ML Hackathon Report

Date- 18/11/18

Group name- TBD

1)D.Soumith Kumar	IMT2016110
2)B.Nikhil Sai	IMT2016091
3)B.Praneeth	IMT2016109

Problem statement :

The aim is to train a model that can predict if an over is a high scoring one using the IPL match statistics from previous years. An over is said to be high scoring if the runs scored in that over is more than a threshold.

Dataset Used:

https://www.kaggle.com/raghu07/ipl-data-till-2017

This dataset conrains IPL statistics from 2008 to 2017.

There are five files in the dataset.

1. Ball_By_Ball.csv – Contains information about each ball bowled from all the matches across all seasons.
2. Match.csv – Contains details regarding each match like venues, teams playing, results etc.
3. Player.csv – Contains details of all the players who have played in the IPL.
4. Player_Match.csv – Contains player statistics from each match of the IPL.
5. Team.csv – Contains details of all the teams played in the IPL.

The data from the years 2008 to 2016 has been used for training and data of the year 2017 has been used for testing.

Preprocessing the data:

A lot of the data is redundant for the problem and the required data is spread over five files. We have extracted required data to prepare our dataset to train on.

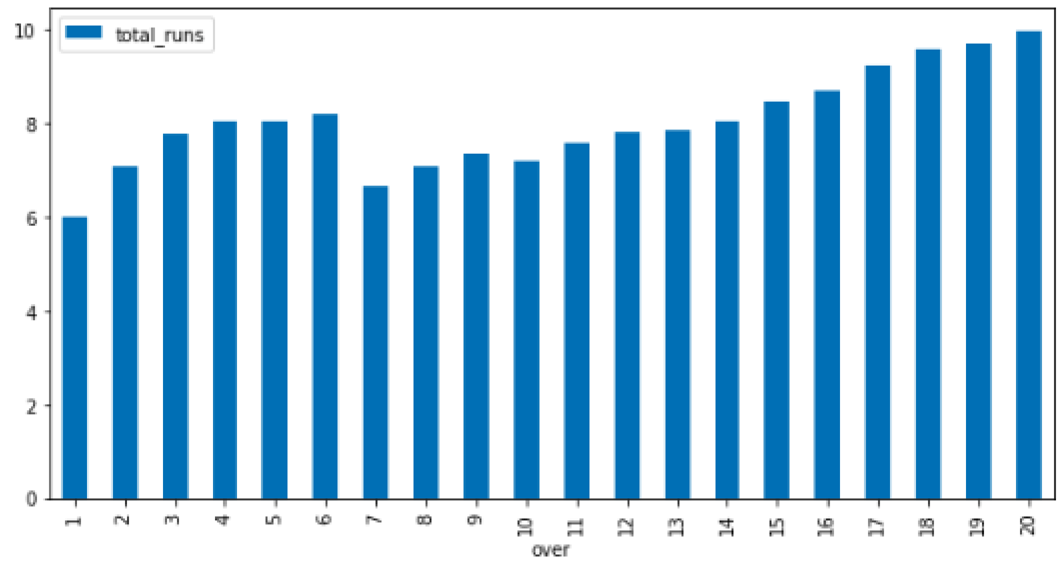
Firstly we have grouped the ball by ball data to get the over statistics as our prediction is based on over data. The following attributes have been extracted from the data.

Innings	-	First or second innings.
Over	-	The over number in the innings.
Runs_Scored	-	Runs scored by the batsmen in an over.
Extra_runs	-	Extras given away by the bowler.
Bowler	-	Id of the bowler.
Striker	-	Id of the batsmen on strike.
Strike_position	-	The position of the batsmen on strike in the batting order.
Playerout	-	Number of wickets fallen in that over.
Matchdate	-	Match day date (This can be used as a key to get match details).

Runs_Scored is calculated by adding the runs_scored column from ball_by_ball.csv for all the balls of an over. Same for Extra_runs and Player_out.
Strike may change during the over so we used the id of the batsman who faced majority of the balls.The same goes for the Strike_position.

Analysis of the dataset:

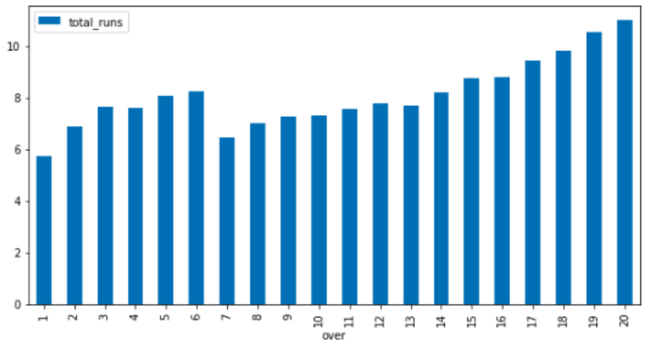
Ploting the mean of runs per over for each over.



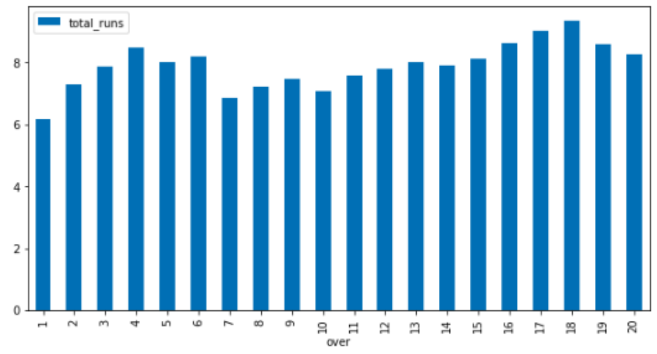
The drop in the runrate right after the power play i.e 6th over can be noted. The runrate increases as we go to the end of the innings. So the number of the over is going to have an effect on the output.

