IIT Bombay

Machine Learning Project

Cricket Match Result Prediction

Team Members :
Abhishek Kumar (153050043)
Akash Kumar (153050071)
Sankalp Rangare (153050087)
Swaresh Sankpal (153050085)

April 3, 2017

1 Introduction

Objective of this report is to prepare a model for predicting outcome of an *One Day International* Cricket match and to measure accuracy of model on the basis of historical data. We all know that ODI Cricket is very popular. What makes ODI so popular is not only its shorter duration but also unpredictability. Only a single player with the good hitting of 10 overs can change the mood of a game unlike test cricket, where to win game team has to dominate days not some overs. Not only temperament, calmness, experience but aggression and hitting matters a lot and this all make ODI most unpredictable format of the game. Just to support my point further, Bangladesh has won 101 ODI yet but could only manage to win 8 tests so far and all these 8 win comes in recent time. You can expect teams like Ireland, Afghanistan or Canada to win an ODI game but never in a test match. Unpredictability makes ODI most popular format of cricket. Although unpredictable but the outcome of a game depends on a lot of factors like toss, venue, weather conditions, ground previous results statistics, players experience and current form and teams previous results against the opponent. So it would be nice to use these parameters and to see up to what extent each particular parameter can affect the outcome and to prepare a model which can predict the outcome of a game.

Problem Statement: Build a model which predicts the output of an ODI Cricket match various data related to that match.

2 Datasets

Various sources of data sets are available online. Two main sources of cricket dataset which we found related are:

1. Kaggle
2. Cricsheet [5]

But Kaggle only had datasets of T-20 matches and our project's objective was to build a model for ODI matches. Cricsheet contains data of 1,415 ODI matches but the datasets contains only batsmen and bowlers name. Player's name who did not bat or bowl in that match is not present in Cricsheet's datasets, such players also play an important role in winning a match. So we wanted that the dataset must contains names of all the 11 players of a team. **Cricinfo [1]** is a very popular site for getting cricket related information. It contains statistics of all the matches year wise. To create our own dataset we **crawled** data from cricinfo website. We first crawled URLs of all the matches which were played in the time-span of 1990-2016.

Beautiful Soup [4] is a Python library for extracting data out of HTML and XML files very easily. We used Beautiful Soup to extract required data from each of ODI match's URL. Each row of our data-set contains following information:

- Team-1
- Team-2

- City
- Date
- Toss Winner
- First batting team
- Playing 11 of team-1
- Playing 11 of team-2

Above was our initial dataSet where the player name in playing 11 for each team is used. Initially, we started working with it. In the second approach we realized instead of having player name we should be using ranking of player at that time because ranking will signify player potential as batsman or bowler and also will describe his form at the time of match. For extracting player ranking we used ICC Men's ODI Cricket Rankin [7] as source. In site, ranking of top 100 batsman or bowler were present at the date of match. If any batsman or bowler is not present in that top 100 we give him 101 as ranking. The intuition behind this rule is that if a player is not present in top 100 batting or bowling ranking table then he is not likely to affect the result of a match, also often it is the case that good batsman is not a good bowler and vice versa. If a player is good in both batting and bowling then he will be present in table top list most likely. So, in our final dataset each player name is replaced by two rankings, his **current** batting and bowling rank. Our final dataset contains following features:

- Team-1
- Team-2
- City
- Toss Winner
- First batting team
- {Batting rank, Bowling Rank} for each player in Playing 11 of team-1
- {Batting rank, Bowling Rank} for each player in Playing 11 of team-2

Now our dataset has 50 features which we though to be large so in order to reduce it. We though to use overall(kind of average) team's batting and bowling rankings. Batting and bowling ranking of each team is calculated by using individual player's rankings. We used following formula to calculate these two features:

$$Rank = (1111 - \sum_{i=1}^{11} ith\ player\ rank) \div 11$$

Intuition behind subtracting each rank from 101 is to nullify the contribution of each player having rank greater than 100. Here we are assuming that any player not present in say top 100 bowler list is actually a batsman and does not bowl and vice versa. 1111 comes from the fact that 11 players are present in a team so combined rank which could occur is, $101 \times 11 = 1111$

3 Machine Learning Techniques

Our problem is a classification problem with 3 classes :

1. Win
2. Lose
3. Draw/Tie

The classification machine learning algorithms that we have implemented are

- Bayesian [2]:
    - Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set.
    - We have used Pandas library for mapping categorical values into numerical values.
    - We have used sklearn library to implement this method.
- Decision tree[3]:
    - Decision tree learning uses a decision tree as a predictive model which maps observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves).
    - The goal is to create a model that predicts the value of a target variable based on several input variables.
    - In our case the target variable is win/lose/draw in context of ODI cricket match and input variables are features generated from dataset.
- Gradient Boosting classification:
    - It starts with rough prediction and building series of decision trees.
    - Its gives weights to every model based on their accuracy.
    - At the end a consolidated result is generated.
- Support Vector Classifier:
    - It is very effective in high dimensional spaces.
    - Unlike other classifiers, it tries to find the best separating line.
    - Its a versatile method as different kernel function can be used as decision function.
- Logistic Regression:
    - It is a classification technique which involves linear discriminant(dividing plane).
    - It tries to predict the probability of the given input belongs to a certain class.
    - It is also used to solve multi-classification problems.

