IPL Cricket Match Winning Prediction

- By Soumya Ranjan Nayak

Overview:

This project uses the IPL dataset of (2008-2017) from Kaggle to predict the winning and losing percentages of both team playing in a particular match.

The project uses different python libraries and Machine learning algorithms to analyse,visualize and predict the match result.

Objective:

The main objective of this project is to predict the winning and losing percentage of any two teams playing a match at a particular situation and place based on the past data.

The second objective deals with, the diplaying the prediction analytics in a website.

Importing required libraries

In [38]:
```
import numpy as np
import pandas as pd
pd.options.mode.chained_assignment = None  # Suppress SettingWithCopyWarning
import matplotlib.pyplot as plt

import warnings
warnings.simplefilter(action='ignore', category=FutureWarning) # for ignoring the F
```

Reading the IPL ***match*** dataset

In [39]:
```
mdf=pd.read_csv("matches.csv")
mdf
```

Out[39]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	c
0	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	
1	2	2017	Pune	2017-04-06	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	
2	3	2017	Rajkot	2017-04-07	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	
3	4	2017	Indore	2017-04-08	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	
4	5	2017	Bangalore	2017-04-08	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	normal	
...	...	...	...	...	...	...	...	...	...	
631	632	2016	Raipur	2016-05-22	Delhi Daredevils	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	
632	633	2016	Bangalore	2016-05-24	Gujarat Lions	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	
633	634	2016	Delhi	2016-05-25	Sunrisers Hyderabad	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	
634	635	2016	Delhi	2016-05-27	Gujarat Lions	Sunrisers Hyderabad	Sunrisers Hyderabad	field	normal	
635	636	2016	Bangalore	2016-05-29	Sunrisers Hyderabad	Royal Challengers Bangalore	Sunrisers Hyderabad	bat	normal	

636 rows × 18 columns

Exploratory Data Analysis with ***match*** dataset:

In [40]:
```
#getting no.of rows and columns of match DataFrame
mdf.shape
```

Out[40]:
```
(636, 18)
```

In [41]:
```
#getting no.of cells in the DataFrame
mdf.size
```

Out[41]:
```
11448
```

In [42]:
```
#concise summary of the DataFrame
mdf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 636 entries, 0 to 635
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   id               636 non-null    int64  
 1   season           636 non-null    int64  
 2   city             629 non-null    object 
 3   date             636 non-null    object 
 4   team1            636 non-null    object 
 5   team2            636 non-null    object 
 6   toss_winner      636 non-null    object 
 7   toss_decision    636 non-null    object 
 8   result           636 non-null    object 
 9   dl_applied       636 non-null    int64  
 10  winner           633 non-null    object 
 11  win_by_runs      636 non-null    int64  
 12  win_by_wickets   636 non-null    int64  
 13  player_of_match  633 non-null    object 
 14  venue            636 non-null    object 
 15  umpire1          635 non-null    object 
 16  umpire2          635 non-null    object 
 17  umpire3          0 non-null      float64
dtypes: float64(1), int64(5), object(12)
memory usage: 89.6+ KB
```

In [43]:
```
#getting the summary statistics for numerical data
mdf.describe()
```

Out[43]:

	id	season	dl_applied	win_by_runs	win_by_wickets	umpire3
count	636.000000	636.000000	636.000000	636.000000	636.000000	0.0
mean	318.500000	2012.490566	0.025157	13.682390	3.372642	NaN
std	183.741666	2.773026	0.156726	23.908877	3.420338	NaN
min	1.000000	2008.000000	0.000000	0.000000	0.000000	NaN
25%	159.750000	2010.000000	0.000000	0.000000	0.000000	NaN
50%	318.500000	2012.000000	0.000000	0.000000	4.000000	NaN
75%	477.250000	2015.000000	0.000000	20.000000	7.000000	NaN
max	636.000000	2017.000000	1.000000	146.000000	10.000000	NaN

In [44]:
```
#top 5 records of match Dataframe
mdf.head(5)
```

Out[44]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_ap
0	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	
1	2	2017	Pune	2017-04-06	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	
2	3	2017	Rajkot	2017-04-07	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	
3	4	2017	Indore	2017-04-08	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	
4	5	2017	Bangalore	2017-04-08	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	normal	

In [45]:
```
#Last 5 records of match DataFrame
mdf.tail(5)
```

Out[45]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl
631	632	2016	Raipur	2016-05-22	Delhi Daredevils	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	
632	633	2016	Bangalore	2016-05-24	Gujarat Lions	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	
633	634	2016	Delhi	2016-05-25	Sunrisers Hyderabad	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	
634	635	2016	Delhi	2016-05-27	Gujarat Lions	Sunrisers Hyderabad	Sunrisers Hyderabad	field	normal	
635	636	2016	Bangalore	2016-05-29	Sunrisers Hyderabad	Royal Challengers Bangalore	Sunrisers Hyderabad	bat	normal	

Reading the IPL ***deliveries*** dataset

In [46]:
```
ddf=pd.read_csv("deliveries.csv")
ddf
```

Out[46]:

