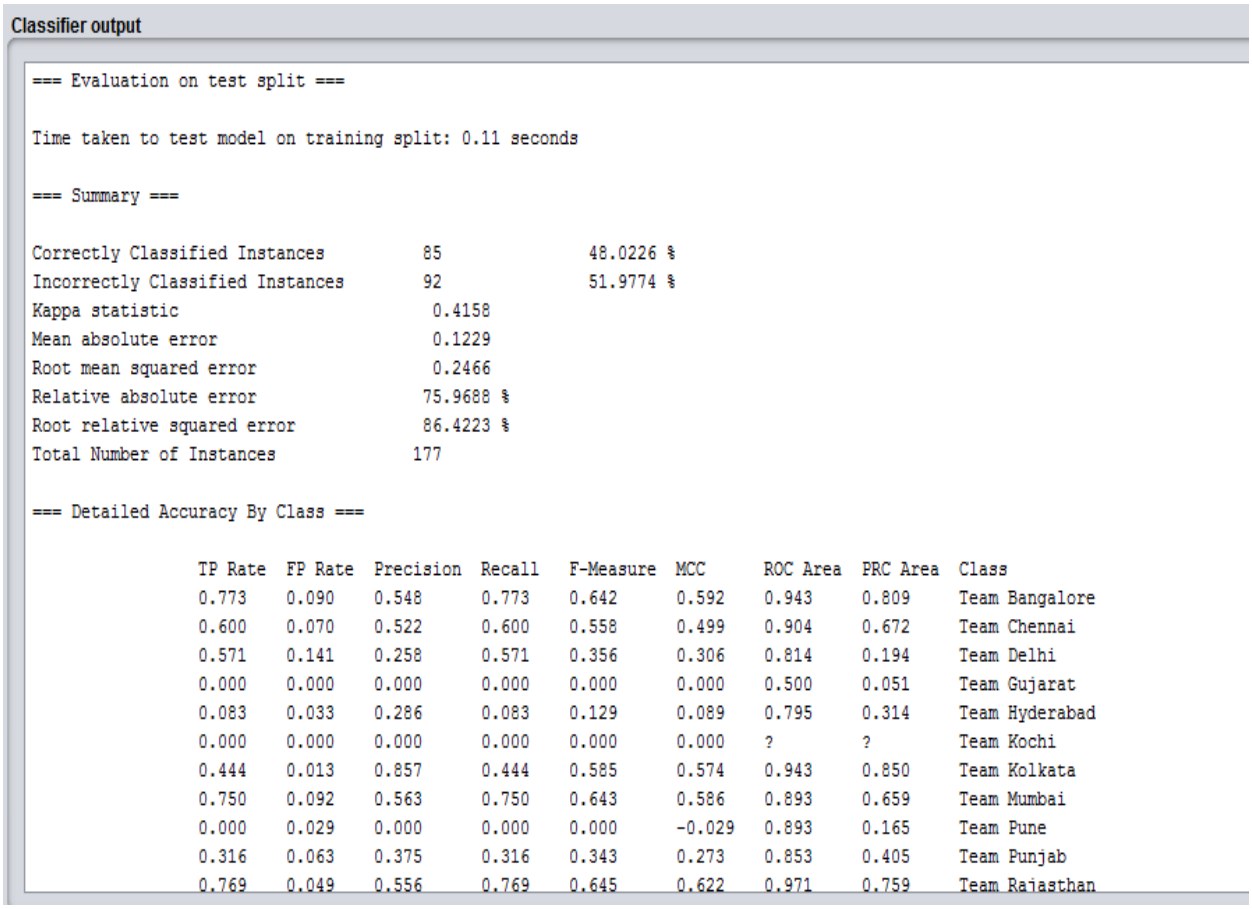
```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         363               63.2404 %
Incorrectly Classified Instances       211               36.7596 %
Kappa statistic                          0.582
Mean absolute error                      0.0754
Root mean squared error                  0.22
Relative absolute error                 46.7855 %
Root relative squared error             77.5179 %
Total Number of Instances              574

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.783    0.046    0.701      0.783    0.740      0.703    0.877     0.609     Team Bangalore
               0.785    0.048    0.721      0.785    0.752      0.711    0.923     0.732     Team Chennai
               0.246    0.035    0.438      0.246    0.315      0.275    0.776     0.268     Team Delhi
               0.556    0.007    0.556      0.556    0.556      0.548    0.727     0.289     Team Gujarat
               0.508    0.029    0.681      0.508    0.582      0.546    0.864     0.501     Team Hyderabad
               0.167    0.005    0.250      0.167    0.200      0.197    0.879     0.204     Team Kochi
               0.594    0.030    0.732      0.594    0.656      0.619    0.879     0.571     Team Kolkata
               0.850    0.176    0.439      0.850    0.579      0.526    0.918     0.807     Team Mumbai
               0.235    0.004    0.667      0.235    0.348      0.386    0.766     0.199     Team Pune
               0.578    0.020    0.787      0.578    0.667      0.641    0.857     0.552     Team Punjab
               0.738    0.019    0.818      0.738    0.776      0.752    0.921     0.644     Team Rajasthan
Weighted Avg.  0.632    0.051    0.657      0.632    0.625      0.591    0.874     0.579
```

