Sports Analytics - Generating Actionable Insights using Cricket Commentary

BEGINNER DATA EXPLORATION DATA VISUALIZATION PYTHON SPORTS TECHNIQUE UNSTRUCTURED DATA

Overview

- What is sports analytics? What are the different use cases of sports analytics? We answer these questions here
- Understand how sports analytics can impact sports like cricket, football, and tennis
- We'll work on a fascinating sports analytics use case – analyzing India's performance using cricket commentary data!

Introduction

The scope of professional sports has changed over the years. I remember watching every minute of the 2003 Cricket World Cup and spending every waking minute tracking statistics, like the total runs scored, highest run-scorer, highest run rate, and so on.

It was fairly rudimentary stuff but enough to keep me glued to the screen. How times have changed since then!

Sports analytics is quickly becoming mainstream. Media outlets and leading sports websites regularly curate statistics, produce deep technical insights, and add a whole new level of analysis we haven't seen before.

We can now answer questions like the below ones with a high degree of confidence:

- Sports analytics in cricket: What is the probability of the team winning a match while batting first and second?
- Sports analytics in football: What is the expected outcome of a shot taken from the left side of the penalty area?
- Sports analytics in tennis: Where should you place your serve based on the return statistics of your opponent

And so on. Honestly, the sky is the limit when it comes to sports analytics use cases. I'm a sports lover and I'm always looking out for applications where I can apply my analytics and machine learning knowledge to improve the team strategy as well as fan experience.

I'll introduce you to the awe-inspiring world of sports analytics in this article. We will look at the different types of sports analytics, why this field is important, and we'll also work on a use case of sports analytics – analyzing cricket commentary to generate insights.

Table of Contents

What is Sports Analytics?

> Sports Analytics is all about analyzing and extracting useful insights from sports data.

I would broadly dive Sports Analytics into 2 categories:

1. Descriptive Sports Analytics
2. Predictive Sports Analytics

Let us discuss each category here.

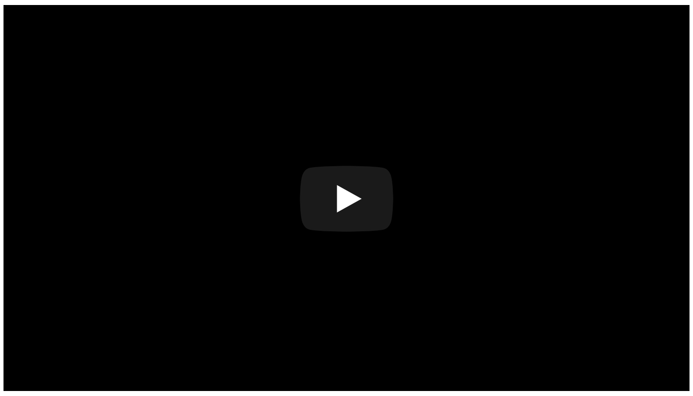
Descriptive Sports Analytics

Descriptive Sports Analytics is about **summarizing the sports data** in the form of numbers. In other words, to come up with important statistics. This might sound like a simple concept but it's a very powerful one.

The thought behind descriptive sports analytics plays a crucial role in team tactics.

Let's take cricket for example. Here, we can analyze how frequently a batsman gets out to a specific bowler. This number will decide the bowling strategy of a team.

Here is an awesome video that analyzes the dismissals of Virat Kohli against Adam Zampa:



This is the reason why **Adam Zampa was brought back into the attack whenever Virat Kohli was at the crease** during Australia's tour of India in 2020. In this series, Virat Kohli lost his wicket to Zampa in two out of three matches!

Another interesting use case in cricket is to **analyze the team's probability of winning a match while batting first as well as second.** This influences the captain who wins the toss and has to make a decision – bat or bowl first.

Predictive Sports Analytics

Predictive Sports Analytics is about **making predictions using sports data**. One such use case in cricket is to **predict the number of runs a batsman scores** against an opponent in a particular match. This would help the team management and captain select the best team for every match.

In a sport like football, predictive sports analytics helps to understand the chances of scoring a goal from any location on the pitch.

You can think of similar use cases for your favorite sport and let me know in the comments section below the article.

Importance of Sports Analytics

Sports Analytics is a game-changer – there’s no other way to put it. Using analytics in sports directly impacts the decision making of a team and can alter the future of the franchise or club (or country). It can easily change the outcome of the match.

> Sports Analytics can be a Game Changer.

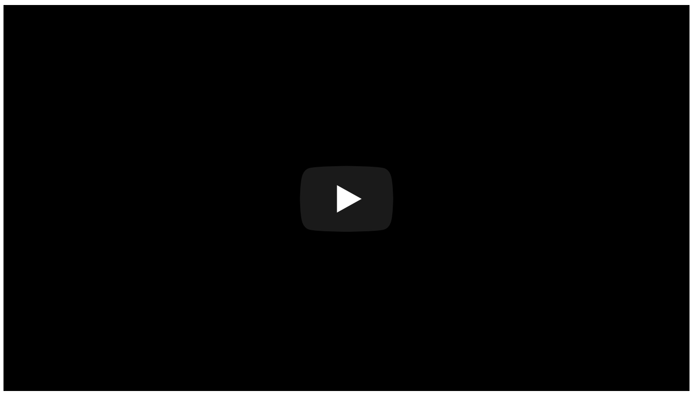
There is a lot of scope for analytics in sports. In this section, I am going to discuss a few use cases of analytics in different sports, like cricket, football, and tennis.

Sports Analytics in Cricket

In cricket, we can **analyze the strong and weak zone of a player.** This would help the opponent and player understand the strengths and weaknesses of how he plays.

- Opponents can develop a strategy to bowl against a player (like Adam Zampa against Virat Kohli)
- The player can invest more time on his weakness to improve his game

Here is an awesome video that showcases the weak zone of Virat Kohli:



Sports Analytics in Football

The footballing world has been slow to adopt analytics but it’s quickly gathering pace now. We’re seeing the mainstream media using analytics numbers, such as expected goals and expected assists to analyze players and matches.

You should definitely keep an eye out on the Expected Goals (xG) metric. xG basically tells us the probability of a shot converting into a goal. This varies from player to player and from what position the shot is being taken. It’s quite a fascinating concept and you can read more about it here.

Another example of analytics in football is analyzing team formation while the match is going on. This would help the opponent to understand the team strategy and play according to it.

Sports Analytics in Tennis

In tennis, we can identify the combination of shots a player usually plays to win a point. This can be of great use to prepare a strategy against the opponent as well.

I'm sure you must have seen the statistics that come up on screen after the end of each set at a tennis Grand Slam. Features like the number of first serves returned, the placement of the serve, the bounce of the serve and where the opponent picked it up – these are all examples of sports analytics in tennis.

Sports Analytics Case Study: Analyzing Cricket Commentary

Let's take up a real-world case study now to understand how sports analytics works. I am going to delve into my personal passion, cricket, for this case study.

I'm an avid follower of text commentary in cricket. An insightful commentator describes the events happening on the ground in good detail, right? There is a lot of online cricket commentary available on many sports websites like CricBuzz, ESPN Cricinfo, etc. This is a gold mine that can reveal many interesting and valuable insights into a team and player.

About the Dataset for Sports Analytics

I have collected the commentary of the last 4 years of the T20 matches played by India. Download a sample dataset from here. It's time to analyze the commentary and find some appealing insights. Let's do it!