To make more sense out of it, we have plotted the graph for first and second innings.

Innings 1

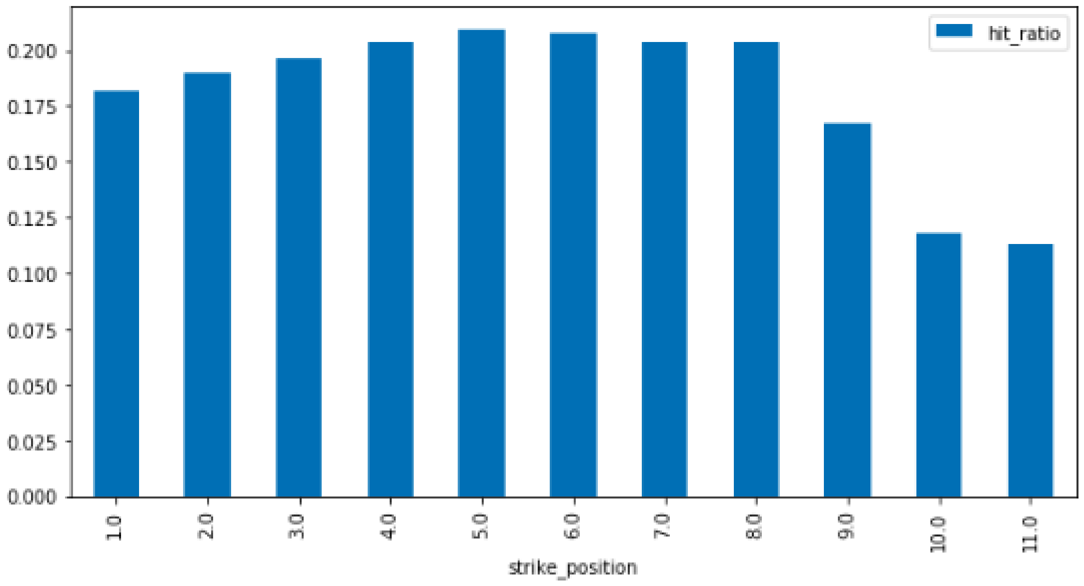


Innings 2



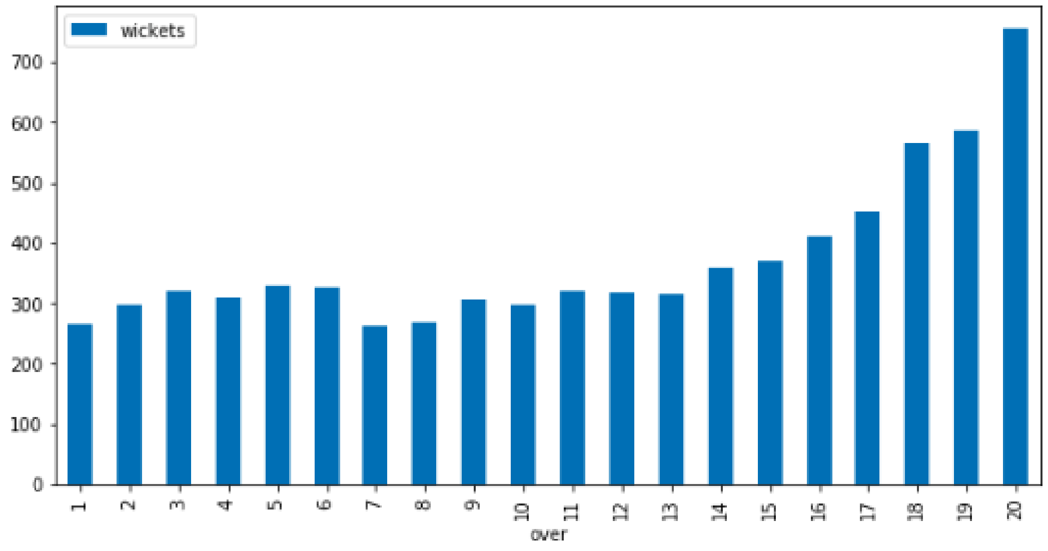
It can be observed that runrates tend to be lower in the begining for first innings when compared to second innings. This may be due to the fact that the team that's chasing has a better idea how much to score per over from the begining. So they try not to take it deep. Also towards the end, runrates of first innings tend to be lot higher than the second innings since players try to hit boundaries to end up with a high score. So the innings is definitly going to play a role in the output.

Looking at the other factors, the batsmen at the crease and the bowler make a large impact on the output. One of the ways to include this is to look at the position of the batsman at the crease in the batting order.

