INTERNATIONAL JOURNAL OF SCIENTIFIC & ENGINEERING RESEARCH, VOLUME 9, ISSUE 8, AUGUST-2018
ISSN 2229-5518

Score Prediction and Player Classification Model in the Game of Cricket Using Machine Learning

Sonu Kumar, Sneha Roy

Abstract— Score prediction is something we always try in our sports life. An early prediction is always helpful for the team management to work on their plans quickly. Generally, prediction is always biased but this model attempts to predict the innings total in ODI after the first 5 over of the match.Player's selection is one of the most important tasks for any sports and thus cricket is not any exception. The performance of the player depends on various factors such as current form, record against the opposition, nature of the pitch, format of the game, venue etc. The team management and captain select 11 players out of 15-16 squad members. This model classifies the players based on their stats that who should play the limited over and which player should play in the test format of the game.For predictive model Linear Regression and MLPRegressor have been used and for classification KNN, SVM, Naïve Bayes, and MLPClassifier have been used.

Keywords— Cricket, KNN, Linear Regression, MLP, Naïve Bayes, Prediction Model, SVM.

1 INTRODUCTION

Cricket is like Religion in our Country. Every fan tries to predict the score and also they want the playing 11 according to their choice. Cricket is increasingly popular among the statistical science community, but the unpredictable and inconsistent natures of this game make it challenging to apply in common probability models. However, numerous researchers successfully applied various statistical methods to cricket data. I got inspiration from WASP (Winning and Score Prediction) model that was developed by Mr. Samuel back in 2011. ICC used the same model in 2012, and I noticed that for the first time in a match against India vs. New Zealand that WASP was predicting 282 according to model and if run rate would have been used then the score would have been 226 (go through the image) .

Fig. 1: WASP Model Prediction

- *Sonu Kumar is currently pursuing bachelors degree program in computer science and engineering in Dr. B. C. Roy Engineering College, Durgapur, India, PH-8609751381. E-mail: sonu1000raw@gmail.com*
- *Sneha Roy is currently pursuing bachelors degree program in computer science and engineering in Dr. B. C. Roy Engineering College, Durgapur, India, PH-8944921593. E-mail: sneha27feb@gmail.com*

I ask to Google about it but didn't understand a single line when I was in class 8-9. Later, I studied about it and came with these models and my implementation is still in process on the different datasets of Cricket (including Test and T20s). There are numerous factors that can affect a cricket game's score. In Cricket, It is believed that wickets in hand and current run rate are very important factor to get a good total.
Like in many sports, ODI cricket has both controllable and uncontrollable variables. Playing combination, in and out field tactics including aggressive and offensive playing behaviors may be considered controllable variables. However, coin-toss result is the main uncontrollable variables in the ODI format.

2 DATA AND TOOLS

I obtained all the data from different sources on the internet like www.cricinfo.com, www.cricbuzz.com and Wikipedia using web scrapping application that is developed by me in Node JS using "Cheerio" module. The data (for predictive modeling) contains the matches' information between the periods of 2006 to 2017. The data has many crucial factors which are important for the prediction of the inning total.

```
In [4]: mydata.head()
Out[4]:
```

	mid	date	venue	inning	bat_team	bowl_team	batsman	bowler	runs	wickets	overs	run_rate	runs_last_5	wickets_last_5	rr_last_5	striker	ns
0	1	2006-06-13	Civil Service Cricket Club, Stormont	1	England	Ireland	ME Trescothick	DT Johnston	0	0	0.1	0.0	0	0	0.0	0	
1	1	2006-06-13	Civil Service Cricket Club, Stormont	1	England	Ireland	ME Trescothick	DT Johnston	0	0	0.2	0.0	0	0	0.0	0	
2	1	2006-06-13	Civil Service Cricket Club, Stormont	1	England	Ireland	ME Trescothick	DT Johnston	4	0	0.3	8.0	4	0	8.0	0	