Figure 7: J48 Result with 10-Fold Cross Validation

Here, the J48 algorithm could classify 363 instances correctly out of the total 574 instances. This shows that the 10-Fold Cross Validation method works on the entire data set rather than just the test data set. So, the accuracy rate achieved in the 10-Fold Cross Validation method was 63.24%. This shows that the accuracy rate doubled as compared to the one in Percentage Split.

3.4.2 Random Forest

The Random Forest algorithm or in other words the Random Decision Forests algorithm are an ensemble classification algorithm which create many decision trees during the training phase and then predicts the classifier for all those decision trees and thereby gives the best possible accuracy rate from all the decision trees. The Random Forest algorithm with a Percentage Split method divides the data set into 70% Training data and 30% Test data and gives an accuracy rate of 48.02%. This accuracy improves drastically when we use the 10-Fold Cross Validation method. The results of both the methods can be seen below.



Classifier output

```
=== Evaluation on test split ===

Time taken to test model on training split: 0.11 seconds

=== Summary ===

Correctly Classified Instances          85               48.0226 %
Incorrectly Classified Instances        92               51.9774 %
Kappa statistic                          0.4158
Mean absolute error                      0.1229
Root mean squared error                  0.2466
Relative absolute error                 75.9688 %
Root relative squared error             86.4223 %
Total Number of Instances              177

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.773    0.090    0.548      0.773    0.642      0.592    0.943     0.809     Team Bangalore
               0.600    0.070    0.522      0.600    0.558      0.499    0.904     0.672     Team Chennai
               0.571    0.141    0.258      0.571    0.356      0.306    0.814     0.194     Team Delhi
               0.000    0.000    0.000      0.000    0.000      0.000    0.500     0.051     Team Gujarat
               0.083    0.033    0.286      0.083    0.129      0.089    0.795     0.314     Team Hyderabad
               0.000    0.000    0.000      0.000    0.000      0.000    ?         ?         Team Kochi
               0.444    0.013    0.857      0.444    0.585      0.574    0.943     0.850     Team Kolkata
               0.750    0.092    0.563      0.750    0.643      0.586    0.893     0.659     Team Mumbai
               0.000    0.029    0.000      0.000    0.000      -0.029   0.893     0.165     Team Pune
               0.316    0.063    0.375      0.316    0.343      0.273    0.853     0.405     Team Punjab
               0.769    0.049    0.556      0.769    0.645      0.622    0.971     0.759     Team Rajasthan
```

Figure 8: Random Forest Result with Percentage Split

Here, the Random Forest algorithm could only classify 85 instances correctly out of the total 177 Test instances. So, the accuracy rate achieved in the Percentage Split method was low at 48.02%.