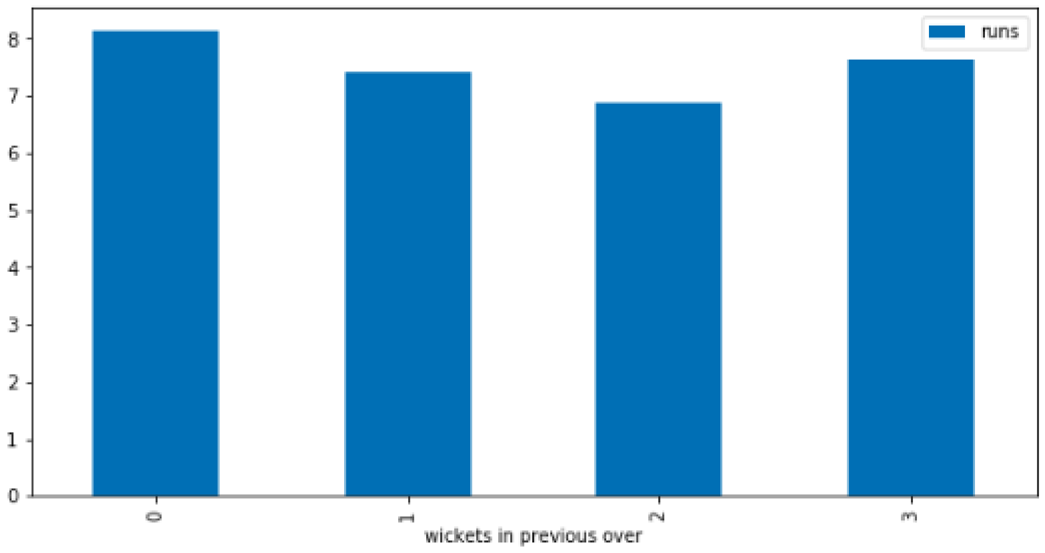


Hit rate is the ratio of high scoring overs played to the total number of overs played by the batsmen. Clearly the middle order batsmen tend to hit more. The ratio decreases as we go to the tailenders.

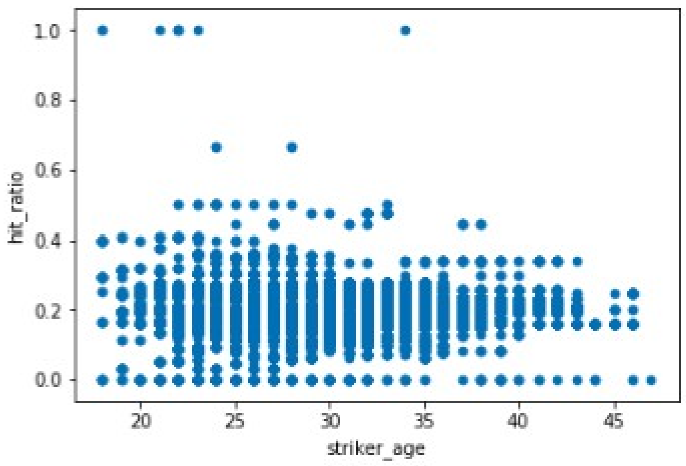
How do wickets effect the runs.



It can be seen that more wickets fall in the death overs. But we have seen typically runrates are more in death overs. So how do we factor this. To predict the outcome of an uver, we can look if the runrates decreases if a wicket falls in the previous over.

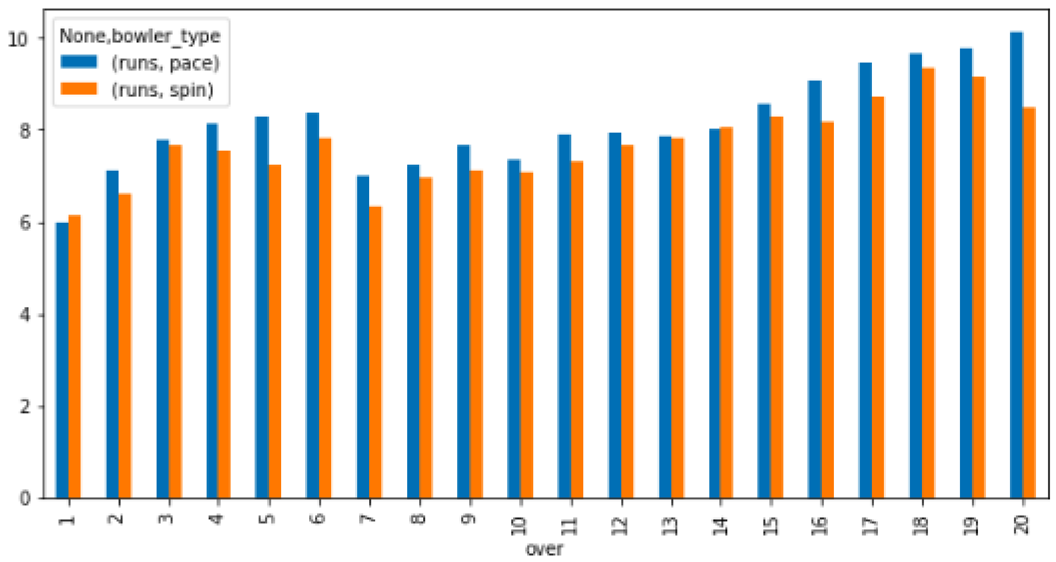


The mean runs per over decrease as more wickets fall in the previous over. But this trend breaks at 3 wickets. This seems odd but if we look at the data set, three wickets in an over has occured very few times. So at the third wicket, the runs are not a good average estimate. But still, following the trend at first two cases, this can impact the outcome.