Fig. 2: ODI Dataset

For classification of players, I have considered Indian team squad statistics which is manually prepared from the ESPN cricinfo site.

```
In [5]: mydata.head()
Out[5]:
```

	id	name of the player	test strike rate	odi strike rate	test average	odi average	test no of not out	odi no of not out	test no of centuries	odi no of centuries	test balls played	odi balls played	test_or_odi_or_both
0	1	virat kohli	58.27	92.12	53.40	58.21	8	35	21	35	9532	10515	3
1	2	mahendra singh dhoni	59.11	88.13	38.09	51.26	16	78	6	10	8249	11399	2
2	3	rohit sharma	55.15	87.18	39.97	44.99	6	27	3	18	2682	7740	2
3	4	shikhar dhawan	68.00	93.88	43.94	45.72	1	6	7	13	3166	4773	2
4	5	lokesh rahul	57.00	78.59	40.86	32.13	1	3	4	1	2648	327	1

Fig. 3: Player Career Stats.

You can find the data on my Google Drive as follows:-
https://drive.google.com/open?id=16JoFayeALv_J7WQ9znA7TlTi8J4McUio

3 IMPLEMENTATION

3.1. PREDICTION MODEL

Currently we see that ICC uses Current run rate for the inning total and following image is the reason that why ICC should not use the existing system!! Run rate is no way near the best feature for the prediction of the inning total. There are many crucial factors that are must for the prediction. Before all that, we know that Prediction is always bias and Cricket is a game of uncertainty.

Fig. 4: Run Rate Based Score Prediction

```
In [53]: from matplotlib import pyplot as pt

In [58]: pt.plot(X_test[:10,[3]],y_test[:10],"r*")
         pt.plot(X_test[:10,[3]],ypMLP[:10])
         pt.xlabel("Run Rate")
         pt.ylabel("Total Run")
         pt.title("Run Rate vs Total Run")
         pt.show()
```

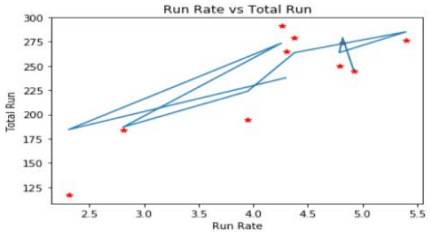

Fig. 5: Run Rate vs. Total Run

I have used 9 crucial features to predict the ODI inning total of a cricket match. The features are as follows:-

1. Current Runs Scored(at least 5 over of play)
2. Current Wickets fall
3. Current over
4. Current Run rate
5. Runs scored in last 5 over
6. Wickets fell in last 5 over
7. Run rate in last 5 over
8. Max(striker_runs_scored, non_striker_runs_scored)
9. Min(striker_runs_scored, non_striker_runs_scored)

All the above features are self-explanatory except the feature 8 and 9. This is an important feature because that matters as both the batsman are settled or new on the crease. The settled batsmen always score more and thus the inning total will be more.
Calculation of Current Run rate:-

Run rate= (Runs scored/Total no. of over bowled)

The model has a custom function which has a window of 20 runs. If the predicted scores matches the Actual scores then model is acceptable. The function is as follows:-

```
In [1]: def custom_accuracy(y_test,y_pred):
            right = 0

            l = len(y_pred)
            for i in range(0,l):
                if(abs(y_pred[i]-y_test[i]) <= 20):
                    right += 1
            print((right/l)*100)
```

Fig. 6: Accuracy Calculating Function

3.1.1. REGRESSION ANALYSIS

Multiple linear regression (MLR) is used to determine a mathematical relationship among a number of random variables. In other terms, MLR examines how multiple independent variables are related to one dependent variable. Once each of the independent factors have been determined to predict the dependent variable, the information on the multiple variables can be used to create an accurate prediction on the level of effect they have on the outcome variable. The model creates a relationship in the form of a straight line (linear) that best approximates all the individual data points.

The model for multiple linear regression is: $y_i = B_0 + B_1x_{i1} + B_2x_{i2} + ... + B_px_{ip} + E$.