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler
0	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	1	DA Warner	S Dhawan	TS Mills
1	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	2	DA Warner	S Dhawan	TS Mills
2	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	3	DA Warner	S Dhawan	TS Mills
3	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	4	DA Warner	S Dhawan	TS Mills
4	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	5	DA Warner	S Dhawan	TS Mills
...	...	...	...	...	...	...	...	...	...
150455	636	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20	2	Sachin Baby	CJ Jordan	B Kumar
150456	636	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20	3	Sachin Baby	CJ Jordan	B Kumar
150457	636	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20	4	Iqbal Abdulla	Sachin Baby	B Kumar
150458	636	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20	5	Sachin Baby	Iqbal Abdulla	B Kumar
150459	636	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20	6	Iqbal Abdulla	Sachin Baby	B Kumar

150460 rows × 21 columns

In [47]:
```
#getting no.of rows and columns of deliveries DataFrame
ddf.shape
```

Out[47]:
```
(150460, 21)
```

In [48]:
```
#getting no.of cells in the DataFrame
ddf.size
```

Out[48]:
```
3159660
```

In [49]:
```
#concise summary of the DataFrame
ddf.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150460 entries, 0 to 150459
Data columns (total 21 columns):
 #   Column            Non-Null Count   Dtype 
---  ------            --------------   ----- 
 0   match_id          150460 non-null  int64 
 1   inning            150460 non-null  int64 
 2   batting_team      150460 non-null  object
 3   bowling_team      150460 non-null  object
 4   over              150460 non-null  int64 
 5   ball              150460 non-null  int64 
 6   batsman           150460 non-null  object
 7   non_striker       150460 non-null  object
 8   bowler            150460 non-null  object
 9   is_super_over     150460 non-null  int64 
 10  wide_runs         150460 non-null  int64 
 11  bye_runs          150460 non-null  int64 
 12  legbye_runs       150460 non-null  int64 
 13  noball_runs       150460 non-null  int64 
 14  penalty_runs      150460 non-null  int64 
 15  batsman_runs      150460 non-null  int64 
 16  extra_runs        150460 non-null  int64 
 17  total_runs        150460 non-null  int64 
 18  player_dismissed  7438 non-null    object
 19  dismissal_kind    7438 non-null    object
 20  fielder           5369 non-null    object
dtypes: int64(13), object(8)
memory usage: 24.1+ MB
```

In [50]:
```
#getting the summary statistics for numerical data
ddf.describe()
```

Out[50]:

	match_id	inning	over	ball	is_super_over	wide_runs
count	150460.000000	150460.000000	150460.000000	150460.000000	150460.000000	150460.000000
mean	318.281317	1.482188	10.142649	3.616483	0.000538	0.037498
std	182.955531	0.501768	5.674338	1.807698	0.023196	0.257398
min	1.000000	1.000000	1.000000	1.000000	0.000000	0.000000
25%	161.000000	1.000000	5.000000	2.000000	0.000000	0.000000
50%	319.000000	1.000000	10.000000	4.000000	0.000000	0.000000
75%	476.000000	2.000000	15.000000	5.000000	0.000000	0.000000
max	636.000000	4.000000	20.000000	9.000000	1.000000	5.000000

In [51]:
```
#top 5 records of deliveries Dataframe
ddf.head(5)
```

Out[51]:

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler	is_su
0	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	1	DA Warner	S Dhawan	TS Mills	
1	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	2	DA Warner	S Dhawan	TS Mills	
2	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	3	DA Warner	S Dhawan	TS Mills	
3	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	4	DA Warner	S Dhawan	TS Mills	
4	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	5	DA Warner	S Dhawan	TS Mills	

5 rows × 21 columns

In [52]:
```
#top 5 records of deliveries Dataframe
ddf.tail(5)
```

Out[52]:

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler
150455	636	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20	2	Sachin Baby	CJ Jordan	B Kumar
150456	636	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20	3	Sachin Baby	CJ Jordan	B Kumar
150457	636	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20	4	Iqbal Abdulla	Sachin Baby	B Kumar
150458	636	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20	5	Sachin Baby	Iqbal Abdulla	B Kumar
150459	636	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20	6	Iqbal Abdulla	Sachin Baby	B Kumar

5 rows × 21 columns

Anaylising the DataFrames

In [53]:
```
#getting the total runs for each innings
#increasing each 1st innings score by 1: inorder to assume it as target for 2nd inn
match_score=ddf.groupby(['match_id','inning']).sum()['total_runs'].reset_index()
match_score.loc[match_score['inning'] == 1, 'total_runs'] += 1

match_score
```

Out[53]:

	match_id	inning	total_runs
0	1	1	208
1	1	2	172
2	2	1	185
3	2	2	187
4	3	1	184
...	...	...	...
1279	634	2	140
1280	635	1	163
1281	635	2	163
1282	636	1	209
1283	636	2	200

1284 rows × 3 columns

In [54]:
```
#getting only first innings score from match DataFrame
inning_1_score=match_score[match_score['inning']==1]
inning_1_score
```

Out[54]:

	match_id	inning	total_runs
0	1	1	208
2	2	1	185
4	3	1	184
6	4	1	164
8	5	1	158
...	...	...	...
1274	632	1	139
1276	633	1	159
1278	634	1	163
1280	635	1	163
1282	636	1	209

636 rows × 3 columns

In [55]:
```
#adding match_id and total_runs of 1st innings as columns in the match dataFrame
mdf=mdf.merge(inning_1_score[['match_id','total_runs']],left_on='id',right_on='matc
mdf.head(5)
```