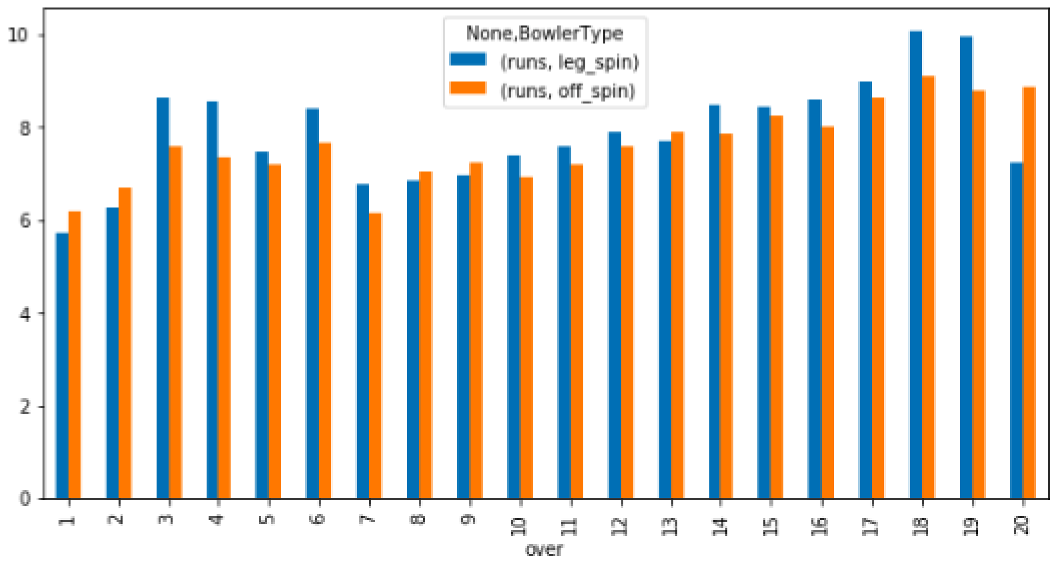


This plot shows the consistency of a batsman’s hitting with respect to his age. As the age increases the hit_ratio obviously reduces. It is maximum around 22-27 as that is the prime period for a cricketer.So this might have an effect on the output.

Looking at different kinds of bowlers.



It can be noticed that spinners typically give away less runs than a pacer. This can be justified as Indian pitches favour spinners.
Also since there are two kinds of spinners and typically leg spinners are more aggressive as they pickup more wickets and giveaway more runs, there might be some effect on our outcome.



This graph justifies our claim that leg spinners give away more runs.The deviations in 1st, 2nd and 20th overs are because there are very less instances (the cricket ball spins more when it becomes old) where the leg spinners bowled the respective overs.

Feature Engineering:

Apart from the features discussed above we can also get more obvious features by using the other data.

Firstly we consider the **features which do not depend on the previous overs** of the match. This model is more generalised and dosen't assume anything about the pitch conditions etc.

Batsman_skill contains the ratio of high scoring overs when the batsman was on strike to the experience. This has a major effect on the output as the outcome heavily depends on the batsman and the bowler. This shows the hitting ability of the batsman. Batsman_experience is the measure of the number of overs played by the batsman. This servs as a proxy for the importance of an experienced batsman.

Similary, bowler_experience and bowler_skill indicate total number of overs bowled by a bowler in IPL,ratio of total number of wickets taken by that bowler to experience respectively. They serve as a proxy for experience and lethality of a bowler.

