“Cricket Clairvoyant: Real-time IPL Match Outcome Predictor using Machine Learning”

Introduction

```
This project is an innovative application of machine learning to the exciting world of
cricket. Utilizing historical data from the Indian Premier League (IPL), the project
aims to predict the outcome of a cricket match in real-time, providing a dynamic and
engaging experience for cricket enthusiasts.


1. Data Preprocessing and Exploration
    - The project begins with loading the IPL match data and performing initial data
exploration.
    - This includes understanding the structure of the data, checking for missing
values, and getting a sense of the distribution of different variables.

2. Feature Engineering
    - New features are created from the existing data to capture important
information.
    - This includes calculating the current and required run rates, the number of runs
left, the number of balls left, and the number of wickets left.

3. Model Training
    - A logistic regression model is trained on the preprocessed data.
    - The model learns to predict the outcome of a match based on the engineered
features.

4. Match Progression Analysis
    - A function `match_progression` is defined to analyze the progression of a
specific match.
    - It calculates the win and loss probabilities for each over in the match and
returns a dataframe with these details.

5. Interactive Prediction
    - The user is prompted to input details about a new match.
    - The trained model is used to predict the win and loss probabilities for the
input match details.
    - The predicted probabilities are then printed out.

6. Visualization
    - A multi-faceted plot is generated to visualize the progression of a match.
    - It shows the number of wickets taken, the predicted win and loss probabilities,
and the number of runs scored after each over.

7. User Interaction
    - The code includes interactive elements, allowing users to input match details
and receive real-time predictions.
    - This enhances the user experience and makes the project more engaging.
```

This project is a comprehensive application of data science and machine learning to predict the outcome of IPL matches, providing a dynamic and engaging tool for cricket enthusiasts. It demonstrates the power of data science in transforming the way we understand and engage with sports.

Importing Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score
import streamlit as st
import pickle
```

In [2]:
```
#Loding the dataset matches and deliveries.

match=pd.read_csv("C:/Internship/IPL Prediction/matches.csv")
delivery=pd.read_csv("C:/Internship/IPL Prediction/deliveries.csv")
```

EXPLORATORY DATA ANALYSIS

In [3]:
```
#head(): returns first five rows of Match dataset and delivery dataset.

match.head()
```

Out[3]:

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

In [4]:
```
delivery.head()
```

Out[4]:

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

5 rows × 21 columns

In [5]:
```
#tail(): returns last 5 rows of Match dataset.

match.tail()
```

Out[5]:

	id	Season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applie
751	11347	IPL-2019	Mumbai	05-05-2019	Kolkata Knight Riders	Mumbai Indians	Mumbai Indians	field	normal	
752	11412	IPL-2019	Chennai	07-05-2019	Chennai Super Kings	Mumbai Indians	Chennai Super Kings	bat	normal	
753	11413	IPL-2019	Visakhapatnam	08-05-2019	Sunrisers Hyderabad	Delhi Capitals	Delhi Capitals	field	normal	
754	11414	IPL-2019	Visakhapatnam	10-05-2019	Delhi Capitals	Chennai Super Kings	Chennai Super Kings	field	normal	
755	11415	IPL-2019	Hyderabad	12-05-2019	Mumbai Indians	Chennai Super Kings	Mumbai Indians	bat	normal	

In [6]:
```
delivery.tail()
```

Out[6]:

	match_id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler	is_super_o
179073	11415	2	Chennai Super Kings	Mumbai Indians	20	2	RA Jadeja	SR Watson	SL Malinga	
179074	11415	2	Chennai Super Kings	Mumbai Indians	20	3	SR Watson	RA Jadeja	SL Malinga	
179075	11415	2	Chennai Super Kings	Mumbai Indians	20	4	SR Watson	RA Jadeja	SL Malinga	
179076	11415	2	Chennai Super Kings	Mumbai Indians	20	5	SN Thakur	RA Jadeja	SL Malinga	
179077	11415	2	Chennai Super Kings	Mumbai Indians	20	6	SN Thakur	RA Jadeja	SL Malinga	

5 rows × 21 columns

In [8]:
```
#describe() : returns description of match data and delivery data in the dataset.