Out[55]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_ap
0	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	
1	2	2017	Pune	2017-04-06	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	
2	3	2017	Rajkot	2017-04-07	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	
3	4	2017	Indore	2017-04-08	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	
4	5	2017	Bangalore	2017-04-08	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	normal	

In [56]:
```
#getting the names of unique IPL teams
mdf['team1'].unique()
```

Out[56]:
```
array(['Sunrisers Hyderabad', 'Mumbai Indians', 'Gujarat Lions',
       'Rising Pune Supergiant', 'Royal Challengers Bangalore',
       'Kolkata Knight Riders', 'Delhi Daredevils', 'Kings XI Punjab',
       'Chennai Super Kings', 'Rajasthan Royals', 'Deccan Chargers',
       'Kochi Tuskers Kerala', 'Pune Warriors', 'Rising Pune Supergiants'],
      dtype=object)
```

In [57]:
```
#creating a list of unique teams that are still playing still registered in IPL
teams = [
'Sunrisers Hyderabad',
'Mumbai Indians',
'Royal Challengers Bangalore',
'Kolkata Knight Riders',
'Kings XI Punjab',
'Chennai Super Kings',
'Rajasthan Royals',
'Delhi Capitals'
]
```

In [58]:
```
#replacing some old teams names to their updated names
mdf['team1']=mdf['team1'].str.replace('Delhi Daredevils','Delhi Capitals')
mdf['team2']=mdf['team2'].str.replace('Delhi Daredevils','Delhi Capitals')

mdf['team1']=mdf['team1'].str.replace('Deccan Chargers','Sunrisers Hyderabad')
mdf['team2']=mdf['team2'].str.replace('Deccan Chargers','Sunrisers Hyderabad')
```

In [59]:
```
#keep only those teams which are in list teams
mdf=mdf[mdf['team1'].isin(teams)]
mdf=mdf[mdf['team2'].isin(teams)]
```

In [60]:
```
#checking the null values in the match dataFrame
mdf.isnull().sum()
```

Out[60]:
```
id                  0
season              0
city                7
date                0
team1               0
team2               0
toss_winner         0
toss_decision       0
result              0
dl_applied          0
winner              2
win_by_runs         0
win_by_wickets      0
player_of_match     2
venue               0
umpire1             1
umpire2             1
umpire3           521
match_id            0
total_runs          0
dtype: int64
```

In [61]:
```
#checking the updated match dataFrame
mdf.shape
```

Out[61]:
```
(521, 20)
```

In [62]:
```
#excluding those matches where the match is interrupted due to rain and DL methods
mdf=mdf[mdf['dl_applied']==0]
```

In [63]:
```
#checking for updation
mdf['dl_applied'].value_counts()
```

Out[63]:
```
dl_applied
0    509
Name: count, dtype: int64
```

In [64]:
```
#modifying the match Dataframe with only those columns we required for further proc
mdf=mdf[['match_id','city','winner','total_runs']]
mdf
```

Out[64]:

	match_id	city	winner	total_runs
0	1	Hyderabad	Sunrisers Hyderabad	208
4	5	Bangalore	Royal Challengers Bangalore	158
6	7	Mumbai	Mumbai Indians	179
7	8	Indore	Kings XI Punjab	149
9	10	Mumbai	Mumbai Indians	159
...	...	...	...	...
627	628	Raipur	Delhi Daredevils	159
630	631	Kolkata	Kolkata Knight Riders	172
631	632	Raipur	Royal Challengers Bangalore	139
633	634	Delhi	Sunrisers Hyderabad	163
635	636	Bangalore	Sunrisers Hyderabad	209

509 rows × 4 columns

In [65]:
```
#adding match_id column to delivery dataframe
ddf=mdf.merge(ddf,on='match_id')
ddf
```

Out[65]:

	match_id	city	winner	total_runs_x	inning	batting_team	bowling_team	over
0	1	Hyderabad	Sunrisers Hyderabad	208	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1
1	1	Hyderabad	Sunrisers Hyderabad	208	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1
2	1	Hyderabad	Sunrisers Hyderabad	208	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1
3	1	Hyderabad	Sunrisers Hyderabad	208	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1
4	1	Hyderabad	Sunrisers Hyderabad	208	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1
...	...	...	...	...	...	...	...	...
121482	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121483	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121484	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121485	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121486	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20

121487 rows × 24 columns

In [66]:
```
#checking for those records of 2nd innings
ddf=ddf[ddf['inning']==2]
ddf.shape
```

Out[66]:
```
(58704, 24)
```

In [67]:
```
# calculating current score
ddf['current_score'] = ddf.groupby('match_id')['total_runs_y'].cumsum()
ddf
```

Out[67]:

	match_id	city	winner	total_runs_x	inning	batting_team	bowling_team	over
125	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
126	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
127	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
128	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
129	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
...	...	...	...	...	...	...	...	...
121482	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121483	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121484	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121485	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121486	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20

58704 rows × 25 columns

In [68]:
```
#calculating the runs left to achive the requireed target after each delivery
ddf['runs_left']=ddf['total_runs_x'] - ddf['current_score']
ddf
```

Out[68]:

	match_id	city	winner	total_runs_x	inning	batting_team	bowling_team	over
125	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
126	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
127	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
128	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
129	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
...	...	...	...	...	...	...	...	...
121482	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121483	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121484	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121485	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121486	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20