Classifier output

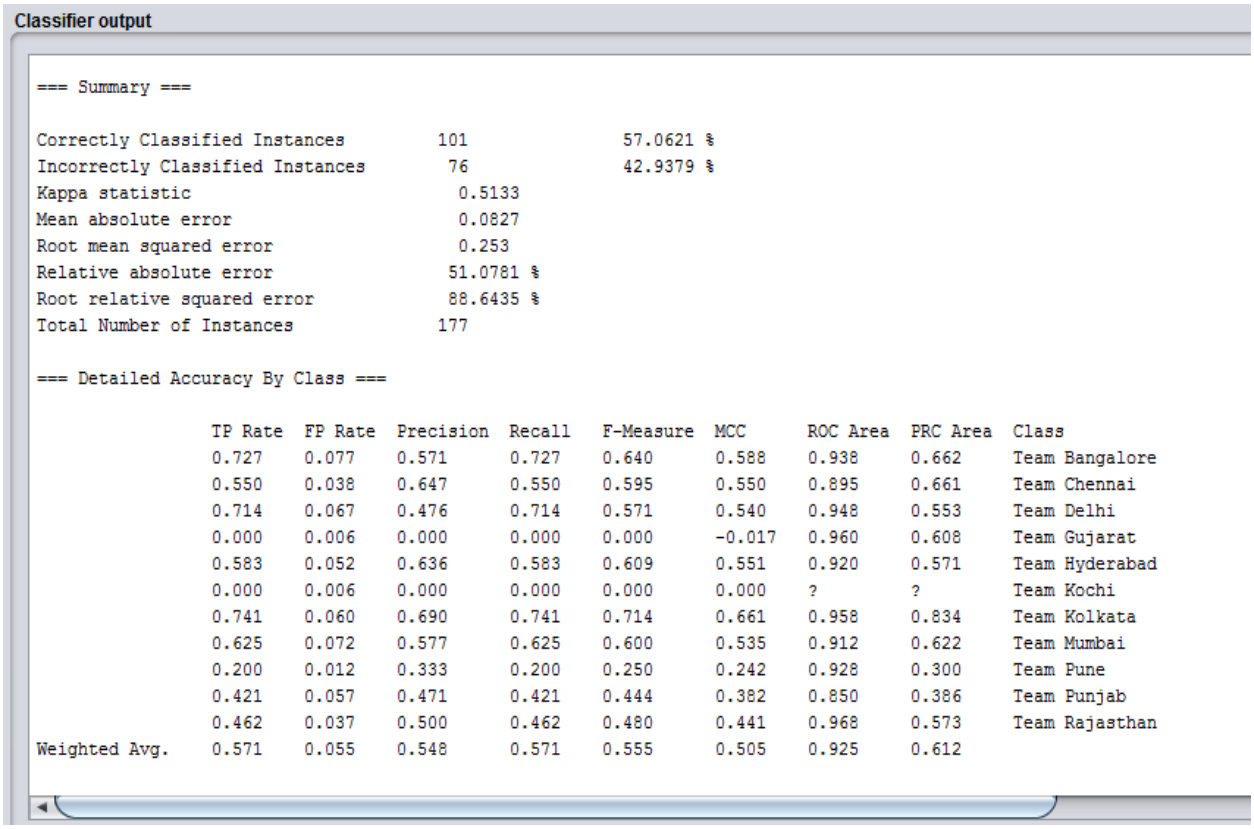
```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         408               71.0801 %
Incorrectly Classified Instances       166               28.9199 %
Kappa statistic                          0.6726
Mean absolute error                      0.0943
Root mean squared error                  0.1991
Relative absolute error                 58.5285 %
Root relative squared error             70.1527 %
Total Number of Instances              574

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               0.783     0.048     0.692      0.783     0.735      0.698     0.951     0.824     Team Bangalore
               0.924     0.051     0.745      0.924     0.825      0.800     0.988     0.939     Team Chennai
               0.439     0.048     0.500      0.439     0.467      0.414     0.906     0.544     Team Delhi
               0.667     0.009     0.545      0.667     0.600      0.596     0.949     0.715     Team Gujarat
               0.571     0.031     0.692      0.571     0.626      0.588     0.936     0.645     Team Hyderabad
               0.333     0.002     0.667      0.333     0.444      0.468     0.986     0.414     Team Kochi
               0.710     0.032     0.754      0.710     0.731      0.696     0.956     0.804     Team Kolkata
               0.813     0.028     0.823      0.813     0.818      0.788     0.980     0.916     Team Mumbai
               0.176     0.007     0.429      0.176     0.250      0.261     0.935     0.349     Team Pune
               0.656     0.037     0.689      0.656     0.672      0.632     0.929     0.755     Team Punjab
               0.869     0.033     0.757      0.869     0.809      0.787     0.982     0.891     Team Rajasthan
Weighted Avg.  0.711     0.037     0.702      0.711     0.701      0.669     0.955     0.782
```

Figure 9: Random Forest Result with 10-Fold Cross Validation

Here, the Random Forest algorithm could classify 408 instances correctly out of the total 574 instances. This shows that the 10-Fold Cross Validation method works on the entire data set rather than just the test data set. So, the accuracy rate achieved in the 10-Fold Cross Validation method was 71.08%. This shows that the accuracy rate improved as compared to the one in Percentage Split.