match.describe()
```

Out[8]:

	id	dl_applied	win_by_runs	win_by_wickets
count	756.000000	756.000000	756.000000	756.000000
mean	1792.178571	0.025132	13.283069	3.350529
std	3464.478148	0.156630	23.471144	3.387963
min	1.000000	0.000000	0.000000	0.000000
25%	189.750000	0.000000	0.000000	0.000000
50%	378.500000	0.000000	0.000000	4.000000
75%	567.250000	0.000000	19.000000	6.000000
max	11415.000000	1.000000	146.000000	10.000000

In [9]:
```
delivery.describe()
```

Out[9]:

	match_id	inning	over	ball	is_super_over	wide_runs	bye
count	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.000000	179078.0
mean	1802.252957	1.482952	10.162488	3.615587	0.000452	0.036721	0.0
std	3472.322805	0.502074	5.677684	1.806966	0.021263	0.251161	0.1
min	1.000000	1.000000	1.000000	1.000000	0.000000	0.000000	0.0
25%	190.000000	1.000000	5.000000	2.000000	0.000000	0.000000	0.0
50%	379.000000	1.000000	10.000000	4.000000	0.000000	0.000000	0.0
75%	567.000000	2.000000	15.000000	5.000000	0.000000	0.000000	0.0
max	11415.000000	5.000000	20.000000	9.000000	1.000000	5.000000	4.0

In [11]:
```
#info() : prints information about the match dataset and delivery dataset.

match.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 756 entries, 0 to 755
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   id               756 non-null    int64 
 1   Season           756 non-null    object
 2   city             749 non-null    object
 3   date             756 non-null    object
 4   team1            756 non-null    object
 5   team2            756 non-null    object
 6   toss_winner      756 non-null    object
 7   toss_decision    756 non-null    object
 8   result           756 non-null    object
 9   dl_applied       756 non-null    int64 
 10  winner           752 non-null    object
 11  win_by_runs      756 non-null    int64 
 12  win_by_wickets   756 non-null    int64 
 13  player_of_match  752 non-null    object
 14  venue            756 non-null    object
 15  umpire1          754 non-null    object
 16  umpire2          754 non-null    object
 17  umpire3          119 non-null    object
dtypes: int64(4), object(14)
memory usage: 106.4+ KB
```

In [12]:
```
delivery.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 179078 entries, 0 to 179077
Data columns (total 21 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   match_id          179078 non-null  int64
 1   inning            179078 non-null  int64
 2   batting_team      179078 non-null  object
 3   bowling_team      179078 non-null  object
 4   over              179078 non-null  int64
 5   ball              179078 non-null  int64
 6   batsman           179078 non-null  object
 7   non_striker       179078 non-null  object
 8   bowler            179078 non-null  object
 9   is_super_over     179078 non-null  int64
 10  wide_runs         179078 non-null  int64
 11  bye_runs          179078 non-null  int64
 12  legbye_runs       179078 non-null  int64
 13  noball_runs       179078 non-null  int64
 14  penalty_runs      179078 non-null  int64
 15  batsman_runs      179078 non-null  int64
 16  extra_runs        179078 non-null  int64
 17  total_runs        179078 non-null  int64
 18  player_dismissed  8834 non-null    object
 19  dismissal_kind    8834 non-null    object
 20  fielder           6448 non-null    object
dtypes: int64(13), object(8)
memory usage: 28.7+ MB
```

In [13]:
```
match.index
```

Out[13]:
```
RangeIndex(start=0, stop=756, step=1)
```

In [14]:
```
#shape is the no of rows and columns of dataset
#shape[0] tells no. of rows

print("no. of matches played: ",match.shape[0])
```

```
no. of matches played:  756
```

In [15]:
```
# In which cities IPL matches were Played?