58704 rows × 26 columns

In [69]:
```
#calculating the balls left after each delivery
ddf['balls_left']=126-(ddf['over']*6 + ddf['ball'])
ddf
```

Out[69]:

	match_id	city	winner	total_runs_x	inning	batting_team	bowling_team	over
125	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
126	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
127	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
128	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
129	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
...	...	...	...	...	...	...	...	...
121482	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121483	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121484	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121485	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121486	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20

58704 rows × 27 columns

Analysis and manipulation of wickets fallen

In [70]:
```
ddf['player_dismissed']=ddf['player_dismissed'].fillna('0')
ddf
```

Out[70]:

	match_id	city	winner	total_runs_x	inning	batting_team	bowling_team	over
125	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
126	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
127	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
128	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
129	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
...	...	...	...	...	...	...	...	...
121482	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121483	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121484	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121485	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121486	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20

58704 rows × 27 columns

In [71]:
```
#manipulating player_dismissed column
ddf['player_dismissed']=ddf['player_dismissed'].apply(lambda x:'1' if x!='0' else '
ddf
```

Out[71]:

	match_id	city	winner	total_runs_x	inning	batting_team	bowling_team	over
125	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
126	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
127	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
128	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
129	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
...	...	...	...	...	...	...	...	...
121482	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121483	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121484	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121485	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121486	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20

58704 rows × 27 columns

In [72]:
```
#checking for wickets left
ddf['player_dismissed']=ddf['player_dismissed'].astype('int')
wickets = ddf.groupby('match_id')['player_dismissed'].cumsum().values
ddf['wickets_left']=10-wickets
ddf
```

Out[72]:

	match_id	city	winner	total_runs_x	inning	batting_team	bowling_team	over
125	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
126	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
127	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
128	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
129	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
...	...	...	...	...	...	...	...	...
121482	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121483	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121484	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121485	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121486	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20

58704 rows × 28 columns

Analysis and manipulation of balls bowled

In [73]:
```
#calculating balls left after each delivery
ddf['balls_played']=120-ddf['balls_left']
ddf
```

Out[73]:

	match_id	city	winner	total_runs_x	inning	batting_team	bowling_team	over
125	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
126	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
127	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
128	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
129	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
...	...	...	...	...	...	...	...	...
121482	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121483	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121484	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121485	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121486	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20

58704 rows × 29 columns

In [74]:
```
#converting balls palayed into overs
ball_to_over=ddf['balls_played']/6
ball_to_over
```

Out[74]:
```
125         0.166667
126         0.333333
127         0.500000
128         0.666667
129         0.833333
             ...
121482     19.333333
121483     19.500000
121484     19.666667
121485     19.833333
121486     20.000000
Name: balls_played, Length: 58704, dtype: float64
```

In [75]:
```
#for calculating current run rate
(ddf['current_score']*6)/ddf['balls_played']
```

Out[75]:
```
125         6.000000
126         3.000000
127         2.000000
128         4.500000
129         8.400000
             ...
121482     10.034483
121483      9.948718
121484      9.915254
121485      9.882353
121486     10.000000
Length: 58704, dtype: float64
```

In [76]:
```
#adding current run rate column in th delivey dataFrame.
#crr:after each delivery
ddf['crr']=(ddf['current_score']*6)/ddf['balls_played']
ddf
```

Out[76]:

	match_id	city	winner	total_runs_x	inning	batting_team	bowling_team	over
125	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
126	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
127	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
128	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
129	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
...	...	...	...	...	...	...	...	...
121482	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121483	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121484	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121485	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121486	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20

58704 rows × 30 columns

In [77]:
```
#calculating and adding required run rate column in the delivey dataFrame
ddf['rrr'] = (ddf['runs_left'] * 6) / ddf['balls_left']
ddf
```

Out[77]:

	match_id	city	winner	total_runs_x	inning	batting_team	bowling_team	over
125	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
126	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
127	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
128	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
129	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
...	...	...	...	...	...	...	...	...
121482	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121483	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121484	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121485	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121486	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20

58704 rows × 31 columns

Analysing the result of the match:

In [78]:
```
def result(row):
    return 1 if row['batting_team'] == row['winner'] else 0
```

In [79]:
```
ddf['result']=ddf.apply(result,axis=1)
ddf
```

Out[79]:

	match_id	city	winner	total_runs_x	inning	batting_team	bowling_team	over
125	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
126	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
127	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
128	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
129	1	Hyderabad	Sunrisers Hyderabad	208	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1
...	...	...	...	...	...	...	...	...
121482	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121483	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121484	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121485	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20
121486	636	Bangalore	Sunrisers Hyderabad	209	2	Royal Challengers Bangalore	Sunrisers Hyderabad	20

58704 rows × 32 columns

In [80]:
```
#final required dataframe
final_df=ddf[['batting_team','bowling_team','city','runs_left','balls_left','wicket
final_df
```