- K-Neighbour Classifier:
  - It is a non-parametric method used for both classification and regression.
  - Data points are plotted on a n-dimensional space(n is number of attributes).
  - For an unclassified data point, its k nearest neighbours class labels are noted and the label which occurs maximum number of times is given to unclassified data point.
- Extra-Tree Classifier:
  - This method implements meta estimator that fits many random decision trees.
  - It works on various sub-samples of datasets and uses averaging to improve accuracy and control over-fitting.
- Neural Network (Multilayer Perceptron Classifier)
  - This model optimizes the log-loss function using LBFGS, ADAM or stochastic gradient descent.
  - MLPClassifier trains iteratively since at each time step the partial derivatives of the loss function with respect to the model parameters are computed to update the parameters.
  - It can also have a regularization term added to the loss function that shrinks model parameters to prevent overfitting.
  - This implementation works with data represented as dense numpy arrays or sparse scipy arrays of floating point values.
  - We used MLP classfier module of sklearn neural network library.
  - Our Parameters :
    - Learning rate : 0.01
    - Solver for weight optimization : Adam (Adam is similar to SGD in a sense that it is a stochastic optimizer, but it can automatically adjust the amount to update parameters based on adaptive estimates of lower-order moments)
    - Hidden layer : Two hidden layers with 10 neurons in each layer

4 Workflow

Our work-flow in this project is distributed among various steps. First, we crawled data of ODI mathces which include players name, dates etc. Again we crawled data to find the player's batting and bowling ranking on the day of match. Then we created our final dataset file, on which we trained various model and tested the accuracy of our models using **Cross Validation**. Our work-flow is shown in the following diagram:

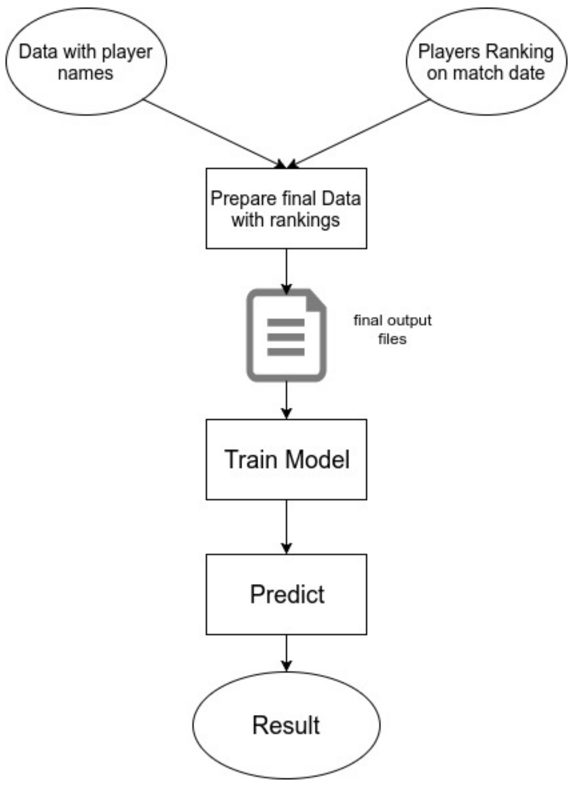

Figure 1: Work-flow of Project

5 Results

We implemented various ML classification algorithm using Sklearn package [8]. We used cross validation techniques to divide our data-set into 5 parts and calculated the **Accuracy** of the model. Following table gives a comparison of the accuracy obtained in the all the methods that we implemented.

Techniques	**Accuracy**
Bayesian	69.12
K-Neighbours Classifier	73.912
Decision-Tree-Classifier	74.12
Multilayer Perceptron	74.95
SVC	75.51
Tree Classifier	78.29
Gradient Boost	79.09
Logistic Regression	80.17

Table 1: Accuracy for different techniques implemented

To find the importance of all features We used a python script from scikit learn webpage [6], which plot the feature importance in decreasing order using *forests of trees.* Picture 2 shows the importance of all fetures in decreasing order. By looking at the figure 2.