print("\n cities played at :",match['city'].unique())
```

```
 cities played at : ['Hyderabad' 'Pune' 'Rajkot' 'Indore' 'Bangalore' 'Mumbai' 'Kolkat
a'
 'Delhi' 'Chandigarh' 'Kanpur' 'Jaipur' 'Chennai' 'Cape Town'
 'Port Elizabeth' 'Durban' 'Centurion' 'East London' 'Johannesburg'
 'Kimberley' 'Bloemfontein' 'Ahmedabad' 'Cuttack' 'Nagpur' 'Dharamsala'
 'Kochi' 'Visakhapatnam' 'Raipur' 'Ranchi' 'Abu Dhabi' 'Sharjah' nan
 'Mohali' 'Bengaluru']
```

In [16]:
```
print("venue :", match['venue'].unique())
```

```
venue : ['Rajiv Gandhi International Stadium, Uppal'
 'Maharashtra Cricket Association Stadium'
 'Saurashtra Cricket Association Stadium' 'Holkar Cricket Stadium'
 'M Chinnaswamy Stadium' 'Wankhede Stadium' 'Eden Gardens'
 'Feroz Shah Kotla' 'Punjab Cricket Association IS Bindra Stadium, Mohali'
 'Green Park' 'Punjab Cricket Association Stadium, Mohali'
 'Sawai Mansingh Stadium' 'MA Chidambaram Stadium, Chepauk'
 'Dr DY Patil Sports Academy' 'Newlands' "St George's Park" 'Kingsmead'
 'SuperSport Park' 'Buffalo Park' 'New Wanderers Stadium'
 'De Beers Diamond Oval' 'OUTsurance Oval' 'Brabourne Stadium'
 'Sardar Patel Stadium, Motera' 'Barabati Stadium'
 'Vidarbha Cricket Association Stadium, Jamtha'
 'Himachal Pradesh Cricket Association Stadium' 'Nehru Stadium'
 'Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket Stadium'
 'Subrata Roy Sahara Stadium'
 'Shaheed Veer Narayan Singh International Stadium'
 'JSCA International Stadium Complex' 'Sheikh Zayed Stadium'
 'Sharjah Cricket Stadium' 'Dubai International Cricket Stadium'
 'M. A. Chidambaram Stadium' 'Feroz Shah Kotla Ground'
 'M. Chinnaswamy Stadium' 'Rajiv Gandhi Intl. Cricket Stadium'
 'IS Bindra Stadium' 'ACA-VDCA Stadium']
```

In [17]:
```
#size returns no of elements

match.size
```

Out[17]:
```
13608
```

In [18]:
```
#count() method counts the no. of not empty values for each row

match.count()
```

Out[18]:
```
id                756
Season            756
city              749
date              756
team1             756
team2             756
toss_winner       756
toss_decision     756
result            756
dl_applied        756
winner            752
win_by_runs       756
win_by_wickets    756
player_of_match   752
venue             756
umpire1           754
umpire2           754
umpire3           119
dtype: int64
```

In [19]:
```
#count(axis=1) count no. of non empty values for each column

match.count(axis=1)
```

Out[19]:
```
0      17
1      17
2      17
3      17
4      15
       ..
751    18
752    18
753    15
754    18
755    18
Length: 756, dtype: int64
```

In [20]:
```
total_score_df=delivery.groupby(['match_id', 'inning']).sum()['total_runs'].reset_index
```

```
C:\Users\sharm\AppData\Local\Temp\ipykernel_25660\1018152582.py:1: FutureWarning: The
default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future versi
on, numeric_only will default to False. Either specify numeric_only or select only col
umns which should be valid for the function.
  total_score_df=delivery.groupby(['match_id', 'inning']).sum()['total_runs'].reset_in
dex()
```

In [21]:
```
total_score_df=total_score_df[total_score_df['inning']==1]
total_score_df
```

Out[21]:

	match_id	inning	total_runs
0	1	1	207
2	2	1	184
4	3	1	183
6	4	1	163
8	5	1	157
...	...	...	...
1518	11347	1	143
1520	11412	1	136
1522	11413	1	171
1524	11414	1	155
1526	11415	1	152

756 rows × 3 columns

In [22]:
```
#no. of times teams won matches