Out[80]:

	batting_team	bowling_team	city	runs_left	balls_left	wickets_left	total_runs_x	
125	Royal Challengers Bangalore	Sunrisers Hyderabad	Hyderabad	207	119	10	208	
126	Royal Challengers Bangalore	Sunrisers Hyderabad	Hyderabad	207	118	10	208	
127	Royal Challengers Bangalore	Sunrisers Hyderabad	Hyderabad	207	117	10	208	
128	Royal Challengers Bangalore	Sunrisers Hyderabad	Hyderabad	205	116	10	208	
129	Royal Challengers Bangalore	Sunrisers Hyderabad	Hyderabad	201	115	10	208	
...	...	...	...	...	...	...	...	
121482	Royal Challengers Bangalore	Sunrisers Hyderabad	Bangalore	15	4	4	209	1
121483	Royal Challengers Bangalore	Sunrisers Hyderabad	Bangalore	15	3	3	209	
121484	Royal Challengers Bangalore	Sunrisers Hyderabad	Bangalore	14	2	3	209	
121485	Royal Challengers Bangalore	Sunrisers Hyderabad	Bangalore	13	1	3	209	
121486	Royal Challengers Bangalore	Sunrisers Hyderabad	Bangalore	9	0	3	209	1

58704 rows × 10 columns

In [81]:
```
final_df=final_df.sample(final_df.shape[0])
```

Handling some undesirred records

In [82]:
```
final_df.sample()
```

Out[82]:

	batting_team	bowling_team	city	runs_left	balls_left	wickets_left	total_runs_x	
84712	Kings XI Punjab	Royal Challengers Bangalore	Chandigarh	146	78	8	191	6.4

In [83]:
```
final_df.isnull().sum()
```

Out[83]:
```
batting_team      0
bowling_team      0
city            832
runs_left         0
balls_left        0
wickets_left      0
total_runs_x      0
crr               0
rrr               5
result            0
dtype: int64
```

In [84]:
```
#dropping those 5 records where the value is null
#also there are only 5 records removing this is not going to affect the result.
final_df=final_df.dropna(subset=['rrr'])
```

In [85]:
```
final_df.isnull().sum()
```

Out[85]:
```
batting_team      0
bowling_team      0
city            832
runs_left         0
balls_left        0
wickets_left      0
total_runs_x      0
crr               0
rrr               0
result            0
dtype: int64
```

In [86]:
```
#checking the mode value of city column
mode_city=final_df['city'].mode()[0]
mode_city
```

Out[86]:
```
'Mumbai'
```

In [87]:
```
#replacing null records with mode value of city column
final_df['city'].fillna(mode_city, inplace=True)
```

In [88]:
```
#checking for null values in the final dataframe
final_df.isnull().sum()
```

Out[88]:
```
batting_team    0
bowling_team    0
city            0
runs_left       0
balls_left      0
wickets_left    0
total_runs_x    0
crr             0
rrr             0
result          0
dtype: int64
```

In [89]:
```
#handling the infinete value
final_df=final_df[final_df['balls_left']!=0]
```

In [90]:
```
final_df.reset_index(drop=True, inplace=True)
```

In [91]:
```
ddf['match_id'].rank(method='dense').astype(int)
```

Out[91]:
```
125         1
126         1
127         1
128         1
129         1
          ...
121482    508
121483    508
121484    508
121485    508
121486    508
Name: match_id, Length: 58704, dtype: int32
```

In [92]:
```
is_continuous = final_df.index.is_monotonic_increasing

if is_continuous:
    print("Rows are aligned in a continuous manner.")
else:
    print("Rows are not aligned in a continuous manner.")
```

```
Rows are aligned in a continuous manner.
```

Machine learning:

In [93]:
```
#importing libraries for model building

from sklearn.model_selection import train_test_split
```

In [94]:
```
X=final_df.iloc[:,:-1]
y=final_df.iloc[:,-1]

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=1)
```

In [95]:
```
X_train
```

Out[95]:

	batting_team	bowling_team	city	runs_left	balls_left	wickets_left	total_runs_x	
23892	Delhi Daredevils	Royal Challengers Bangalore	Bangalore	182	117	10	192	20.0
42207	Royal Challengers Bangalore	Mumbai Indians	Mumbai	182	114	10	188	6.0
42701	Delhi Daredevils	Mumbai Indians	Mumbai	175	113	8	179	3.4
55716	Mumbai Indians	Kolkata Knight Riders	Abu Dhabi	52	13	6	164	6.2
31517	Kings XI Punjab	Royal Challengers Bangalore	Bangalore	62	69	9	127	7.6
...	...	...	...	...	...	...	...	
50057	Rajasthan Royals	Delhi Daredevils	Jaipur	0	13	9	155	8.6
32511	Mumbai Indians	Chennai Super Kings	Mumbai	104	65	9	174	7.6
5192	Rajasthan Royals	Royal Challengers Bangalore	Pune	77	13	1	181	5.8
12172	Chennai Super Kings	Royal Challengers Bangalore	Bangalore	150	92	9	172	4.7
33003	Chennai Super Kings	Rajasthan Royals	Chennai	96	78	9	148	7.4

46804 rows × 9 columns

In [96]:
```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

''' This is specifying a transformation for three columns
 ('batting_team', 'bowling_team', 'city').: as they are non-numerical '''
trf=ColumnTransformer([
    ('trf',OneHotEncoder(sparse=False,drop='first'),['batting_team','bowling_team',
],remainder='passthrough')
```

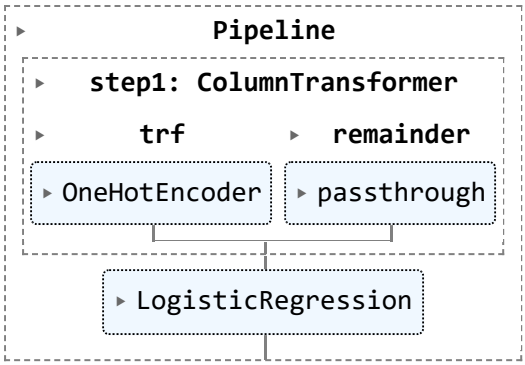
In [97]:
```
from sklearn.linear_model import LogisticRegression
#from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
```

In [98]:
```
#creating a pipline
pipe=Pipeline(steps=[('step1',trf),
                     ('step2',LogisticRegression(solver='liblinear'))
                    ])
```

Fitting and traing the created model.