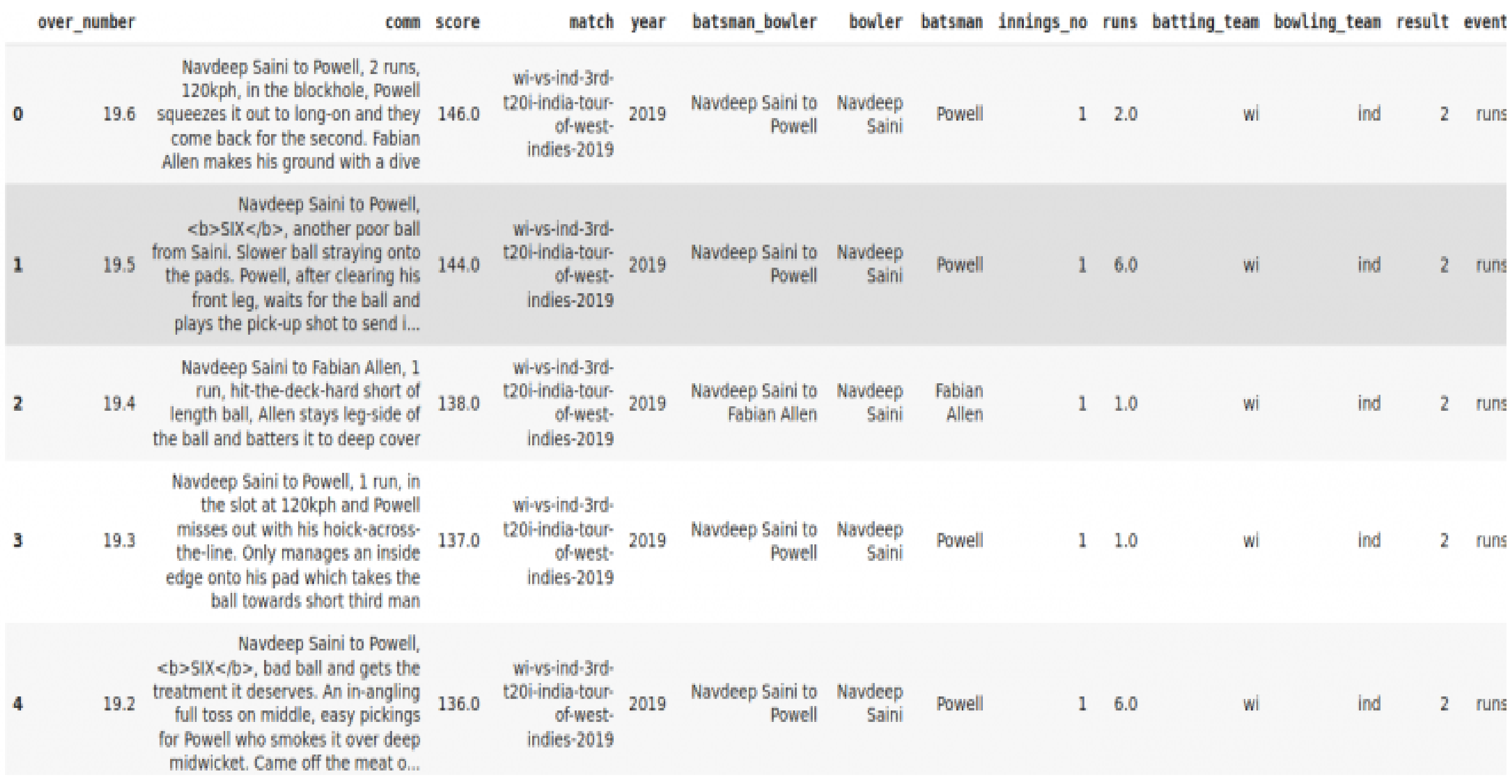
Implementation

Let us first read the dataset and understand the different columns in the dataset:

```python
df = pd.read_csv('commentary.csv')
print(df.head())
```

view raw

cc_1.py hosted with ❤ by GitHub



	over_number	comm	score	match	year	batsman_bowler	bowler	batsman	innings_no	runs	batting_team	bowling_team	result	event
0	19.6	Navdeep Saini to Powell, 2 runs, 120kph, in the blockhole, Powell squeezes it out to long-on and they come back for the second. Fabian Allen makes his ground with a dive	146.0	wi-vs-ind-3rd-t20i-india-tour-of-west-indies-2019	2019	Navdeep Saini to Powell	Navdeep Saini	Powell	1	2.0	wi	ind	2	runs
1	19.5	Navdeep Saini to Powell, <b>SIX</b>, another poor ball from Saini. Slower ball straying onto the pads. Powell, after clearing his front leg, waits for the ball and plays the pick-up shot to send i...	144.0	wi-vs-ind-3rd-t20i-india-tour-of-west-indies-2019	2019	Navdeep Saini to Powell	Navdeep Saini	Powell	1	6.0	wi	ind	2	runs
2	19.4	Navdeep Saini to Fabian Allen, 1 run, hit-the-deck-hard short of length ball, Allen stays leg-side of the ball and batters it to deep cover	138.0	wi-vs-ind-3rd-t20i-india-tour-of-west-indies-2019	2019	Navdeep Saini to Fabian Allen	Navdeep Saini	Fabian Allen	1	1.0	wi	ind	2	runs
3	19.3	Navdeep Saini to Powell, 1 run, in the slot at 120kph and Powell misses out with his hoick-across-the-line. Only manages an inside edge onto his pad which takes the ball towards short third man	137.0	wi-vs-ind-3rd-t20i-india-tour-of-west-indies-2019	2019	Navdeep Saini to Powell	Navdeep Saini	Powell	1	1.0	wi	ind	2	runs
4	19.2	Navdeep Saini to Powell, <b>SIX</b>, bad ball and gets the treatment it deserves. An in-angling full toss on middle, easy pickings for Powell who smokes it over deep midwicket. Came off the meat o...	136.0	wi-vs-ind-3rd-t20i-india-tour-of-west-indies-2019	2019	Navdeep Saini to Powell	Navdeep Saini	Powell	1	6.0	wi	ind	2	runs

Team performance

After this section, you will be able to answer the below questions:

1. What is the team average when batting first and second?
2. How frequently does the Indian team win a match?
3. What is the probability of India winning a match against a particular team?
4. What is the target to be set by the Indian team to win the match?
5. How many times has the Indian team defended a low scoring target?
6. Which was the most successful year for Team India?

Ready? Let's get our hands dirty now!

The total number of T20s India played in the last 4 years:

```python
batting_df=df[df['batting_team']=='ind']
num = len(batting_df['match'].unique())
print("Total no. of matches India played:",num)
```

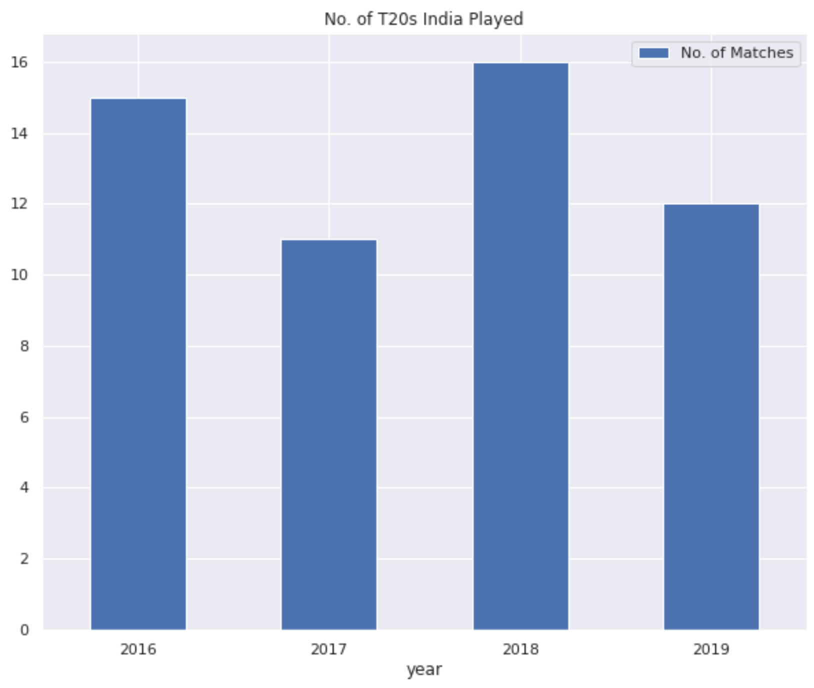
view raw

cc_2.py hosted with ❤ by GitHub

Output: `Total no. of T20s India played in last 4 years:54`

No. of T20s India played each year:

```python
num_of_matches=batting_df.groupby(['year']).apply(lambda x:x['match'].nunique()).reset_index(name='No. of Matches')
fig = num_of_matches.plot.bar(x="year", y="No. of Matches", rot=0, title="No. of T20s India Played",figsize=(10,8)).get_figure()
```