match['winner'].value_counts()
```

Out[22]:
```
Mumbai Indians                109
Chennai Super Kings           100
Kolkata Knight Riders          92
Royal Challengers Bangalore    84
Kings XI Punjab                82
Rajasthan Royals               75
Delhi Daredevils               67
Sunrisers Hyderabad            58
Deccan Chargers                29
Gujarat Lions                  13
Pune Warriors                  12
Rising Pune Supergiant         10
Delhi Capitals                 10
Kochi Tuskers Kerala            6
Rising Pune Supergiants         5
Name: winner, dtype: int64
```

In [23]:
```
match['team1'].unique()
```

Out[23]:
```
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',
       'Delhi Capitals'], dtype=object)
```

In [24]:
```
match['team1'].value_counts()
```

Out[24]:
```
Mumbai Indians                101
Kings XI Punjab                91
Chennai Super Kings            89
Royal Challengers Bangalore    85
Kolkata Knight Riders          83
Delhi Daredevils               72
Rajasthan Royals               67
Sunrisers Hyderabad            63
Deccan Chargers                43
Pune Warriors                  20
Gujarat Lions                  14
Rising Pune Supergiant          8
Kochi Tuskers Kerala            7
Rising Pune Supergiants         7
Delhi Capitals                  6
Name: team1, dtype: int64
```

In [25]:
```
teams=[
    'Sunrisers Hyderabad',
    'Mumbai Indians',
    'Royal Challengers Bangalore',
    'Kolkata Knight Riders',
    'Kings XI Punjab',
    'Chennai Super Kings',
    'Rajasthan Royals',
    'Delhi Capitals'
]
```

DATA PREPROCESSING

In [26]:
```
#Find the matches where there was no winner.

match[match['winner'].isnull()==True]
```

Out[26]:

	id	Season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied
300	301	IPL-2011	Delhi	21-05-2011	Delhi Daredevils	Pune Warriors	Delhi Daredevils	bat	no result	0
545	546	IPL-2015	Bangalore	29-04-2015	Royal Challengers Bangalore	Rajasthan Royals	Rajasthan Royals	field	no result	0
570	571	IPL-2015	Bangalore	17-05-2015	Delhi Daredevils	Royal Challengers Bangalore	Royal Challengers Bangalore	field	no result	0
744	11340	IPL-2019	Bengaluru	30-04-2019	Royal Challengers Bangalore	Rajasthan Royals	Rajasthan Royals	field	no result	0

In [27]:
```
#replacing null results with 'draw'
#fillna() replaces null values with specific values

match['winner'].fillna('Draw',inplace=True)
match.iloc[300]
```

Out[27]:
```
id                             301
Season                     IPL-2011
city                          Delhi
date                     21-05-2011
team1              Delhi Daredevils
team2                 Pune Warriors
toss_winner        Delhi Daredevils
toss_decision                 bat
result                  no result
dl_applied                      0
winner                       Draw
win_by_runs                     0
win_by_wickets                  0
player_of_match               NaN
venue              Feroz Shah Kotla
umpire1                  SS Hazare
umpire2                  RJ Tucker
umpire3                       NaN
Name: 300, dtype: object
```

In [28]:
```
#replacing null values.

match[match['winner'].isnull()==True]
```

Out[28]:

	id	Season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs

In [29]:
```
#isnull() tells no. of null values for each column.

match.isnull().sum()
```

Out[29]:
```
id                  0
Season              0
city                7
date                0
team1               0
team2               0
toss_winner         0
toss_decision       0
result              0
dl_applied          0
winner              0
win_by_runs         0
win_by_wickets      0
player_of_match     4
venue               0
umpire1             2
umpire2             2
umpire3           637
dtype: int64
```

In [30]:
```
#matches held in each city