In [99]:
```
pipe.fit(X_train,y_train)
```

Out[99]:



In [100…
```
y_pred=pipe.predict(X_test)
```

In [101…
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[101]:
```
0.8198444577386548
```

In [102…
```
accuracy_score(y_test,y_pred)*100
```

Out[102]:
```
81.98444577386547
```

In [103…
```
pipe.predict_proba(X_test)[1]
```

Out[103]:
```
array([0.15108502, 0.84891498])
```

Conclusion:

Thus, using logistic regression, we can conclude that the model predicts with a probability of approximately 0.849.

In [104…
```
#pipelining for RandomForest
'''pipe2=Pipeline(steps=[('step1',trf),
                      ('step2',RandomForestClassifier())
                      ])
pipe2.fit(X_train,y_train)
y_pred2=pipe2.predict(X_test)
accuracy_score(y_test,y_pred2)
pipe2.predict_proba(X_test)[5]'''
```

Out[104]:
```
"pipe2=Pipeline(steps=[('step1',trf),\n                      ('step2',RandomForest
Classifier())\n                      ])\npipe2.fit(X_train,y_train)\ny_pred2=pipe
2.predict(X_test)\naccuracy_score(y_test,y_pred2)\npipe2.predict_proba(X_test)[5]"
```

In [105…
```
'''def match_summary(row):
    print("Batting Team-" + row['batting_team'] + " | Bowling Team-" + row['bowling

match_summary(ddf.iloc[6210])  '''
```

Out[105]:
```
'def match_summary(row):\n    print("Batting Team-" + row[\'batting_team\'] + " |
Bowling Team-" + row[\'bowling_team\'] + " | Target- " + str(row[\'total_runs_x
\']))\n\nmatch_summary(ddf.iloc[6210])  '
```

In [106…
```
ddf['match_id']=ddf['match_id'].rank(method='dense').astype(int)
```

In [107…
```
#checking continuous and unique match_id
ddf['match_id'].unique()
```

Out[107]:
```
array([  1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11,  12,  13,
        14,  15,  16,  17,  18,  19,  20,  21,  22,  23,  24,  25,  26,
        27,  28,  29,  30,  31,  32,  33,  34,  35,  36,  37,  38,  39,
        40,  41,  42,  43,  44,  45,  46,  47,  48,  49,  50,  51,  52,
        53,  54,  55,  56,  57,  58,  59,  60,  61,  62,  63,  64,  65,
        66,  67,  68,  69,  70,  71,  72,  73,  74,  75,  76,  77,  78,
        79,  80,  81,  82,  83,  84,  85,  86,  87,  88,  89,  90,  91,
        92,  93,  94,  95,  96,  97,  98,  99, 100, 101, 102, 103, 104,
       105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117,
       118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130,
       131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143,
       144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156,
       157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169,
       170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182,
       183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195,
       196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208,
       209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221,
       222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234,
       235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247,
       248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260,
       261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273,
       274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286,
       287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299,
       300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312,
       313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325,
       326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338,
       339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351,
       352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364,
       365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377,
       378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390,
       391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403,
       404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416,
       417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429,
       430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442,
       443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455,
       456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468,
       469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481,
       482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494,
       495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507,
       508])
```

In [128…
```
ddf[ddf['match_id']==10].head(2)
```

Out[128]:

	match_id	city	winner	total_runs_x	inning	batting_team	bowling_team	over	ball	bats
2298	10	Delhi	Kolkata Knight Riders	169	2	Kolkata Knight Riders	Delhi Daredevils	1	1	Gan
2299	10	Delhi	Kolkata Knight Riders	169	2	Kolkata Knight Riders	Delhi Daredevils	1	2	Gan

2 rows × 32 columns

A function to calculate and analysise the match progression factors.

In [109…
```
def match_progression(x_df,match_id,pipe):
    match = x_df[x_df['match_id'] == match_id]
    match = match[(match['ball'] == 6)]
    temp_df = match[['batting_team','bowling_team','city','runs_left','balls_left',
```

```
    temp_df = temp_df[temp_df['balls_left'] != 0]
    result = pipe.predict_proba(temp_df)
    temp_df['lose'] = np.round(result.T[0]*100,1)
    temp_df['win'] = np.round(result.T[1]*100,1)
    temp_df['end_of_over'] = range(1,temp_df.shape[0]+1)

    target = temp_df['total_runs_x'].values[0]
    runs = list(temp_df['runs_left'].values)
    new_runs = runs[:]
    runs.insert(0,target)
    temp_df['runs_after_over'] = np.array(runs)[:-1] - np.array(new_runs)
    wickets = list(temp_df['wickets_left'].values)
    new_wickets = wickets[:]
    new_wickets.insert(0,10)
    wickets.append(0)
    w = np.array(wickets)
    nw = np.array(new_wickets)
    temp_df['wickets_in_over'] = (nw - w)[0:temp_df.shape[0]]

    print("Target-",target)
    temp_df = temp_df[['end_of_over','runs_after_over','wickets_in_over','lose','wi
    return temp_df,target
```