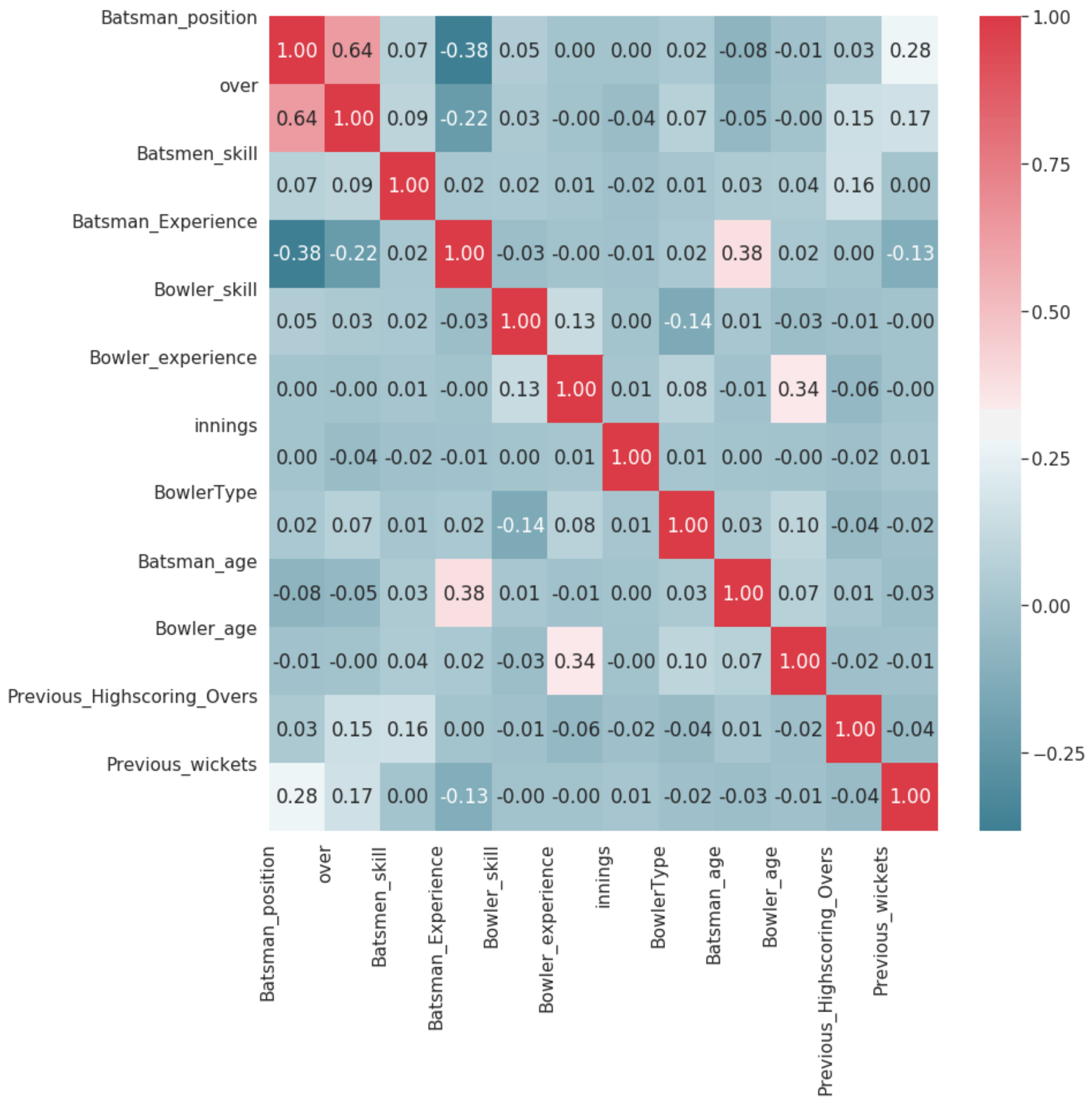
Batsman_age and bowler_age have the age of the player at the time of the match. This can be calculated from the DOB given in player.csv and the matchdate given in ball_by_ball.csv.

Then we look at how game has progressed until the over we are looking at by **including the features depending on the game**. This will help us capture the pitch and other game conditions on the day the game is played. This includes number of wickets fallen until that point and the number of high run overs in the game.

Typically pitch conditions play a major role in the outcome of a game. But pitches keep changing often very frequently. Adding a venue feature to the model makes it more generalised since a venue which is completly new can be choosen for a game. So instead we tried to make the model generalised by looking at the game as it progresses. So the model will be making more accurate predictions as the game progresses.

We kept a count of the high scoring overs until that over and used it as a feature. This has major effect on the accuracy as explained above. Also we kept a count of the number of fallen wickets in the previous over.

Correlation matrix:



	Batsman_position	over	Batsmen_skill	Batsman_Experience	Bowler_skill	Bowler_experience	innings	BowlerType	Batsman_age	Bowler_age	Previous_Highscoring_Overs	Previous_wickets
Batsman_position	1.00	0.64	0.07	-0.38	0.05	0.00	0.00	0.02	-0.08	-0.01	0.03	0.28
over	0.64	1.00	0.09	-0.22	0.03	-0.00	-0.04	0.07	-0.05	-0.00	0.15	0.17
Batsmen_skill	0.07	0.09	1.00	0.02	0.02	0.01	-0.02	0.01	0.03	0.04	0.16	0.00
Batsman_Experience	-0.38	-0.22	0.02	1.00	-0.03	-0.00	-0.01	0.02	0.38	0.02	0.00	-0.13
Bowler_skill	0.05	0.03	0.02	-0.03	1.00	0.13	0.00	-0.14	0.01	-0.03	-0.01	-0.00
Bowler_experience	0.00	-0.00	0.01	-0.00	0.13	1.00	0.01	0.08	-0.01	0.34	-0.06	-0.00
innings	0.00	-0.04	-0.02	-0.01	0.00	0.01	1.00	0.01	0.00	-0.00	-0.02	0.01
BowlerType	0.02	0.07	0.01	0.02	-0.14	0.08	0.01	1.00	0.03	0.10	-0.04	-0.02
Batsman_age	-0.08	-0.05	0.03	0.38	0.01	-0.01	0.00	0.03	1.00	0.07	0.01	-0.03
Bowler_age	-0.01	-0.00	0.04	0.02	-0.03	0.34	-0.00	0.10	0.07	1.00	-0.02	-0.01
Previous_Highscoring_Overs	0.03	0.15	0.16	0.00	-0.01	-0.06	-0.02	-0.04	0.01	-0.02	1.00	-0.04
Previous_wickets	0.28	0.17	0.00	-0.13	-0.00	-0.00	0.01	-0.02	-0.03	-0.01	-0.04	1.00

Model Building:

As mentioned earlier, we have trained the model in two ways namely, using features related to the match and with out them.

Using features which are not specific to the match and the threshold as 12 runs., the accuracy scores are as follows.