3.4.3 Naïve Bayes

The Naïve Bayes algorithm is a probabilistic classification algorithm which is based on the Bayes Theorem of conditional probability. The Naïve Bayes algorithm assumes that all the attributes in the data set are independent of each other and then computes the probabilities of each attribute. The Naïve Bayes algorithm with a Percentage Split method divides the data set into 70% Training data and 30% Test data and gives an accuracy rate of 57.06%. Here, this accuracy improves only slightly when we use the 10-Fold Cross Validation method. The results of both the methods can be seen below.



Classifier output

```
=== Summary ===

Correctly Classified Instances         101               57.0621 %
Incorrectly Classified Instances        76               42.9379 %
Kappa statistic                          0.5133
Mean absolute error                      0.0827
Root mean squared error                  0.253
Relative absolute error                 51.0781 %
Root relative squared error             88.6435 %
Total Number of Instances              177

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.727    0.077    0.571      0.727    0.640      0.588    0.938     0.662     Team Bangalore
                 0.550    0.038    0.647      0.550    0.595      0.550    0.895     0.661     Team Chennai
                 0.714    0.067    0.476      0.714    0.571      0.540    0.948     0.553     Team Delhi
                 0.000    0.006    0.000      0.000    0.000      -0.017   0.960     0.608     Team Gujarat
                 0.583    0.052    0.636      0.583    0.609      0.551    0.920     0.571     Team Hyderabad
                 0.000    0.006    0.000      0.000    0.000      0.000    ?         ?         Team Kochi
                 0.741    0.060    0.690      0.741    0.714      0.661    0.958     0.834     Team Kolkata
                 0.625    0.072    0.577      0.625    0.600      0.535    0.912     0.622     Team Mumbai
                 0.200    0.012    0.333      0.200    0.250      0.242    0.928     0.300     Team Pune
                 0.421    0.057    0.471      0.421    0.444      0.382    0.850     0.386     Team Punjab
                 0.462    0.037    0.500      0.462    0.480      0.441    0.968     0.573     Team Rajasthan
Weighted Avg.    0.571    0.055    0.548      0.571    0.555      0.505    0.925     0.612
```

Figure 10: Naïve Bayes Result with Percentage Split

Here, the Naïve Bayes algorithm could classify 101 instances correctly out of the total 177 Test instances. So, the accuracy rate achieved in the Percentage Split method was low at 57.06%.

Classifier output

```
=== Summary ===

Correctly Classified Instances         350               60.9756 %
Incorrectly Classified Instances       224               39.0244 %
Kappa statistic                          0.5596
Mean absolute error                      0.073
Root mean squared error                  0.241
Relative absolute error                 45.3259 %
Root relative squared error             84.9019 %
Total Number of Instances              574

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.609    0.057    0.592      0.609    0.600      0.545    0.927     0.653     Team Bangalore
                 0.684    0.048    0.692      0.684    0.688      0.638    0.955     0.748     Team Chennai
                 0.544    0.064    0.484      0.544    0.512      0.456    0.897     0.461     Team Delhi
                 0.778    0.004    0.778      0.778    0.778      0.774    0.957     0.636     Team Gujarat
                 0.476    0.061    0.492      0.476    0.484      0.421    0.915     0.501     Team Hyderabad
                 0.500    0.004    0.600      0.500    0.545      0.543    0.991     0.530     Team Kochi
                 0.710    0.048    0.671      0.710    0.690      0.647    0.928     0.735     Team Kolkata
                 0.638    0.049    0.680      0.638    0.658      0.605    0.942     0.707     Team Mumbai
                 0.471    0.018    0.444      0.471    0.457      0.440    0.936     0.297     Team Pune
                 0.469    0.057    0.508      0.469    0.488      0.427    0.890     0.525     Team Punjab
                 0.738    0.031    0.738      0.738    0.738      0.707    0.956     0.766     Team Rajasthan
Weighted Avg.    0.610    0.049    0.611      0.610    0.610      0.561    0.929     0.633

=== Confusion Matrix ===
```

Figure 11: Naïve Bayes Result with 10-Fold Cross Validation

Here, the Naïve Bayes algorithm could classify 350 instances correctly out of the total 574 instances. This shows that the 10-Fold Cross Validation method works on the entire data set rather than just the test data set. So, the accuracy rate achieved in the 10-Fold Cross Validation method was 60.97%. This shows that the accuracy rate improved only slightly as compared to the one in Percentage Split.

