The problem statement can be rephrased as follows:

Given a state of match till a certain point in a cricket match, predict the score at the end of 20 overs.

I defined **state** of a match using the following features:

1. **runs\_now**: total runs scored till that point

runs\_now = data.group\_by(‘match\_id’).cumsum()[‘total\_runs’]

1. **wickets\_now**: total wickets fallen for the batting team

Before running the below command, I modified the player\_dismissed column as a boolean field to denote if the batsman was dismissed at that ball or not.

wickets\_now = data.groupby(‘match\_id’).cumsum()[‘player\_dismissed’]

1. **balls\_done**: total balls done by the bowling team (including the extras)

balls\_done = (data[‘over’] - 1) \* 6 + data[‘ball’]

1. **run\_rate**: current scoring run rate(per over)

**run\_rate** = runs\_now / (ball\_done / 6)

1. **striker\_runs**: runs scored by the batsman at striker end.
2. **non\_striker\_runs**: runs scored by the batsman at non-striker end.

The batting and bowling team might affect the total score but since it is difficult to pass a string in a machine learning model, I have not considered them as features.

I tried one-hot-encoding the team(total = 14 teams), but the results were inferior to the earlier approach.

To predict the score at the end of 20 overs, I considered the last state for each match provided in the test dataset for prediction.

Random forest regressor is used for training on the dataset.

X = [runs\_now, wickets\_now, balls\_done, run\_rate, striker\_runs, non\_striker\_runs]

Y = total runs made by the team in 20 overs.