XG Boost	0.780541
Random Forest	0.772989
Logistic Regression	0.774322
SVM	0 .777432
KNN	0.777876

Using features which are specific to the match along with the previous features and the threshold as 12 runs, the accuracy scores are

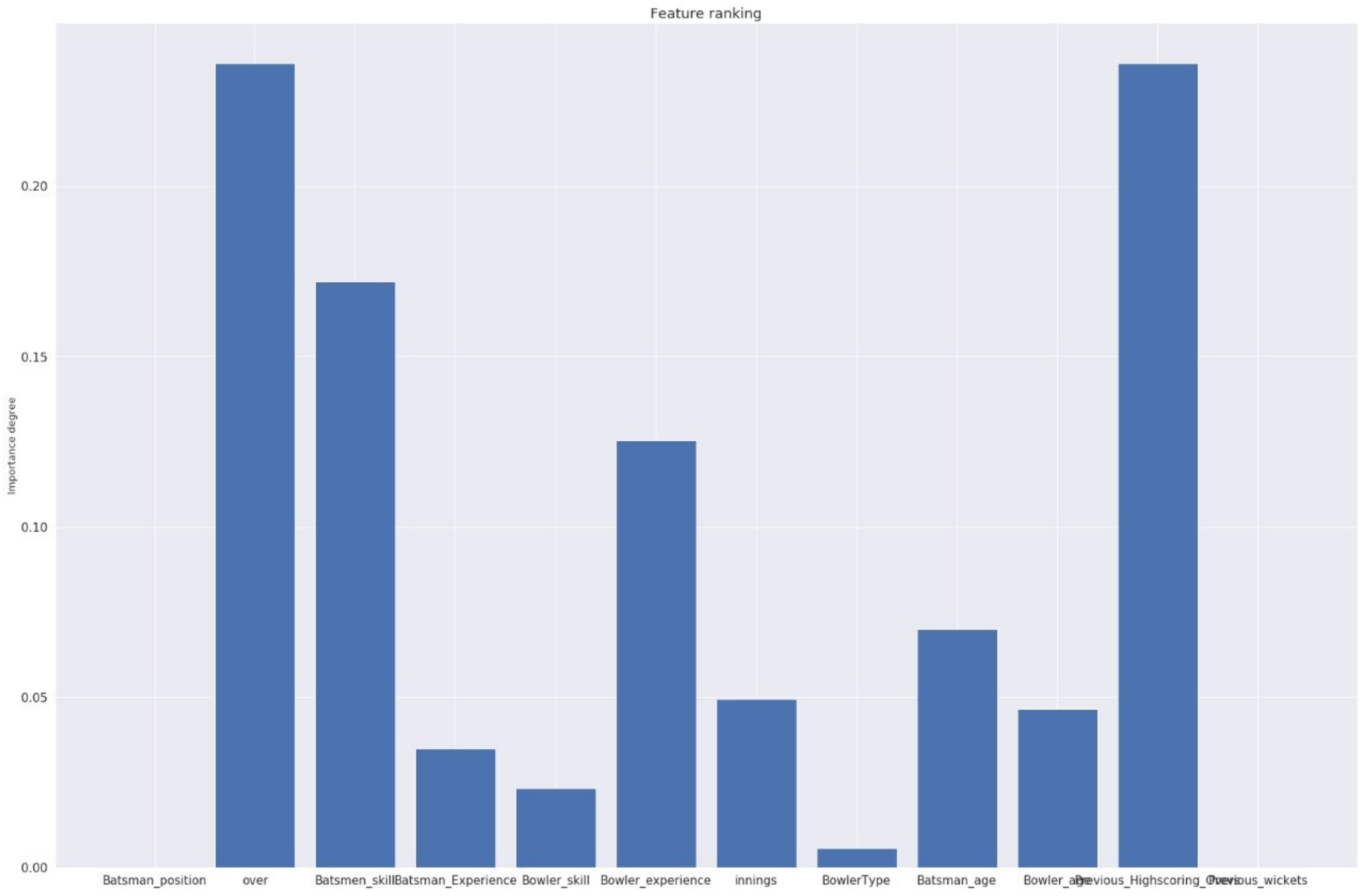
XG Boost	0.996001
Random Forest	0.996899
Logistic Regression	0.996800
SVM	0.996890
KNN	0.996446

Here we are able to predict more accurately so we tried different thresholds. If we use the threshold as 10 runs,