3.4.4 K-Nearest Neighbor

The K-Nearest Neighbor algorithm also called as KNN consists of the inputs of the K closest training attributes in the attributes set. The class of the instance is predicted on the basis of the majority votes of all the neighbors of that instance. The class which is the most common among all the K neighbors of the instance is assigned to that instance. The KNN algorithm with a Percentage Split method divides the data set into 70% Training data and 30% Test data and gives an accuracy rate of 52.54%. Here, this accuracy does not improve even slightly when we use the 10-Fold Cross Validation method. The results of both the methods can be seen below.

Classifier output

```
=== Summary ===

Correctly Classified Instances          93               52.5424 %
Incorrectly Classified Instances        84               47.4576 %
Kappa statistic                          0.4625
Mean absolute error                      0.0973
Root mean squared error                  0.274
Relative absolute error                 60.1248 %
Root relative squared error             96.0121 %
Total Number of Instances              177

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.773    0.129    0.459      0.773    0.576      0.522    0.869     0.498     Team Bangalore
               0.650    0.045    0.650      0.650    0.650      0.605    0.857     0.458     Team Chennai
               0.643    0.104    0.346      0.643    0.450      0.411    0.820     0.277     Team Delhi
               0.000    0.000    0.000      0.000    0.000      0.000    0.140     0.032     Team Gujarat
               0.333    0.026    0.667      0.333    0.444      0.418    0.712     0.339     Team Hyderabad
               0.000    0.000    0.000      0.000    0.000      0.000    ?         ?         Team Kochi
               0.556    0.040    0.714      0.556    0.625      0.573    0.818     0.500     Team Kolkata
               0.708    0.085    0.567      0.708    0.630      0.569    0.851     0.483     Team Mumbai
               0.200    0.000    1.000      0.200    0.333      0.442    0.756     0.241     Team Pune
               0.368    0.082    0.350      0.368    0.359      0.280    0.743     0.348     Team Punjab
               0.462    0.024    0.600      0.462    0.522      0.494    0.827     0.426     Team Rajasthan
Weighted Avg.  0.525    0.061    0.544      0.525    0.508      0.466    0.775     0.400

=== Confusion Matrix ===
```

Figure 12: KNN with Percentage Split

Here, the KNN algorithm could only classify 93 instances correctly out of the total 177 Test instances. So, the accuracy rate achieved in the Percentage Split method was low at 52.54%.

Classifier output

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         295               51.3937 %
Incorrectly Classified Instances       279               48.6063 %
Kappa statistic                          0.4497
Mean absolute error                      0.0919
Root mean squared error                  0.272
Relative absolute error                 57.0279 %
Root relative squared error             95.8344 %
Total Number of Instances              574