match['city'].value_counts()
```

Out[30]:
```
Mumbai            101
Kolkata            77
Delhi              74
Bangalore          66
Hyderabad          64
Chennai            57
Jaipur             47
Chandigarh         46
Pune               38
Durban             15
Bengaluru          14
Visakhapatnam      13
Centurion          12
Ahmedabad          12
Rajkot             10
Mohali             10
Indore              9
Dharamsala          9
Johannesburg        8
Cuttack             7
Ranchi              7
Port Elizabeth      7
Cape Town           7
Abu Dhabi           7
Sharjah             6
Raipur              6
Kochi               5
Kanpur              4
Nagpur              3
Kimberley           3
East London         3
Bloemfontein        2
Name: city, dtype: int64
```

In [31]:
```
#find missing values in city column

match[match['city'].isnull()==True]
```

Out[31]:

	id	Season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	
461	462	IPL-2014	NaN	19-04-2014	Mumbai Indians	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	Chall Bar
462	463	IPL-2014	NaN	19-04-2014	Kolkata Knight Riders	Delhi Daredevils	Kolkata Knight Riders	bat	normal	0	Dar
466	467	IPL-2014	NaN	23-04-2014	Chennai Super Kings	Rajasthan Royals	Rajasthan Royals	field	normal	0	C
468	469	IPL-2014	NaN	25-04-2014	Sunrisers Hyderabad	Delhi Daredevils	Sunrisers Hyderabad	bat	normal	0	Su Hyd
469	470	IPL-2014	NaN	25-04-2014	Mumbai Indians	Chennai Super Kings	Mumbai Indians	bat	normal	0	C
474	475	IPL-2014	NaN	28-04-2014	Royal Challengers Bangalore	Kings XI Punjab	Kings XI Punjab	field	normal	0	K
476	477	IPL-2014	NaN	30-04-2014	Sunrisers Hyderabad	Mumbai Indians	Mumbai Indians	field	normal	0	Su Hyd

In [32]:
```
#handling missing values

match['city'].fillna('Dubai',inplace=True)
match.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 756 entries, 0 to 755
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   id               756 non-null    int64 
 1   Season           756 non-null    object
 2   city             756 non-null    object
 3   date             756 non-null    object
 4   team1            756 non-null    object
 5   team2            756 non-null    object
 6   toss_winner      756 non-null    object
 7   toss_decision    756 non-null    object
 8   result           756 non-null    object
 9   dl_applied       756 non-null    int64 
 10  winner           756 non-null    object
 11  win_by_runs      756 non-null    int64 
 12  win_by_wickets   756 non-null    int64 
 13  player_of_match  752 non-null    object
 14  venue            756 non-null    object
 15  umpire1          754 non-null    object
 16  umpire2          754 non-null    object
 17  umpire3          119 non-null    object
dtypes: int64(4), object(14)
memory usage: 106.4+ KB
```

In [33]:
```
#umpire3 column has null values.so it can be remove

match=match.drop(["umpire3"],axis=1)
match.head(2)
```

Out[33]:

	id	Season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	
0	1	IPL-2017	Hyderabad	05-04-2017	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	Su Hyd
1	2	IPL-2017	Pune	06-04-2017	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0	Sup

In [34]:
```
#Find the teams which had participated in IPL in any season?

print("\n Teams participated: \n",match["team1"].unique())
```

```
 Teams participated:
['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'
'Delhi Capitals']
```

In [35]:
```
#add new column season i.e. in which year match was held

match['season']=pd.DatetimeIndex(match['date']).year
match.head()
```

```
C:\Users\sharm\AppData\Local\Temp\ipykernel_25660\1681766259.py:3: UserWarning: Parsin
g dates in DD/MM/YYYY format when dayfirst=False (the default) was specified. This may
lead to inconsistently parsed dates! Specify a format to ensure consistent parsing.
  match['season']=pd.DatetimeIndex(match['date']).year
```

Out[35]:

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

In [36]:
```
# Count the number of matches played in each session.
#groupby involves some combination of splitting the object,applying a function,and comb
#This can be used to group large amounts of data and compute operations on thse groups.