The multiple regression model is based on the following assumptions:

- There is a linear relationship between the dependent variables and the independent variables
- The independent variables are not too highly correlated with each other
- y_i observations are selected independently and randomly from the population
- Residuals should be normally distributed with a mean of 0 and variance σ

Fig. 7: Linear Regression Algorithm Modeling

3.1.2. MULTILAYER PERCEPTRON

A multilayer perceptron (MLP) is a feed forward artificial neural network model that maps sets of input data onto a set of appropriate outputs.
Constructor Parameters

- inputLayerFeatures (int) - the number of input layer features
- hiddenLayers (array) - array with the hidden layers configuration, each value represent number of neurons in each layers
- classes (array) - array with the different training set classes (array keys are ignored)
- iterations (int) - number of training iterations
- learningRate (float) - the learning rate
- activationFunction (ActivationFunction) - neuron activation function

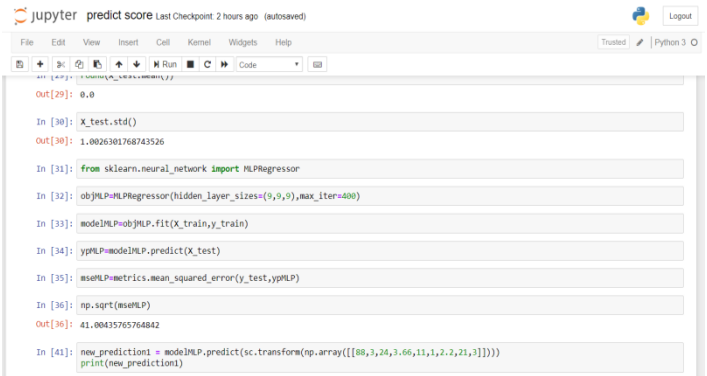

Fig. 8: Multilayer Perceptron Algorithm Modeling

3.1.3. RESULT

TABLE 1
RESULT TABLE OF PREDICTION MODELS

Match	Date	Result		
		Actual Score	Predicted Score	
			MLP	LR
India vs. Sri Lanka (Sri Lanka)	02/04/2011	274	256	250
Zimbabwe vs. Pakistan (Pakistan)	20/07/2018	399	390	418
Zimbabwe vs. Pakistan (Pakistan)	22/07/2018	364	359	377
West Indies vs. Bangladesh (West Indies)	23/07/2018	231	222	243

3.2. CLASSIFICATION MODEL

In Indian Cricket, There are players who play Test cricket only and players who play Limited over only (exception of players who play both the formats). Pujara is comfortable in the Test Cricket only because his strike rate is low, patience is high, and temperament is of that level. Virat Kohli can play all the formats of the game because his statistics shows everything. This classification model classifies the players based on their stats that which player should play the Test and Limited over Cricket.

Fig. 9: Virat Kohli Stats.

There are several factors which classifies the players (in both ODI and Test) as follows:-

1. Total No. of Balls played in his career
2. Total No. of Centuries Scored
3. Total Average
4. Total Strike Rate
5. Total No. of Not outs

The above factors are self explanatory for a cricket fan. A test player always plays more no. of balls, scored bigger runs, has more average, low strike rate, more no. of not outs (less in rare cases) than a ODI player.
The target has 3 classes as follows:-

1. Test only- 1
2. ODI only – 2
3. Test and ODI both – 3

For generating the classification models, we used supervised machine learning algorithms. In supervised learning algorithms, each training tuple is labeled with the class to which it belongs. We used Naïve Bayes, K-Nearest Neighbors, Multilayer Perceptron Classifier and Multiclass Support Vector Machines for our experiments. These algorithms are explained in brief.

3.2.1. NAÏVE BAYES

Bayesian classifiers are statistical classifiers that predict the probability with which a given tuple belongs to a particular class. 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.

Bayes Theorem: Let X be a data tuple and C be a class label. Let X belongs to class C, then

$$P(C|X) = P(X|C)P(C) / P(X)$$

where;

• P(C|X) is the posterior probability of class C given predictor X.
• P(C) is the prior probability of class.
• P(X|C) is the posterior probability of X given the class C.
• P(X) is the prior probability of predictor.