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.623    0.089    0.489      0.623    0.548      0.482    0.811     0.466     Team Bangalore
               0.671    0.061    0.639      0.671    0.654      0.598    0.877     0.558     Team Chennai
               0.368    0.072    0.362      0.368    0.365      0.295    0.715     0.328     Team Delhi
               0.556    0.002    0.833      0.556    0.667      0.676    0.727     0.603     Team Gujarat
               0.429    0.057    0.482      0.429    0.454      0.392    0.788     0.364     Team Hyderabad
               0.000    0.002    0.000      0.000    0.000      -0.004   0.436     0.064     Team Kochi
               0.580    0.069    0.533      0.580    0.556      0.492    0.837     0.496     Team Kolkata
               0.575    0.073    0.561      0.575    0.568      0.497    0.789     0.423     Team Mumbai
               0.176    0.018    0.231      0.176    0.200      0.181    0.621     0.105     Team Pune
               0.328    0.067    0.382      0.328    0.353      0.280    0.760     0.312     Team Punjab
               0.590    0.041    0.632      0.590    0.610      0.566    0.870     0.538     Team Rajasthan
Weighted Avg.  0.514    0.063    0.507      0.514    0.508      0.448    0.798     0.429
```

Figure 13: KNN Result with 10-Fold Cross Validation

Here, the KNN algorithm could classify 295 instances correctly out of the total 574 instances. This shows that the 10-Fold Cross Validation method works on the entire data set rather than just the test data set. So, the accuracy rate achieved in the 10-Fold Cross Validation method was 51.39%. This shows that the accuracy rate did not improve even slightly as compared to the one in Percentage Split.

4. Results

The accuracy rate of the model on the test data set was used as the evaluation criteria. The table below gives the accuracy rate of each of the algorithm along with how it performed using both the Percentage Split method as well as the K-Fold Cross Validation method.

Algorithm	Percentage Split	K-Fold Cross Validation
Decision Tree (J48)	31.07%	63.24%
Random Forest	48.02%	71.08%
Naïve Bayes	57.06%	60.97%
KNN	52.54%	51.39%

Figure 14: Accuracy Rate of the 4 Algorithms

The model gives an accuracy rate of about 60% to 70% when using the K-Fold Cross Validation whereas the accuracy rate in the case of the Percentage Split is not very good. This is mainly because of the fact that in Percentage Split method, WEKA splits the data manually on the basis of the 70% training and 30% test split that I mentioned. The K-Cross Validation method overcomes this by dividing the data set into K different subsets and thereby computing the results for K different subsets. We can notice that the accuracy rate almost doubles in some algorithms as shown above. The Random Forest algorithm gives the best accuracy rate for the model with 71.08% accuracy rate.

5. Conclusion

The model which is used to predict the results of the matches was built successfully with an accuracy rate of about 60% to 70%. The list of attributes was cut down to 10 important ones out of the 21 attributes available in the data set by using the attribute selection algorithms. The 4 data mining algorithms that were performed on the model were J48, Random Forest, Naïve Bayes and KNN. The prediction results were better when K-Fold Cross Validation method was used as compared to the Percentage Split. The accuracy of the Random Forest algorithm was the best with 71.08%. Although the accuracy was between 60% and 70% but it was still low because of the fact that the total number of instances in the data set was 574 and the total number of classes were 11. We need at least 100 instances per class to identify the patterns in the data set and perform a prediction with a high accuracy rate. So, with 11 classes in the data set we needed at least 1100 instances to perform a prediction with a high accuracy rate. Since, the data set consisted of 574 instances; in future it may improve the accuracy with more number of instances in the data set because with a larger number of instances the model will have the flexibility to deduce better rules and identify more patterns in the data set as compared to with a lesser number of instances.

6. Future Work

There are some future works that can be done in order to improve this project.

- The data set can include some of the external factors like player injury, player fatigue, winning streak with a particular team, overall winning streak, average runs scored by a team against a particular team in previous matches, etc. and on the basis of these data, we can try to do the prediction and check to see if the accuracy improves.
- The prediction can also be done taking into consideration the performance of the players in the team like the total number of runs scored by a player in the tournament, the form guide of the player, the number of man of the match awards earned, etc. rather than only using a high level data about the different matches like toss winner, toss decision, home team, etc.
- There is no web/mobile application or UI that my project contains. So, a web/mobile application can be made which would take in the entire data set as input and display the prediction result for each of the instances to a pdf or text file.

7. References

[1] Dataset: http://cricsheet.org [Online accessed 02-September-2016]

[2] Trawinski, Krzysztof. "A fuzzy classification system for prediction of the results of the basketball games." *Fuzzy Systems (FUZZ), 2010 IEEE International Conference on*. IEEE, 2010.

[3] Haghighat, Maral, Hamid Rastegari, and Nasim Nourafza. "A review of data mining techniques for result prediction in sports." *Advances in Computer Science: an International Journal* 2.5 (2013): 7-12.

[4] Zdravevski, Eftim, and Andrea Kulakov. "System for Prediction of the Winner in a Sports Game." *ICT Innovations 2009*. Springer Berlin Heidelberg, 2010. 55-63.

[5] http://espncricinfo.com/ [Online accessed 20-September-2016]