In [110…

```
#passing for match_id=10
temp_df,target = match_progression(ddf,10,pipe)
temp_df
```

```
Target- 169
```

Out[110]:

	end_of_over	runs_after_over	wickets_in_over	lose	win
2303	1	6	1	62.3	37.7
2309	2	13	1	64.1	35.9
2315	3	2	1	81.6	18.4
2321	4	7	0	79.5	20.5
2327	5	6	0	78.5	21.5
2333	6	15	0	65.1	34.9
2339	7	3	0	68.2	31.8
2345	8	9	0	62.3	37.7
2351	9	10	0	54.3	45.7
2358	10	13	0	41.2	58.8
2365	11	9	0	35.0	65.0
2371	12	6	0	33.7	66.3
2378	13	11	0	25.5	74.5
2384	14	15	0	14.9	85.1
2390	15	10	1	19.6	80.4
2396	16	6	0	18.6	81.4
2402	17	6	0	17.7	82.3
2408	18	5	1	30.2	69.8
2415	19	8	0	26.5	73.5

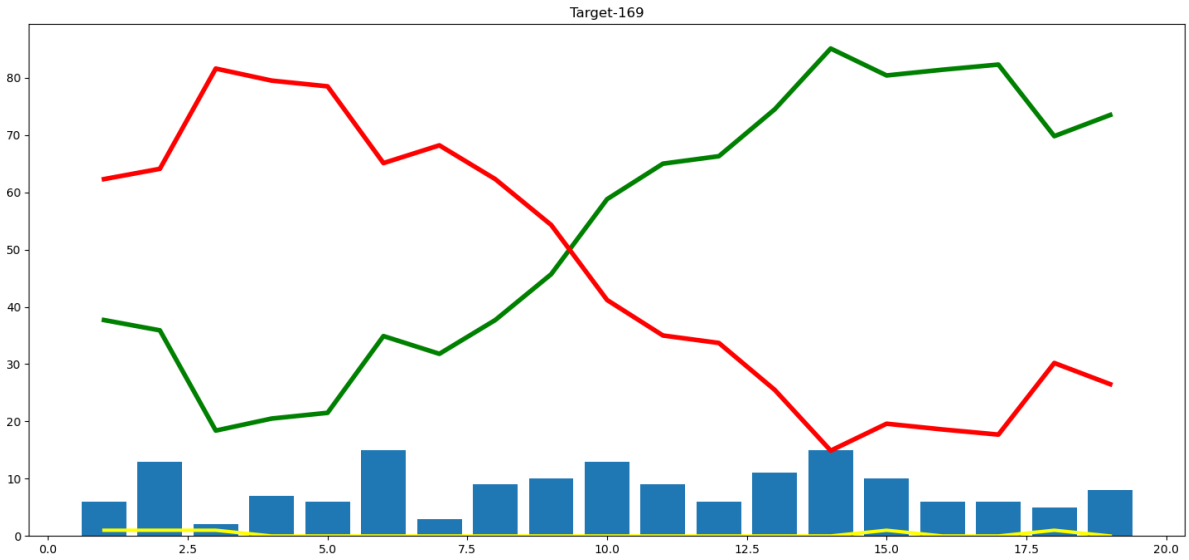
Visualizing the Match Result

In [126…

```
plt.figure(figsize=(18,8))
plt.plot(temp_df['end_of_over'],temp_df['wickets_in_over'],color='yellow',linewidth
plt.plot(temp_df['end_of_over'],temp_df['win'],color='green',linewidth=4)
plt.plot(temp_df['end_of_over'],temp_df['lose'],color='red',linewidth=4)
plt.bar(temp_df['end_of_over'],temp_df['runs_after_over'])
plt.title('Target-' + str(target))
```

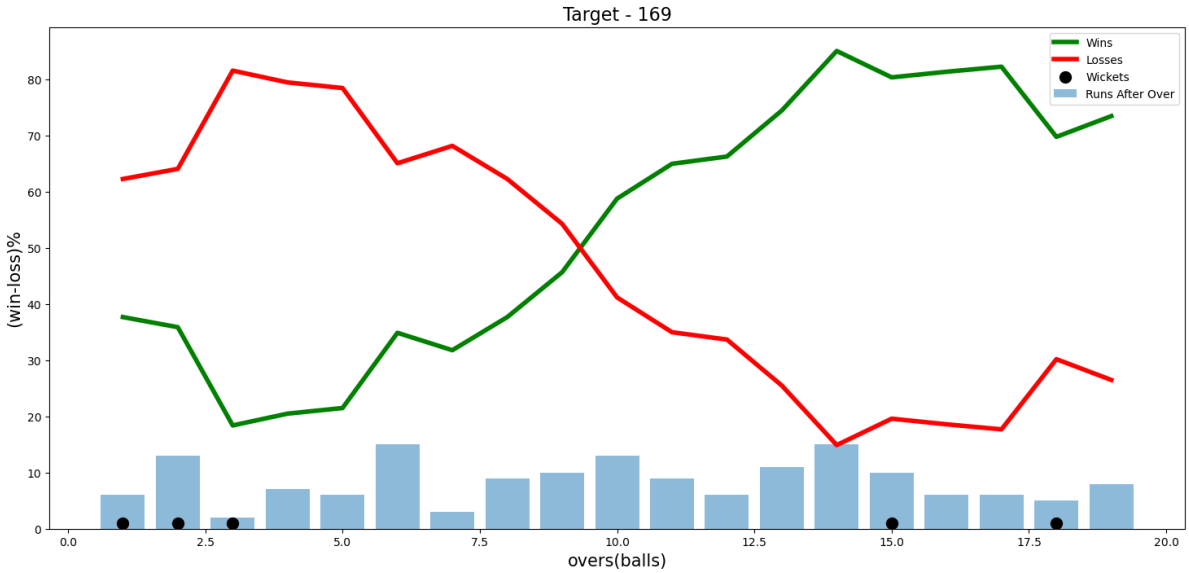
Out[126]:

```
Text(0.5, 1.0, 'Target-169')
```