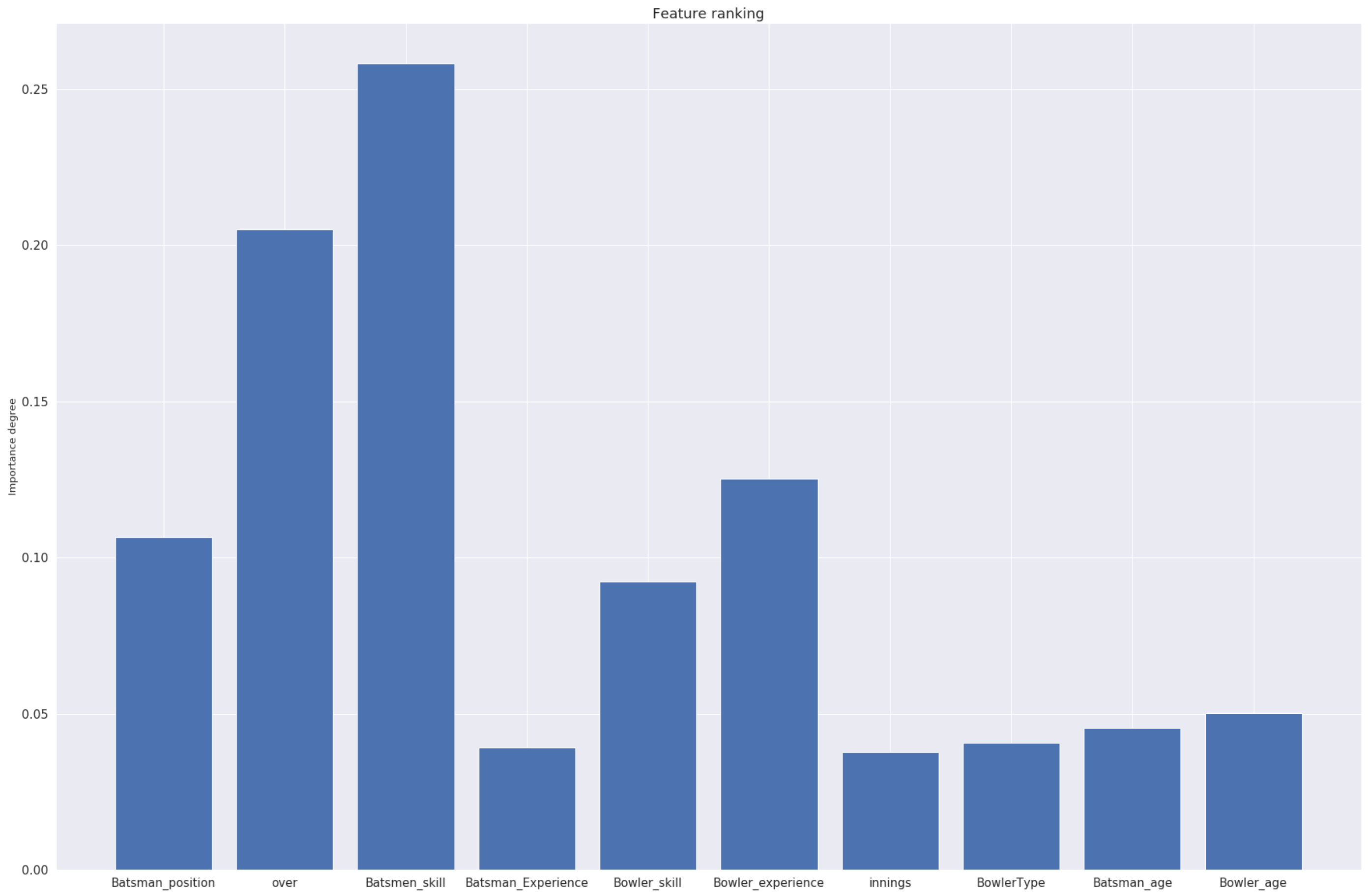
XG Boost	0.986228
Random Forest	0.986672
Logistic Regression	0.986672
SVM	0.986672
KNN	0.986672

Since XGBoost gives a better accuracy we used it to improve by using k-fold validation

Feature ranking using features which are not specific to the match and the threshold as 12 runs

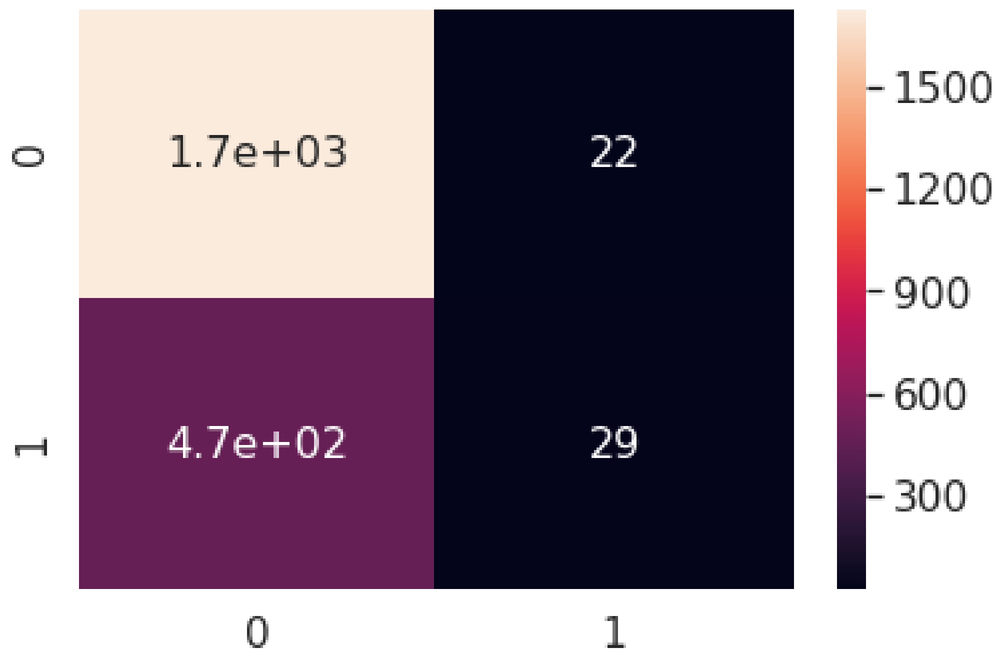


Feature ranking without using features which are not specific to the match and the threshold as 12 runs