Inferences:

1. India has played the most number of T20s in 2018 and the least number of T20s in 2017 & 2019. This is because of the **ICC Champions trophy' 17 and ICC World Cup, 19** tournaments

Team Average Score (Batting First & Second):

```python
score = batting_df.groupby(['year','match','innings_no']).apply(lambda x:x['runs'].sum()).reset_index(name='score')
print("Batting First Average score:",np.median(score[score['innings_no']==1]['score'].values))
print("Batting Second Average score:",np.median(score[score['innings_no']==2]['score'].values))
```

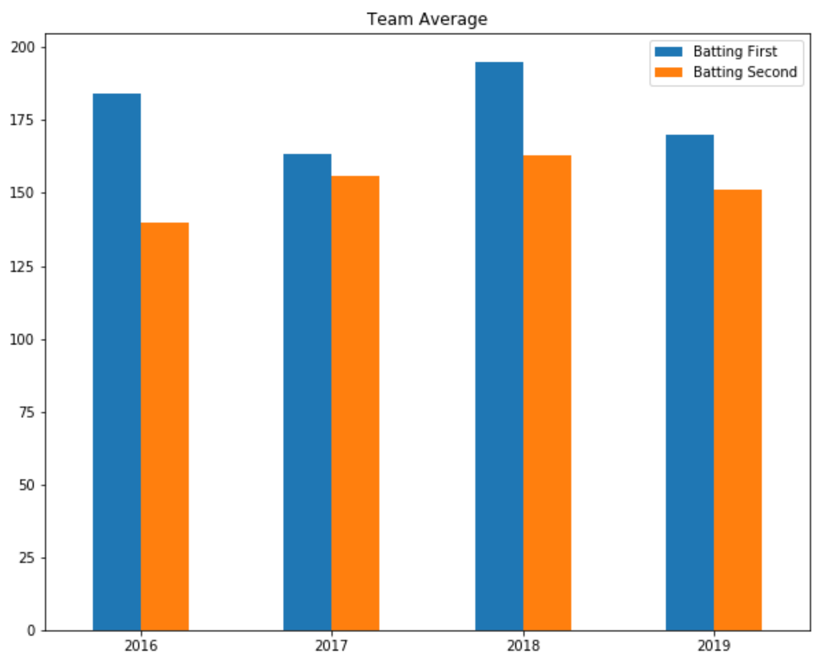
```
Output: Batting First Team Average :180.0 Batting Second Team Average:156.0
```

Team Average Innings wise over the years:

```python
#split into innings
first_innings_df=score[score['innings_no']==1]
second_innings_df=score[score['innings_no']==2]

df1=first_innings_df.groupby('year').apply(lambda x:np.median(x['score'].values)).reset_index(name='score')
df2=second_innings_df.groupby('year').apply(lambda x:np.median(x['score'].values)).reset_index(name='score')

data = {"Batting First": df1['score'].values,
"Batting Second":df2['score'].values
}
index = df1['year'].values
```

```python

# Dictionary loaded into a DataFrame
dataFrame = pd.DataFrame(data=data, index=index);

# Draw a vertical bar chart
fig= dataFrame.plot.bar(rot=0, title="Team Average",figsize=(10,8)).get_figure()
```

view raw
cc_5.py hosted with ❤ by GitHub



Inferences:

1. The highest batting first average was close to 180 in 2018. On an average, India scored 180+ runs while batting first in 2018. So, we can infer that India had the **best batting line-up** during 2018
2. Batting Second average is always less than the batting first average over all the years. From this, we can infer that the target set by the Indian team is higher than the opponents

Overall Winning % (Batting First & Second):

```python
grp = batting_df.groupby('match').first().reset_index()
matches_won = grp[grp['innings_no'] == grp['result']]
print("Over all Winning %:",(matches_won.shape[0]/num)*100)

#Batting first & second
first  = grp[grp['innings_no'] == 1]
second = grp[grp['innings_no'] == 2]

#Batting first winning %
won1 = first[first['innings_no'] == first['result']]
print("Batting First Winning % :",(won1.shape[0]/first.shape[0])*100)

#Batting secong winning %
won2 = second[second['innings_no'] == second['result']]
print("Batting Second Winning % :",(won2.shape[0]/second.shape[0])*100)
```

view raw
cc_25.py hosted with ❤ by GitHub

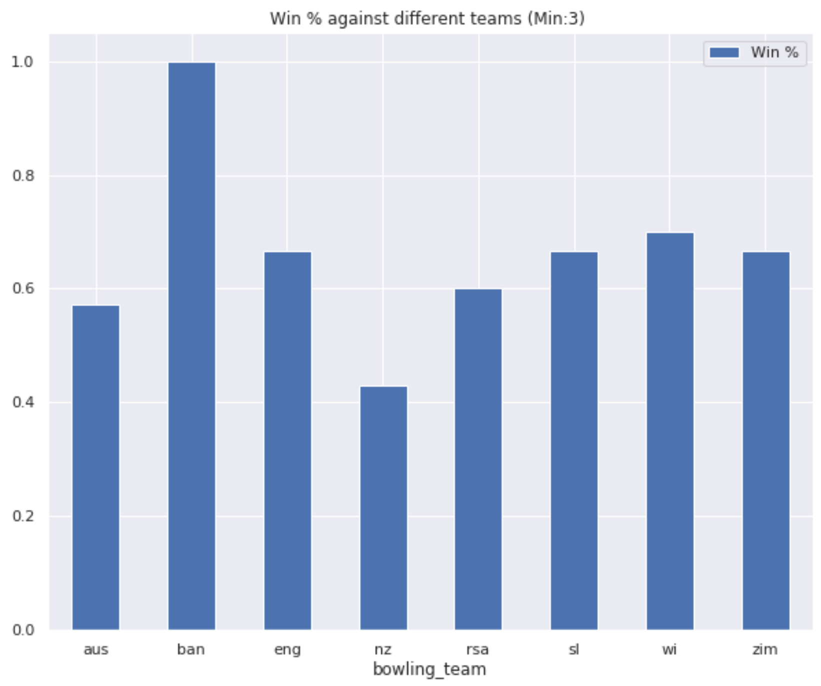
Output: `Over all Winning % : 66.66 Batting First Winning % : 59.0 Batting Second Winning %: 76.0`

Winning % against different teams:

```python
df1 = grp.groupby('bowling_team').apply(lambda x:x.shape[0]).reset_index(name='no')
df2=grp.groupby('bowling_team').apply(lambda x: x[x['innings_no']==x['result']].shape[0]).reset_index(name='won')
df2['Win %']=df2['won']/df1['no']
fig=df2.plot.bar(x="bowling_team", y="Win %", rot=0, title="Win % against different teams",figsize=(10,8)).get_figure() #weighted
```

view raw

cc_7.py hosted with ❤ by **GitHub**



Inferences:

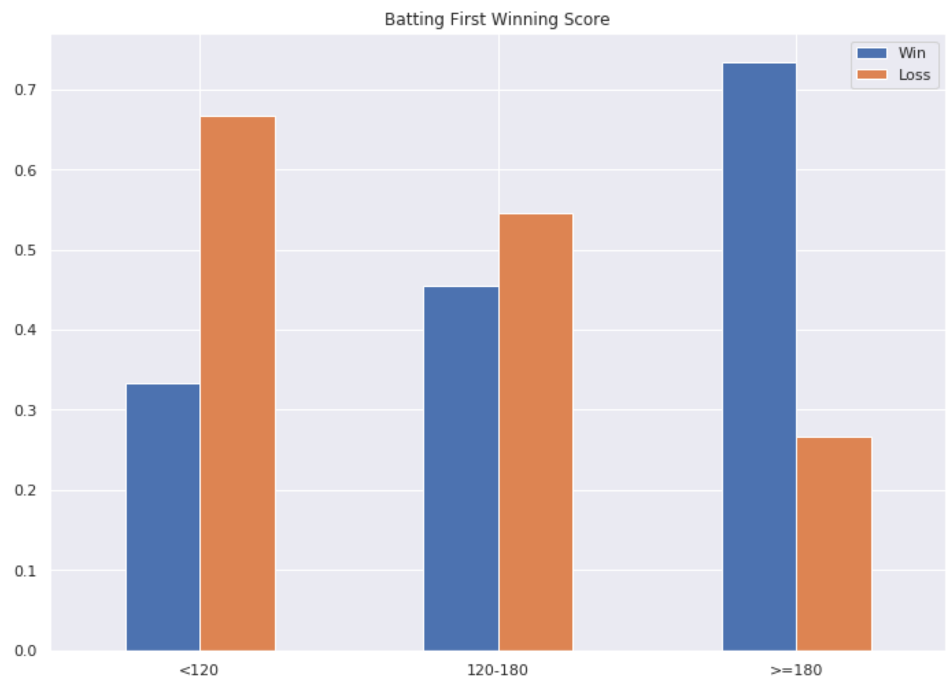
1. India wins almost every match when playing against **Bangladesh**
2. The team records the lowest winning percentage against New Zealand. India loses 60% of the matches since New Zealand are well known as the **best spin playing team**

Batting First Winning Score:

```python
block1 = first[first['score']>=180]
won1 = block1[block1['innings_no']==block1['result']].shape[0]
lost1=block1[block1['innings_no']!=block1['result']].shape[0]

block2 = first[(first['score']>=120) & (first['score']<180) ]
won2 = block2[block2['innings_no']==block2['result']].shape[0]
lost2=block2[block2['innings_no']!=block2['result']].shape[0]

block3 = first[first['score']<120]
won3 = block3[block3['innings_no']==block3['result']].shape[0]
lost3=block3[block3['innings_no']!=block3['result']].shape[0]
```

view raw

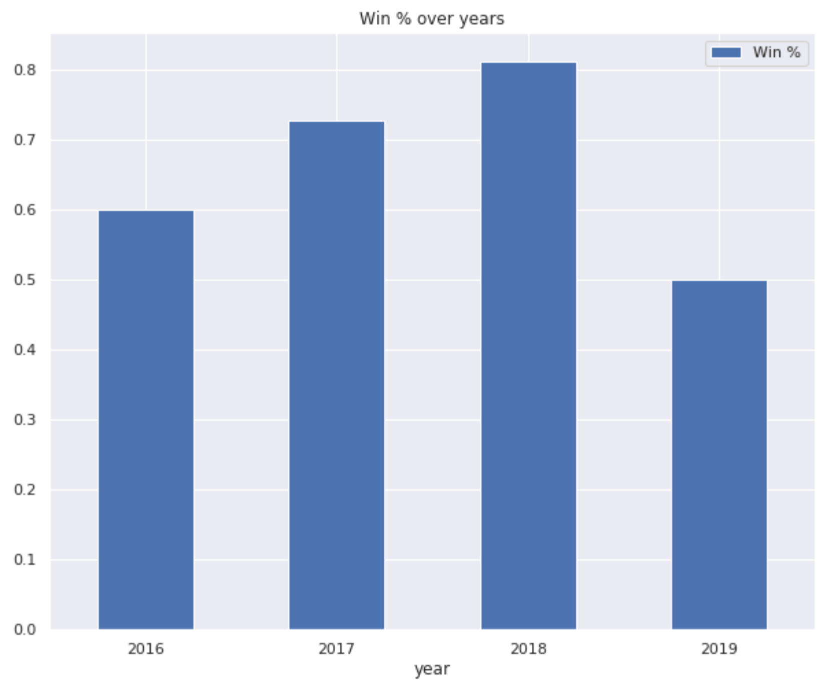
cc_8.py hosted with ❤ by **GitHub**



Inferences:

1. Probability of India winning a match after scoring:
    1. Less than 120 runs (<120) is around 0.33
    2. Between 120 and 180 runs (120-180) is around 0.4
    3. Greater than 180 runs (>180) is 0.75
2. Glad to see that India has **defended low scoring games** too

Winning % over the years:



Inferences:

1. Team performance has increased over the years and then drops down in 2019
2. India records the highest winning percentage (82%) in 2018. The team won most of the matches played during 2018
3. India lost half of the T20s played in 2019. Hence, the lowest winning percentage (50%) in 2019. One of the possible reasons could be due to the lack of senior players as the team opened up the doors for youngsters after ICC World Cup 2019

Batting performance

In this section, I will focus on the batting performance of team India in terms of the strike rate. We'll also discuss how India's performance has evolved over a period of time.

Strike rate can be defined as the average number of runs scored per 100 balls. The higher the strike rate, the better the batsman is.

Let's find out the phases where team India can improve its batting.

Overall batting strike rate of Indian team:

```
batting_df=df[df['batting_team']=='ind']
bowling_df=df[df['bowling_team']=='ind']

#strike rate of indian team
sr=(np.sum(batting_df['runs'].values)/batting_df.shape[0])*100
print("Strike rate of Indian team:",sr)
```

view raw

cc_9.py hosted with ❤ by GitHub

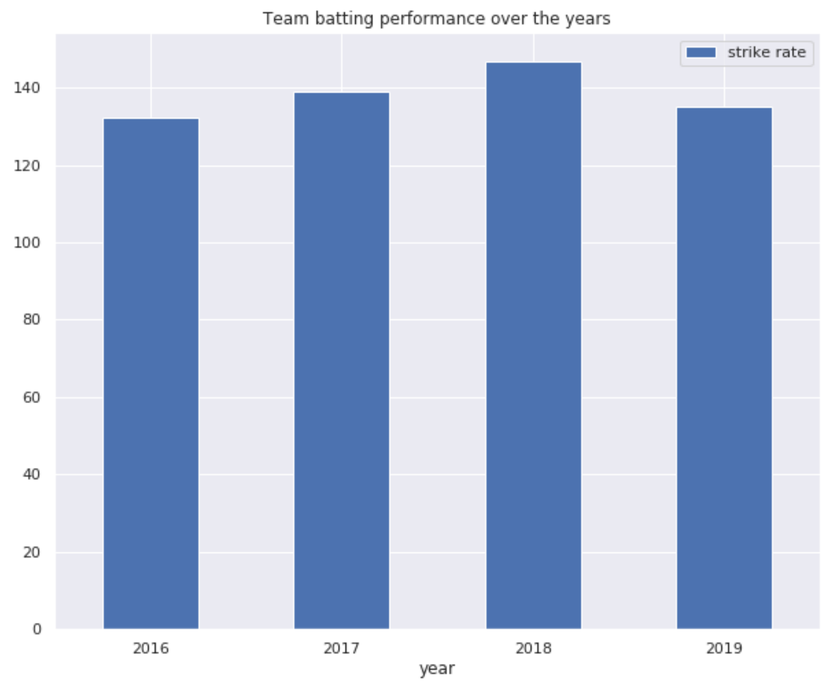
Output: Strike rate of Indian team is 138.66

Team batting strike rate over the years:

```
sr=batting_df.groupby('year').apply( lambda x: (np.sum(x['runs'].values)/x.shape[0])*100  ).reset_index(name='strike rate')
fig=sr.plot.bar(x="year", y="strike rate", rot=0, title="Team Batting performance over the years",figsize=(10,8)).get_figure()
```

view raw

cc_10.py hosted with ❤ by GitHub



Inference:

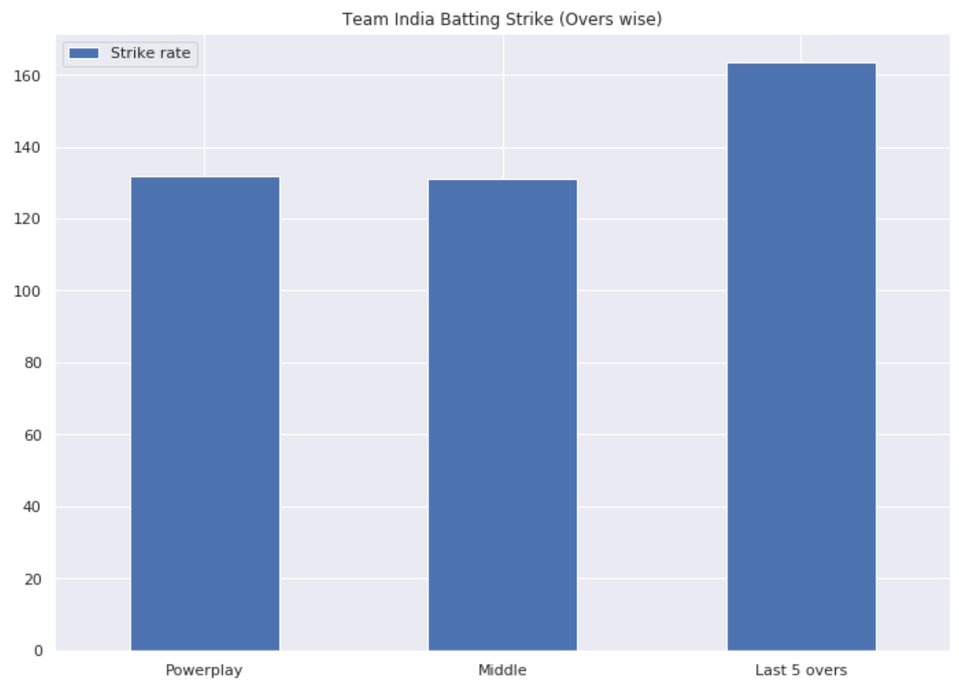
1. India had the highest batting striking rate in 2018. The batsmen were in great touch!

Team batting strike rate across different phases of a match:

```python
powerplay_df = df[df['over_number']<=5.6]
middle_df = df[(df['over_number']>=6.1) & (df['over_number']<=14.6)]
last_df = df[(df['over_number']>=15.1) & (df['over_number']<=19.6)]

powerplay_batting_df = powerplay_df[powerplay_df['batting_team']=='ind']
middle_batting_df = middle_df[middle_df['batting_team']=='ind']
last_batting_df = last_df[last_df['batting_team']=='ind']

sr1=(np.sum(powerplay_batting_df['runs'].values)/powerplay_batting_df.shape[0])*100
sr2=(np.sum(middle_batting_df['runs'].values)/middle_batting_df.shape[0])*100
sr3=(np.sum(last_batting_df['runs'].values)/last_batting_df.shape[0])*100

#plot
data = {"Strike rate":[sr1,sr2,sr3]
        };

index  = ["Powerplay", "Middle", "Last 5 overs"];

# Dictionary loaded into a DataFrame
dataFrame = pd.DataFrame(data=data, index=index);

# Draw a vertical bar chart
axes = dataFrame.plot.bar(rot=0, title="Team India Batting Strike (Overs wise)").get_figure()
```

view raw

cc_11.py hosted with ❤ by **GitHub**



Inferences:

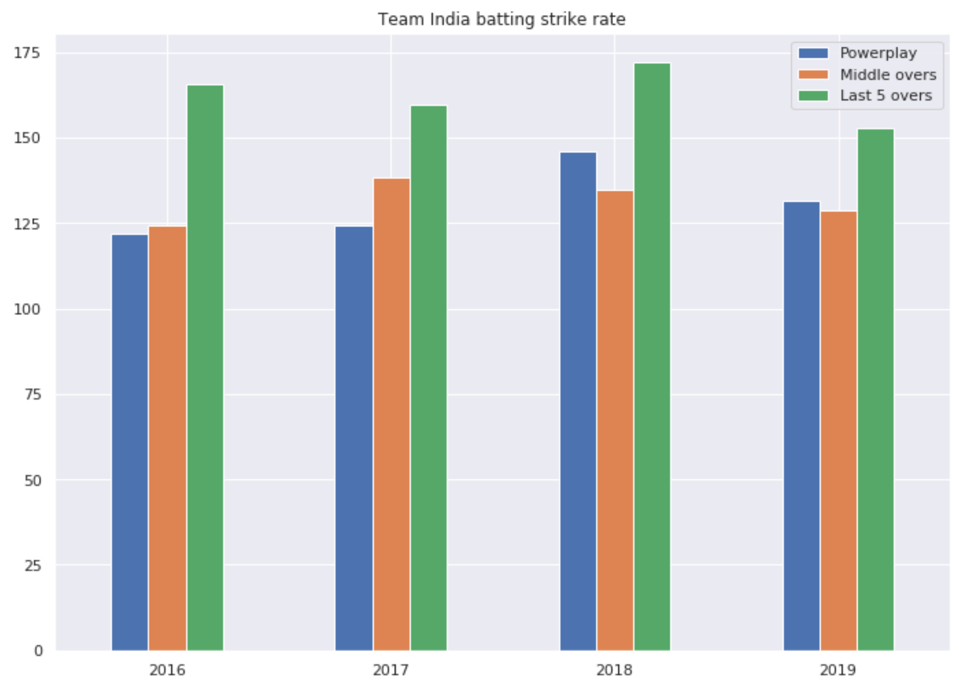
1. The strike rate of the Indian team reaches **around 150+** in the last 5 overs. And around **125+** in power play and middle overs

Team batting strike rate across different phases of a match over the years:

```python
#Strike rate
score = powerplay_batting_df.groupby(['year','match']).apply(lambda x:x['runs'].sum()).reset_index(name='score')
pp=powerplay_batting_df.groupby("year").apply(lambda x:(x['runs'].sum()/x.shape[0])*100).reset_index(name='strike rate')
middle=middle_batting_df.groupby("year").apply(lambda x:(x['runs'].sum()/x.shape[0])*100).reset_index(name='strike rate')
last=last_batting_df.groupby("year").apply(lambda x:(x['runs'].sum()/x.shape[0])*100).reset_index(name='strike rate')

data = {"Powerplay":pp['strike rate'].values,
        "Middle overs":middle['strike rate'].values,
        "Last 5 overs":last['strike rate'].values
        };

index  = last['year'].values
# Dictionary loaded into a DataFrame
dataFrame = pd.DataFrame(data=data, index=index);

# Draw a vertical bar chart
axes = dataFrame.plot.bar(rot=0, title="Team India batting strike rate")
#axes[1].legend(loc=2)

fig=axes.get_figure()
```

view raw

cc_12.py hosted with ❤ by **GitHub**



Inferences:

1. In 2018, India recorded the highest batting strike rate across all the 3 phases (Powerplay, middle overs, and the last 5 overs)
2. The highest batting strike rate, close to 175, was recorded in 2018 during the last 5 overs. Reminds me of **Dhoni & Hardik Pandya's powerful hitting**

Bowling performance

In this section, let's unleash the bowling performance of team India in terms of **Economy rate, Bowling Strike rate, and Bowling Average**. And also how the performance has evolved over time.

- **Economy rate** is defined as the average number of runs conceded per over
- **Bowling strike rate** can be defined as the average number of balls conceded for a wicket
- **Bowling Average** is the average number of runs conceded for a wicket

Its time to analyze the bowling performance of the Indian team.