The classifier calculates P(C|X) for every class Ci for a given tuple X. It will then predict that X
belongs to the class having the highest posterior probability, conditioned on X. That is X belongs
to class Ci if and only if
$P(Ci|X) > P(Cj|X)$ for $1 \le j \le m, j \ne i$.

```
In [15]: from sklearn.naive_bayes import GaussianNB,BernoulliNB,MultinomialNB

In [16]: myobjg=GaussianNB()
         myobjb=BernoulliNB()
         myobjm=MultinomialNB()

In [17]: mygmodel=myobjg.fit(x_train,y_train)
         mybmodel=myobjb.fit(x_train,y_train)
         mymmodel=myobjm.fit(x_train,y_train)

In [18]: ypg=mygmodel.predict(x_test)
         ypb=mybmodel.predict(x_test)
         ypm=mymmodel.predict(x_test)
```

Fig. 10: Naïve Bayes Algorithm Modeling

3.2.2. SUPPORT VECTOR MACHINES

Vladimir Vapnik, Bernhard Boser and IsabellGuyon introduced the concept of support vector machine in their paper. 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. A separating hyperplane can be written as:

$$W \cdot X + b = 0$$

where W is a weight vector, W = {w1, w2, w3,..., wn}, n is the number of attributes and b is a scalar often referred to as a bias. If we input two attributes A1 and A2, training tuples are 2-D, (e.g., X = (x1, x2)), where x1 and x2 are the values of attributes A1 and A2, respectively. Thus, any points above the separating hyperplane belong to Class A1:

$$W \cdot X + b > 0$$

and any points below the separating hyperplane belong to Class A2:

$$W \cdot X + b < 0$$

```
In [4]: from sklearn.model_selection import train_test_split

In [5]: x_train,x_test,y_train,y_test=train_test_split(x_feature,y_target,test_size=.3,random_state=101)
        xx_train=x_train
        xx_test=x_test

In [6]: from sklearn.svm import SVC

In [7]: obj=SVC(gamma=.001)

In [8]: model=obj.fit(x_train,y_train)

In [9]: ypsvm=model.predict(x_test)
```

Fig. 11: Support Vector Machines Algorithm Modeling

3.2.3. K-NEAREST NEIGHBORS

In pattern recognition, the k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k-NN is used for classification or regression.

To determine which of the K instances in the training dataset are most similar to a new input a distance measure is used. For real-valued input variables, the most popular distance measure is Euclidean distance.
Euclidean distance is calculated as the square root of the sum of the squared differences between a new point (x) and an existing point (xi) across all input attributes j.

Euclidean Distance(x, xi) = sqrt(sum((xj – xij)^2))

When KNN is used for classification, the output can be calculated as the class with the highest frequency from the K-most similar instances. Each instance in essence votes for their class and the class with the most votes is taken as the prediction.

Class probabilities can be calculated as the normalized frequency of samples that belong to each class in the set of K most similar instances for a new data instance. For example, in a binary classification problem (class is 0 or 1):

p(class=0) = count(class=0) / (count(class=0)+count(class=1))

If you are using K and you have an even number of classes (e.g. 2) it is a good idea to choose a K value with an odd number to avoid a tie. And the inverse, use an even number for K when you have an odd number of classes.

Here, Cluster of "Total balls played", "No. of centuries", "strike rate" etc. can be mapped by KNN as it picks the nearest neighbor. KNN has application in recommendation system mostly.

```
In [10]: from sklearn.neighbors import KNeighborsClassifier
In [11]: myknn=KNeighborsClassifier()
In [13]: mymodel=myknn.fit(x_train,y_train)
In [14]: yp=mymodel.predict(x_test)
```

Fig. 12: K-Nearest Neighbors Algorithm Modeling

3.2.4. MULTILAYER PERCEPTRON CLASSIFIER

A multilayer perceptron (MLP) is a class of feed forward artificial neural network. An MLP consists of at least three layers of nodes. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called back propagation for training. Its multiple layers and non-linear activation distinguish MLP from a linear perceptron. It can distinguish data that is not linearly separable.

If a multilayer perceptron has a linear activation function in all neurons, that is, a linear function that maps the weighted inputs to the output of each neuron, then linear algebra shows that any number of layers can be reduced to a two-layer input-output model. In MLPs some neurons use a nonlinear activation function that was developed to model the frequency of action potentials, or firing, of biological neurons.