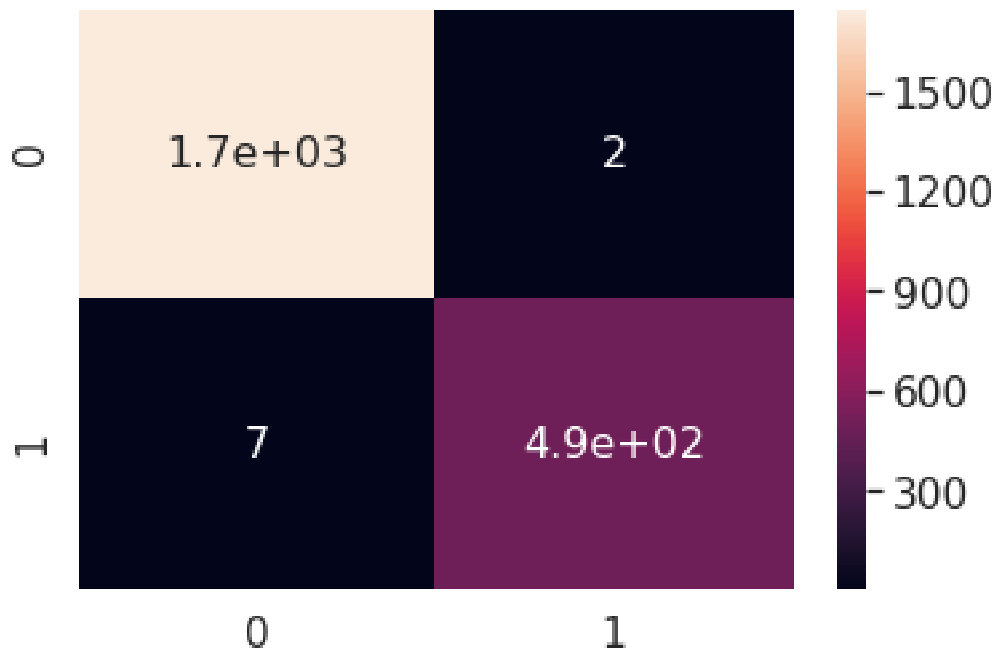


Confusion Matrix:

Using features which are not specific to the match and the threshold as 12 runs,confusion matrix is



Using features which are specific to the match along with the previous features and the threshold as 12 runs,confusion matrix is



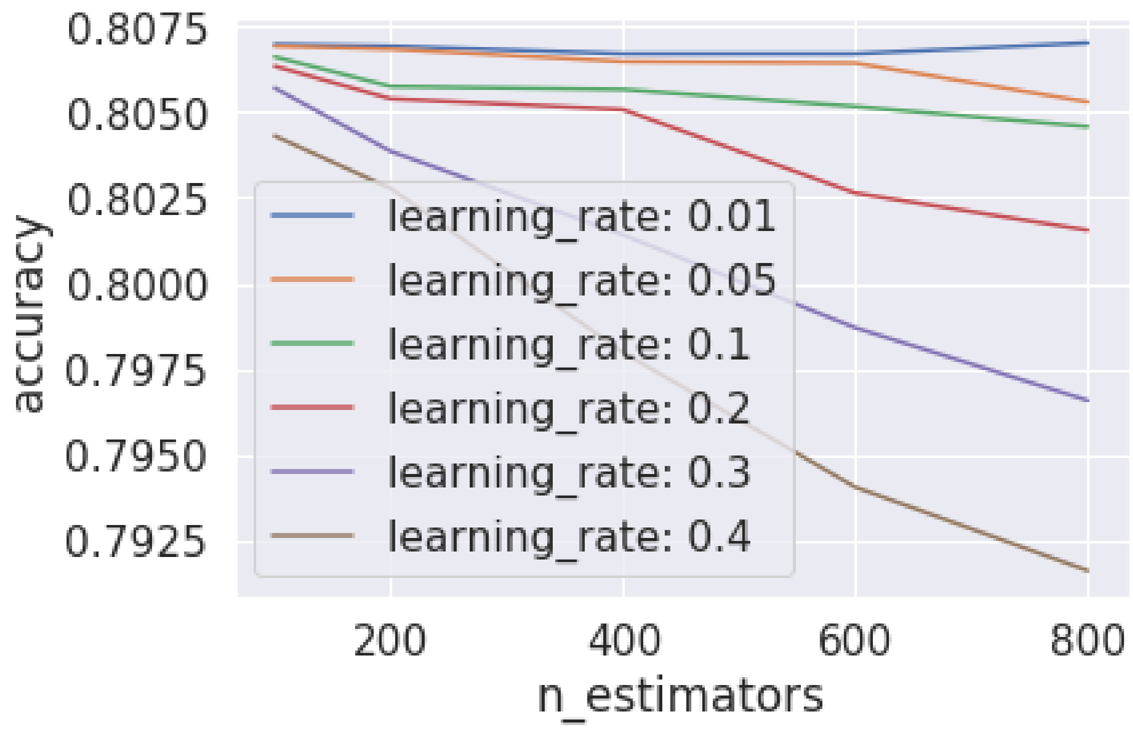
Train,validation,test metrics

Using features which are not specific to the match and the threshold as 12 runs.

Tested using number of estimators 100,200,400,600,800,learning rates 0.01,0.05,0.1,0.2,0.3,0.4

Setting model parameters for XGBoost by using 10-fold cross validation
learning rate=0.01
estimators=800

final accuracy is 0.807020



Using features which are specific to the match and the threshold as 12 runs., the accuracy is already 0.996 which is almost perfect, so we didn’t do any cross validation