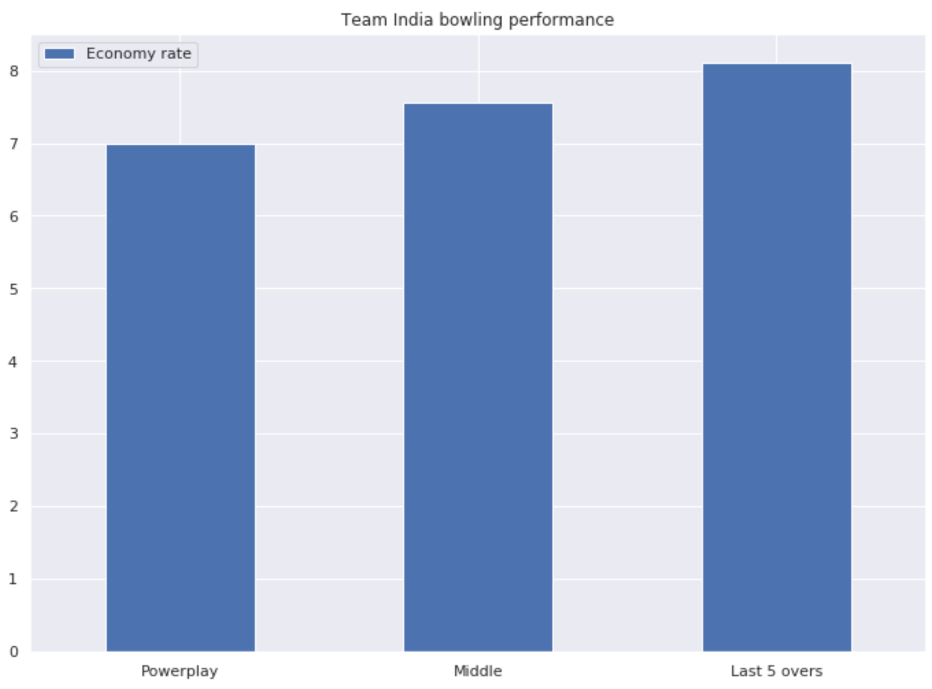
Team India Economy rate in different phases of a match:

```
powerplay_bowling_df = powerplay_df[powerplay_df['bowling_team']=='ind']
middle_bowling_df = middle_df[middle_df['bowling_team']=='ind']
last_bowling_df = last_df[last_df['bowling_team']=='ind']

temp = powerplay_bowling_df.groupby('match').apply(lambda x:x['runs'].sum()).reset_index()
avg1=np.median(temp[0].values)
er1 = avg1/6.0

temp = middle_bowling_df.groupby('match').apply(lambda x:x['runs'].sum()).reset_index()
avg2=np.median(temp[0].values)
er2 = avg2/9.0

temp = last_bowling_df.groupby('match').apply(lambda x:x['runs'].sum()).reset_index()
avg3=np.median(temp[0].values)
```

```python
er3 = avg3/5.0

data = {"Economy rate":[er1,er2,er3] };
index = ["Powerplay", "Middle", "Last 5 overs"];

# Dictionary loaded into a DataFrame
dataFrame = pd.DataFrame(data=data, index=index);

# Draw a vertical bar chart
fig = dataFrame.plot.bar(rot=0, title="Team India bowling performance").get_figure()
```

view raw

cc_14.py hosted with ❤ by GitHub



Inferences:

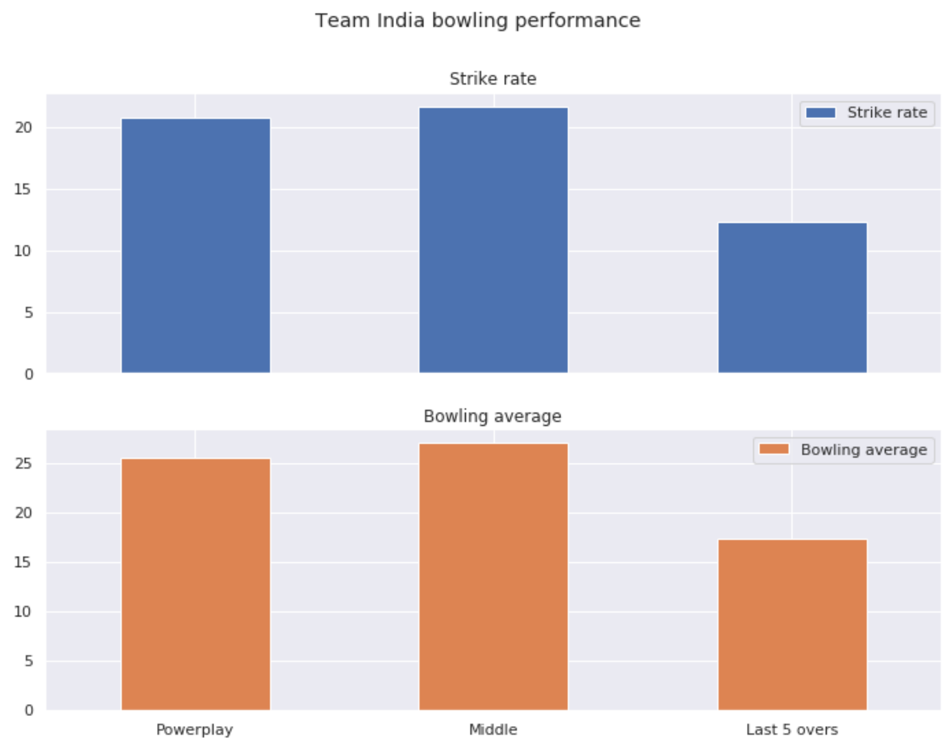
1. Indian bowlers concede around **7-8** runs per over. All credit to the Indian bowlers for such a healthy number!

Team India bowling performance across different phases of a match:

```python
#bowling strike rate
sr1 = powerplay_bowling_df.shape[0]/(powerplay_bowling_df[powerplay_bowling_df['event']=='out'].shape[0])
sr2 = middle_bowling_df.shape[0]/(middle_bowling_df[middle_bowling_df['event']=='out'].shape[0])
sr3 = last_bowling_df.shape[0]/(last_bowling_df[last_bowling_df['event']=='out'].shape[0])

#bowling average
avg1=np.sum(powerplay_bowling_df['runs'].values)/(powerplay_bowling_df[powerplay_bowling_df['event']=='out'].shape[0])
avg2=np.sum(middle_bowling_df['runs'].values)/(middle_bowling_df[middle_bowling_df['event']=='out'].shape[0])
avg3=np.sum(last_bowling_df['runs'].values)/(last_bowling_df[last_bowling_df['event']=='out'].shape[0])

# A python dictionary
data = {"Strike rate":[sr1,sr2,sr3],
"Bowling average":[avg1,avg2,avg3]
}
index = ["Powerplay", "Middle", "Last 5 overs"];

# Dictionary loaded into a DataFrame
dataFrame = pd.DataFrame(data=data, index=index);

```

```
# Draw a vertical bar chart
fig = dataFrame.plot.bar(rot=0, title="Team India bowling performance").get_figure()
```

view raw
cc_15.py hosted with ❤ by GitHub



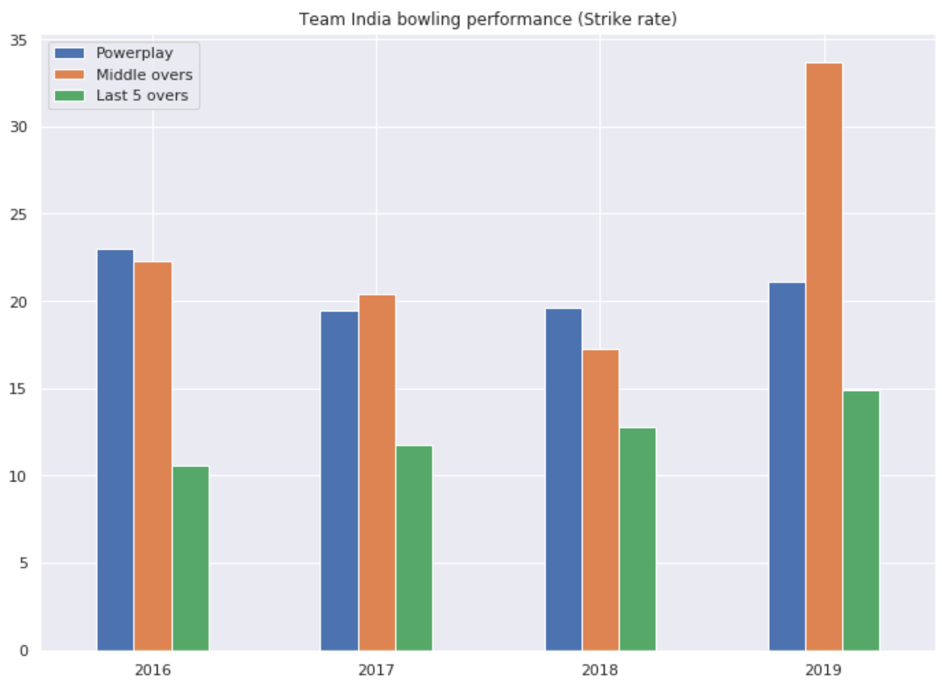
Inferences:

1. On an average, Indian bowlers pick **2 wickets in the last 5 overs** as bowlers concede around 13 balls for each wicket
2. Indian bowlers concede around **27+ runs** for a wicket in the middle overs which can be improved

Team bowling strike rate across different phases of a match over the years:

```
pp=powerplay_bowling_df.groupby('year').apply(lambda x: x.shape[0] / x[x['event']=='out'].shape[0] ).reset_index(name='strike ra
middle=middle_bowling_df.groupby('year').apply(lambda x: x.shape[0] / x[x['event']=='out'].shape[0] ).reset_index(name='strike ra
last=last_bowling_df.groupby('year').apply(lambda x: x.shape[0] / x[x['event']=='out'].shape[0] ).reset_index(name='strike rate'

# A python dictionary
data = {"Powerplay":pp['strike rate'].values,
"Middle overs":middle['strike rate'].values,
"Last 5 overs":last['strike rate'].values
}
index = last['year'].values
# Dictionary loaded into a DataFrame
dataFrame = pd.DataFrame(data=data, index=index);

# Draw a vertical bar chart
fig = dataFrame.plot.bar(rot=0, title="Team India bowling performance (Strike rate)").get_figure()
```

view raw
cc_16.py hosted with ❤ by GitHub



Inferences:

1. **Team India's bowling performance was very poor in 2019** as the bowlers considered 33+ balls on an average to take a wicket in the middle overs
2. India had the best death bowling attack in 2016

Boundary Analysis

In this section, we will be analyzing the average number of balls conceded by team India to score a boundary and also its evolution over the years.

```
print("Avg no. of balls to hit 4:",batting_df.shape[0]/batting_df['runs'].value_counts()[4])
print("Avg no. of balls to hit 6:",batting_df.shape[0]/batting_df['runs'].value_counts()[6])
```

view raw
cc_24.py hosted with ❤ by GitHub

Output: `Avg no. of balls to hit 4: 9 Avg no. of balls to hit 6: 19`

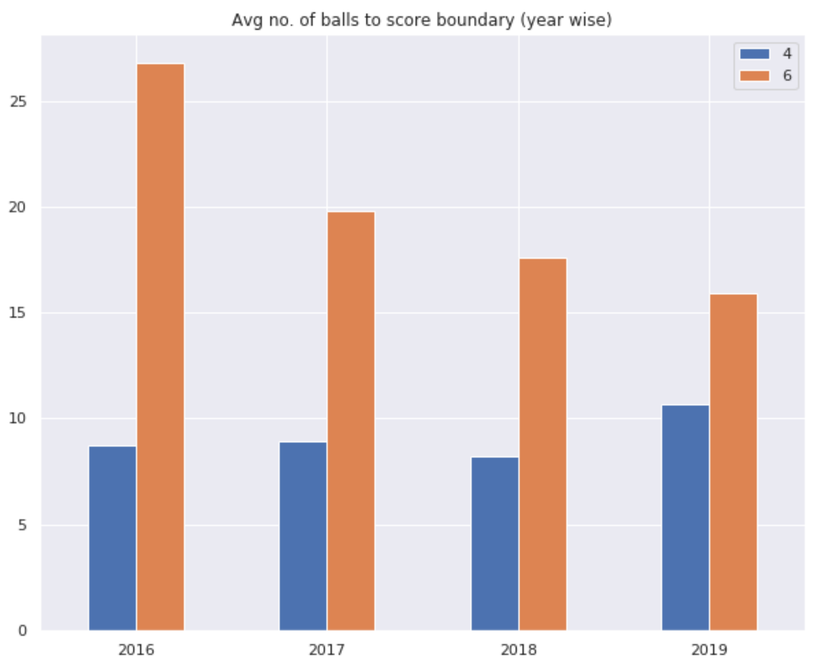
Avg number of balls to score boundary over the years:

```
df_4 = batting_df.groupby('year').apply(lambda x: x.shape[0] / x[x['runs']==4].shape[0] ).reset_index(name='num_of_balls')
df_6 = batting_df.groupby('year').apply(lambda x: x.shape[0] / x[x['runs']==6].shape[0] ).reset_index(name='num_of_balls')

# A python dictionary
data = {"4":df_4['num_of_balls'].values,
"6":df_6['num_of_balls'].values
}

index = df_4['year'].values
# Dictionary loaded into a DataFrame
dataFrame = pd.DataFrame(data=data, index=index);

# Draw a vertical bar chart
```

```
fig = dataFrame.plot.bar(rot=0, title="Avg no. of balls to score boundary (year wise)",figsize=(10,8)).get_figure()
```

view raw

cc_17.py hosted with ❤ by GitHub



Inferences:

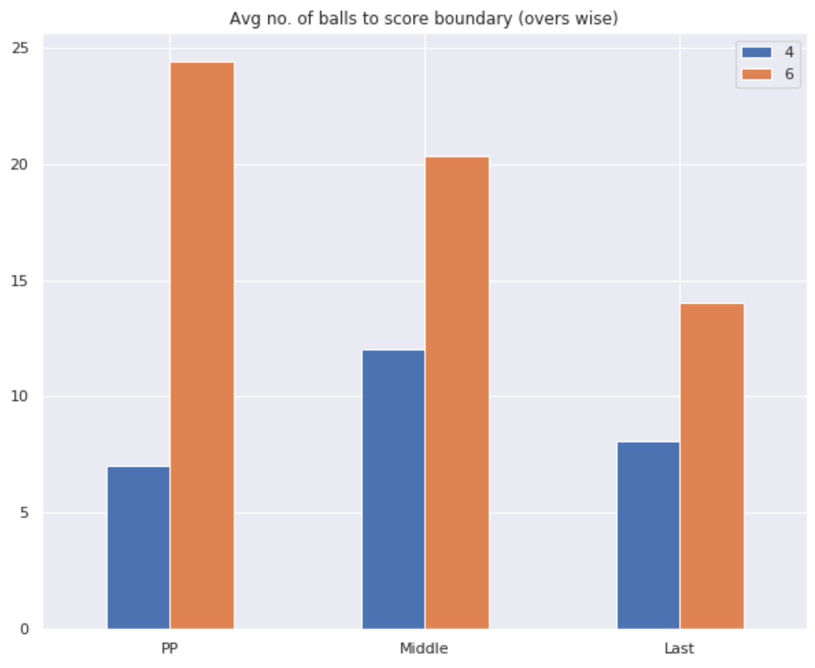
1. Team India **improved power hitting** over the past few years. In 2019, the team cleared six for every 16 balls

Avg number of balls to score boundary across different phases of the match:

```
num1=powerplay_batting_df.shape[0]/powerplay_batting_df['runs'].value_counts()[4]
num2=middle_batting_df.shape[0]/middle_batting_df['runs'].value_counts()[4]
num3=last_batting_df.shape[0]/last_batting_df['runs'].value_counts()[4]

x_axis = ['PP','Middle','Last']
y_axis = [num1, num2, num3]
dataframe= pd.DataFrame({'overs':x_axis,'4':y_axis})

num1=powerplay_batting_df.shape[0]/powerplay_batting_df['runs'].value_counts()[6]
num2=middle_batting_df.shape[0]/middle_batting_df['runs'].value_counts()[6]
num3=last_batting_df.shape[0]/last_batting_df['runs'].value_counts()[6]
dataframe['6'] = [num1,num2,num3]

data = {"4":dataframe['4'].values,
"6":dataframe['6'].values
};

index = dataframe['overs'].values
# Dictionary loaded into a DataFrame
dataFrame = pd.DataFrame(data=data, index=index);

# Draw a vertical bar chart
fig = dataFrame.plot.bar(rot=0, title="Avg no. of balls to score boundary(overs wise)",figsize=(10,8)).get_figure()
```

view raw

cc_18.py hosted with ❤ by GitHub



Inferences:

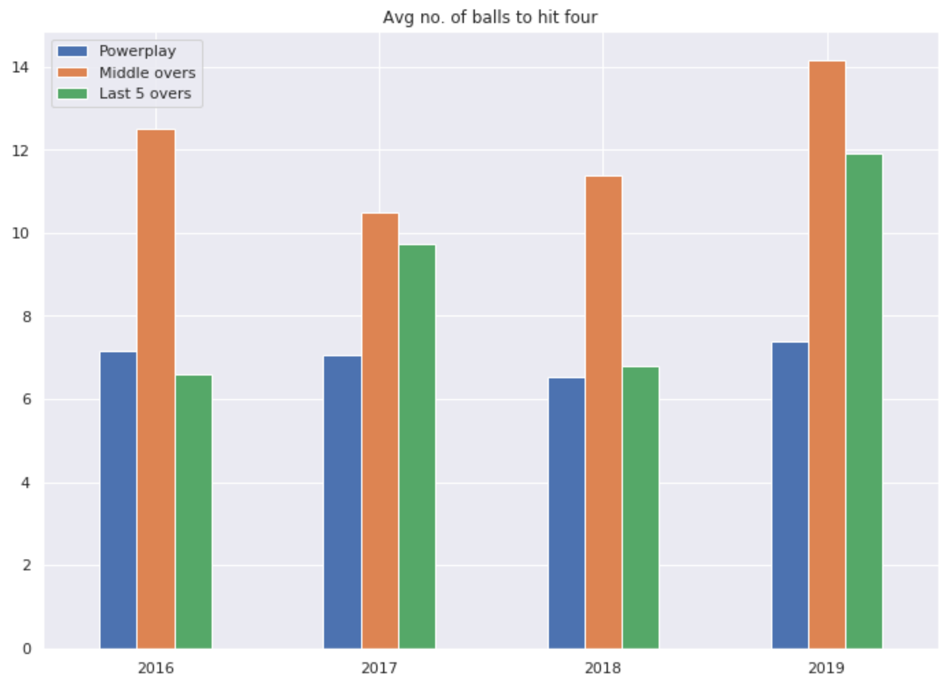
1. India clears 4 for every over in the power play and the last 5 overs
2. Team India smashes only 1 six in power play as the batsmen concede 24+ balls for a six

Avg number of balls to score 4 across different phases over the years:

```python
pp=powerplay_batting_df.groupby('year').apply(lambda x: x.shape[0] / x[x['runs']==4.shape[0] ).reset_index(name='no_of_balls')
middle=middle_batting_df.groupby('year').apply(lambda x: x.shape[0] / x[x['runs']==4].shape[0] ).reset_index(name='no_of_balls')
last=last_batting_df.groupby('year').apply(lambda x: x.shape[0] / x[x['runs']==4].shape[0] ).reset_index(name='no_of_balls')

# A python dictionary
data = {"Powerplay":pp['no_of_balls'].values,
        "Middle overs":middle['no_of_balls'].values,
        "Last 5 overs":last['no_of_balls'].values
        };

index  = last['year'].values
# Dictionary loaded into a DataFrame
dataFrame = pd.DataFrame(data=data, index=index);

# Draw a vertical bar chart
axes = dataFrame.plot.bar(rot=0, title="Avg no. of balls to hit 4")

fig=axes.get_figure()
```

view raw

cc_19.py hosted with ❤ by GitHub



Inferences:

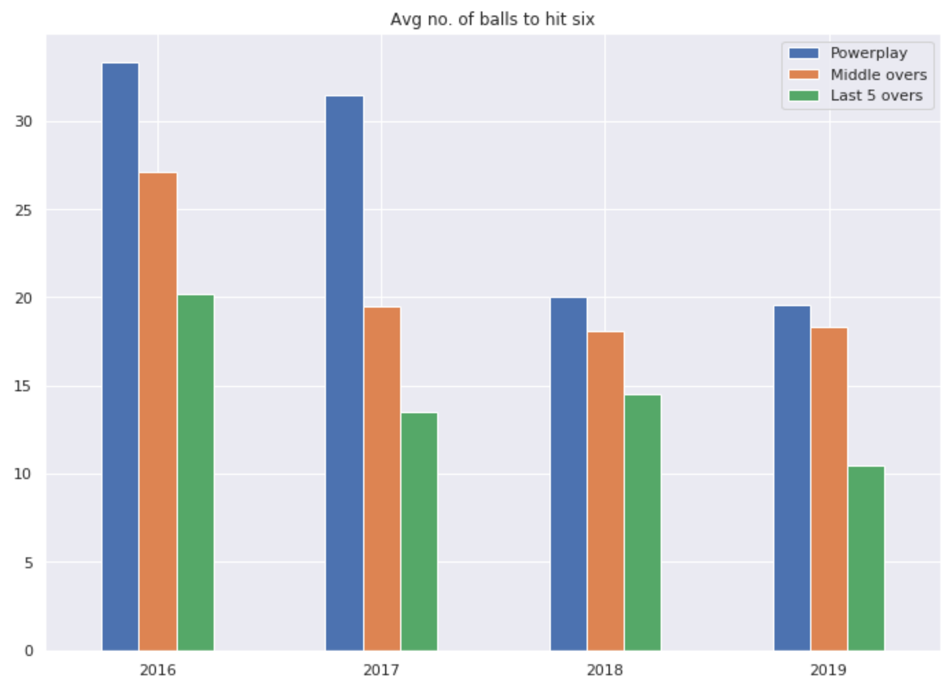
1. In 2019, India conceded the most number of balls (around 14) to hit 4 during the middle overs

Avg number of balls to score 6 across different phases over the years:

```python
pp=powerplay_batting_df.groupby('year').apply(lambda x: x.shape[0] / x[x['runs']==6.shape[0] ).reset_index(name='no_of_balls')
middle=middle_batting_df.groupby('year').apply(lambda x: x.shape[0] / x[x['runs']==6].shape[0] ).reset_index(name='no_of_balls')
last=last_batting_df.groupby('year').apply(lambda x: x.shape[0] / x[x['runs']==6].shape[0] ).reset_index(name='no_of_balls')

# A python dictionary
data = {"Powerplay":pp['no_of_balls'].values,
        "Middle overs":middle['no_of_balls'].values,
        "Last 5 overs":last['no_of_balls'].values
        };

index  = last['year'].values
# Dictionary loaded into a DataFrame
dataFrame = pd.DataFrame(data=data, index=index);

# Draw a vertical bar chart
axes = dataFrame.plot.bar(rot=0, title="Avg no. of balls to hit 6")

fig=axes.get_figure()
```

view raw

cc_20.py hosted with ❤ by GitHub



Inferences:

1. Indian openers, middle order, and finishers improved the ability to clear six over the past years. That's amazing!

End Notes

Unquestionably, Descriptive Sports Analytics has a far-reaching role to play in a team winning strategy compared to Predictive Sports Analytics. In this article, you have learned the importance of Sports Analytics and how analytics can impact different sports. We also analyzed team India's performance over the past 4 years in T20 cricket.

Kindly leave your queries/feedback in the comments sections, I will reach out to you. Have fun implementing these ideas for your favorite sport!

Article Url - https://www.analyticsvidhya.com/blog/2020/02/sports-analytics-generating-actionable-insights-using-cricket-commentary/

[Aravind Pai]()

Aravind is a sports fanatic. His passion lies in developing data-driven products for the sports domain. He strongly believes that analytics in sports can be a game-changer