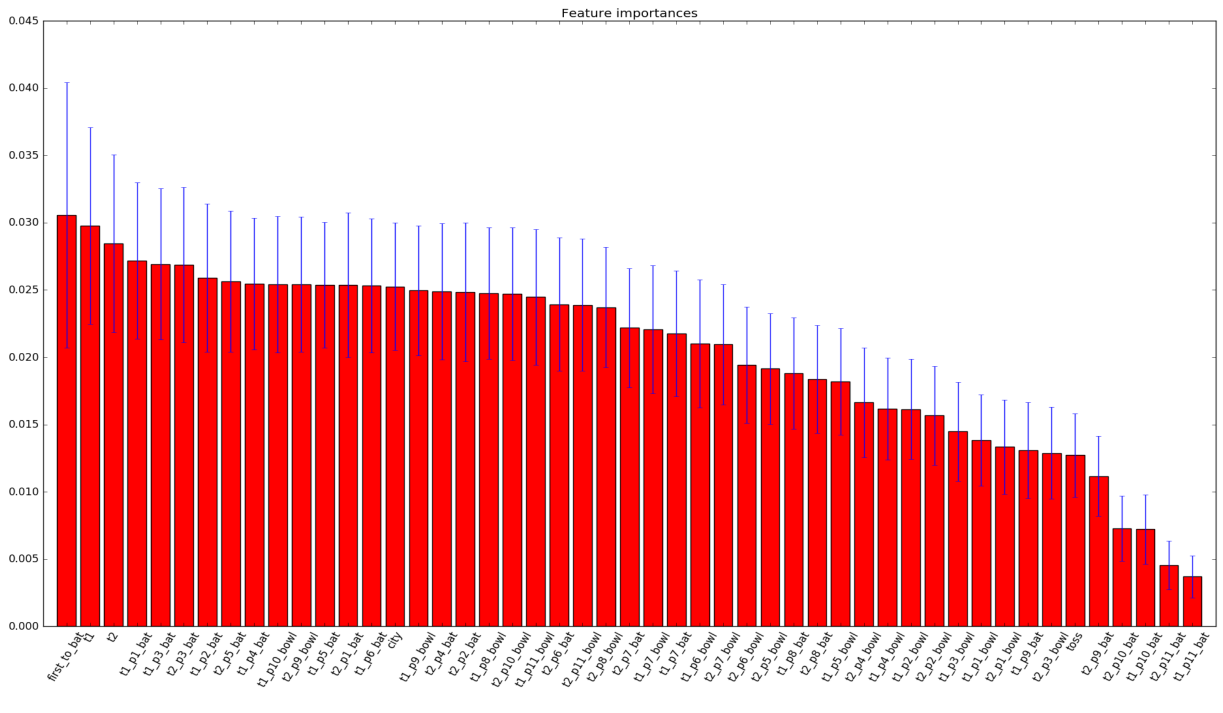

Figure 2: Feature Importances with forests of trees

As we can see from above image that batting rank of player number 10 and 11 is very less important and bowling rank of player number 1 and 2 is very less as compared to others. The above observation is quite practical because in a cricket team top players are pure batsmen and 10th and 11th players are pure bowlers so we don't expect lower order player's batting performance or higher order player's bowling performance to be important in a match. To verify this we actually dropped top 4 player's bowling rank and bottom 4 player's batting rank from our feature set and used different classification methods from sklearn library. The results obtained in newer version was almost same as it was in previous version, which implies that our feature importance graph 2 is correct.

6 Conclusion

As Cricket is a game which is full of uncertainties, still it depends on various factors mostly on players and some other conditions. Developing a model which would predict a match's output with accuracy around 90% or greater is very unrealistic. We used data of 27 years then our model predicted output with accuracy of 80%, which is quite good given the nature of the game.

We learnt so many things in Machine Learning area in order to complete this project. We got to know various libraries which provides implementations of various algorithms which would have been very time consuming to implement from scratch.

7 Contribution of Each Member

Contribution of each member in this project is as follows:

- Selection of features - Brainstorming by all members
- Akash - Search and finalisation for data sources , Data Crawling of year wise data from cricinfo
- Abhishek - Data Crawling of batting and bowling rankings from reliance icc ranking and Feature Engineering, Feature importance
- Sankalp Rangare : Implemented different classification techniques like NaiveBayes, SVM, Logistic regression and Gradient boosting.
- Swaresh Sankpal : Implemented different classification techniques like logistic decision tree, KNN, tree classifier, neural network (MLP).

References

[1] Crickinfo webpage available at `http://www.http://www.espncricinfo.com/`, retrieved March 2017.

[2] Naive Bayes classifier webpage available at `https://en.wikipedia.org/wiki/Naive_Bayes_classifier`, retrieved March 2017.

[3] Decision tree learning webpage available at `https://en.wikipedia.org/wiki/Decision_tree_learning`, retrieved March 2017.

[4] Beautiful Soup Documentation webpage available at `https://www.crummy.com/software/BeautifulSoup/bs4/doc/`, retrieved March 2017.

[5] Cricsheet webpage available at `http://cricsheet.org/`, retrieved March 2017.

[6] Feature importances with forests of trees webpage available at `http://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html`, retrieved April 2017.

[7] ICC Men's ODI Cricket webpage available at `http://www.relianceiccrankings.com/mensodi.php`, retrieved April 2017.

[8] Supervised Learning webpage available at `http://scikit-learn.org/stable/supervised_learning.html`, retrieved April 2017.