In [120…

```
import matplotlib.pyplot as plt

plt.figure(figsize=(18, 8))

# Plot wins and losses
plt.plot(temp_df['end_of_over'], temp_df['win'], color='green', linewidth=4, label=
plt.plot(temp_df['end_of_over'], temp_df['lose'], color='red', linewidth=4, label='

# Bar plot
plt.bar(temp_df['end_of_over'], temp_df['runs_after_over'], label='Runs After Over'

# fall of wickets
wickets_mask = temp_df['wickets_in_over'] > 0
plt.scatter(temp_df.loc[wickets_mask, 'end_of_over'], temp_df.loc[wickets_mask, 'wi
            color='black', marker='o', s=100, label='Wickets')


plt.title('Target - ' + str(target), color='black', fontsize=16)
plt.xticks(color='black')
plt.yticks(color='black')
plt.xlabel('overs(balls)',color='black',size=15)
plt.ylabel('(win-loss)%',color='black',size=15)
plt.legend()
plt.show()
```



Insight from the visualization

Analyzing the slope of their line before and after this point of intersection can revealchanges in their run rate and momentum.

The size of the gap between the lines at the end of their innings indicates the margin of victory i.e the chasing team easily chased the target

In [113…
```
#teams
```

In [114…
```
#ddf['city'].unique()
```

In [115…
```
#saving the pipe object to pipe.pkl file and open with write mode
import pickle
pickle.dump(pipe,open('pipe.pkl','wb'))
```

Frontend Part: Website View

In [ ]:
```
## Front end part

#importing required libraries
import streamlit as st
import pickle
import pandas as pd

#list of unique teams
teams = ['Sunrisers Hyderabad', 'Mumbai Indians', 'Royal Challengers Bangalore',
         'Kolkata Knight Riders', 'Kings XI Punjab', 'Chennai Super Kings',
         'Rajasthan Royals', 'Delhi Capitals']

#list of unique places
cities = ['Hyderabad', 'Bangalore', 'Mumbai', 'Indore', 'Kolkata', 'Delhi',
          'Chandigarh', 'Jaipur', 'Chennai', 'Cape Town', 'Port Elizabeth',
          'Durban', 'Centurion', 'East London', 'Johannesburg', 'Kimberley',
          'Bloemfontein', 'Ahmedabad', 'Cuttack', 'Nagpur', 'Dharamsala',
          'Visakhapatnam', 'Pune', 'Raipur', 'Ranchi', 'Abu Dhabi',
          'Sharjah', 'Mohali', 'Bengaluru']

try:
    pipe = pickle.load(open('pipe.pkl', 'rb'))
```

```
except FileNotFoundError:
    st.error("Error: Model file 'pipe.pkl' not found. Please ensure the model file
    st.stop()
except Exception as e:
    st.error(f"Error loading the model: {e}")
    # Log the exception for further investigation
    raise

#website view
st.title('IPL Win Predictor')

col1, col2 = st.columns(2)

with col1:
    batting_team = st.selectbox('Select the batting team', sorted(teams))
with col2:
    bowling_team = st.selectbox('Select the bowling team', sorted(teams))

selected_city = st.selectbox('Select host city', sorted(cities))

target = st.number_input('Target')

col3, col4, col5 = st.columns(3)

with col3:
    score = st.number_input('Score')
with col4:
    overs = st.number_input('Overs completed')
with col5:
    wickets = st.number_input('Wickets out')

if st.button('Predict Probability'):
    runs_left = target - score
    balls_left = 120 - (overs * 6)
    wickets_left = 10 - wickets
    crr = score / overs

    if balls_left > 0:
        rrr = (runs_left * 6) / balls_left
    else:
        rrr = 0

    input_df = pd.DataFrame({
        'batting_team': [batting_team],
        'bowling_team': [bowling_team],
        'city': [selected_city],
        'runs_left': [runs_left],
        'balls_left': [balls_left],
        'wickets_left': [wickets_left],
        'total_runs_x': [target],
        'crr': [crr],
        'rrr': [rrr]
    })


    try:
        result = pipe.predict_proba(input_df)
        win_probability = result[0][1]
        loss_probability = result[0][0]

        st.header(f"{batting_team} Win Probability: {round(win_probability * 100)}%
        st.header(f"{bowling_team} Loss Probability: {round(loss_probability * 100)

    except Exception as e:
```

```
        st.error(f"Error predicting probability: {e}")
        # Log the exception for further investigation
        raise
```