The two common activation functions are both sigmoid, and are described by

$$y(v_i) = \tanh(v_i) \text{ and } y(v_i) = (1 + e^{-v_i})^{-1}.$$

The first is a hyperbolic tangent that ranges from -1 to 1, while the other is the logistic function, which is similar in shape but ranges from 0 to 1. Here y_i is the output of the ith node (neuron) and v_i is the weighted sum of the input connections. Alternative activation functions have been proposed, including the rectifier and soft plus functions. More specialized activation functions include radial basis functions (used in radial basis networks, another class of supervised neural network models).

Learning occurs in the perceptron by changing connection weights after each piece of data is processed, based on the amount of error in the output compared to the expected result. This is an example of supervised learning, and is carried out through back propagation, a generalization of the least mean squares algorithm in the linear perceptron.

```
In [27]: from sklearn.preprocessing import StandardScaler
In [28]: myscaler=StandardScaler()
In [29]: myscaler.fit(xx_train)
Out[29]: StandardScaler(copy=True, with_mean=True, with_std=True)
In [30]: xx_train=myscaler.transform(xx_train)
In [31]: round(xx_train.mean())
Out[31]: -0.0
In [32]: xx_train.std()
Out[32]: 1.0
In [33]: myscaler.fit(xx_test)
Out[33]: StandardScaler(copy=True, with_mean=True, with_std=True)
In [34]: xx_test=myscaler.transform(xx_test)
In [35]: from sklearn.neural_network import MLPClassifier
In [51]: objMLP=MLPClassifier(hidden_layer_sizes=(10,10,10),max_iter=600)
In [52]: modelMLP=objMLP.fit(xx_train,y_train)
In [53]: ypmlp=modelMLP.predict(xx_test)
```

Fig. 13: Multilayer Perceptron Classifier Algorithm Modeling

3.2.5. RESULT

```
In [21]: from sklearn import metrics

In [22]: metrics.accuracy_score(ypsvm,y_test)
Out[22]: 0.3333333333333333

In [23]: metrics.accuracy_score(yp,y_test)
Out[23]: 0.16666666666666666

In [24]: metrics.accuracy_score(ypg,y_test)
Out[24]: 0.3333333333333333

In [25]: metrics.accuracy_score(ypb,y_test)
Out[25]: 0.3333333333333333

In [26]: metrics.accuracy_score(ypm,y_test)
Out[26]: 0.5

In [54]: metrics.accuracy_score(ypmlp,y_test)
Out[54]: 0.6666666666666666
```

Fig. 14: Efficiency of Different Classification Algorithms

4 CONCLUSION

The main limitation in carrying out this project was the limited dataset, which I had at my disposal. The next logical step in the direction to improve the accuracy of prediction problem at hand would be to test out the approaches and various methodologies proposed in this paper using a larger and more representative dataset. Also I would like to extend the features like Weather condition, Nature of the pitch and Venue. The accuracy will be even higher if Deep Neural network (Tensorflow, Keras and Thaeno) comes into the implementation. Not only this, Similar model can be developed for other sports like Tennis, NBA and Football and newer sports like Pro Kabaddi League.

ACKNOWLEDGMENT

First of all, I am thankful to International Journal of Scientific and Engineering Research to provide me a platform to display my research.
I would also like to extend my heartiest thankfulness to Mr. Chandan Kumar Verma for guiding me through the basics of machine learning so that I could come up with implementing this project. Through this project I have reflected on important aspects of Prediction and Classification Modeling, which when Predicted with the right amount of accuracy and classified with a very good efficiency, can help a lot in terms of prediction modeling and can prove to be extremely crucial in predicting the Inning score and players classification based on their stats.

REFERENCES

[1] ESPN cricinfo
http://www.espncricinfo.com/

[2] Linear Regression, Wikipedia
https://en.wikipedia.org/wiki/Linear_regression

[3] Multilayer Perceptron, Wikipedia
https://en.wikipedia.org/wiki/Multilayer_perceptron\

[4] Support Vector Machine, Wikipedia
https://en.wikipedia.org/wiki/Support_vector_machine

[5] K-Nearest Neighbors, Machine Learning Mastery
https://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/