match_per_season=match.groupby(['season'])['id'].count()
match_per_season
```

Out[36]:
```
season
2008    58
2009    57
2010    60
2011    73
2012    74
2013    76
2014    60
2015    59
2016    60
2017    59
2018    60
2019    60
Name: id, dtype: int64
```

Merge Two Data : Match and Delivery

In [37]:
```
delivery.rename(columns={'match_id':'id'},inplace=True)
delivery.head(3)
```

Out[37]:

	id	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler	is_super_over	...	bye_
0	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	1	DA Warner	S Dhawan	TS Mills	0	...	
1	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	2	DA Warner	S Dhawan	TS Mills	0	...	
2	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	3	DA Warner	S Dhawan	TS Mills	0	...	

3 rows × 21 columns

In [38]:
```
#merging

season_data=match[['id','season']].merge(delivery,left_on='id',right_on='id',how='left'
season_data.head()
```

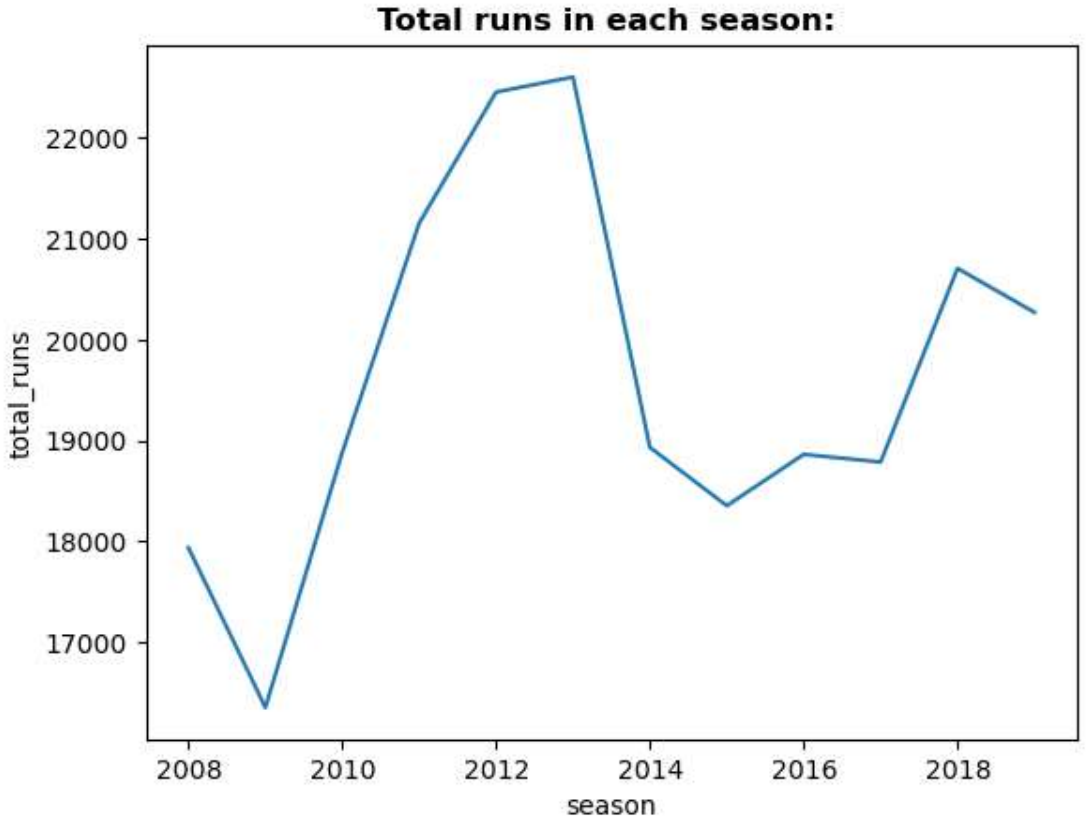
Out[38]:

	season	inning	batting_team	bowling_team	over	ball	batsman	non_striker	bowler	is_super_over	...
0	2017	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	1	DA Warner	S Dhawan	TS Mills	0	...
1	2017	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	2	DA Warner	S Dhawan	TS Mills	0	...
2	2017	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	3	DA Warner	S Dhawan	TS Mills	0	...
3	2017	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	4	DA Warner	S Dhawan	TS Mills	0	...
4	2017	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	5	DA Warner	S Dhawan	TS Mills	0	...

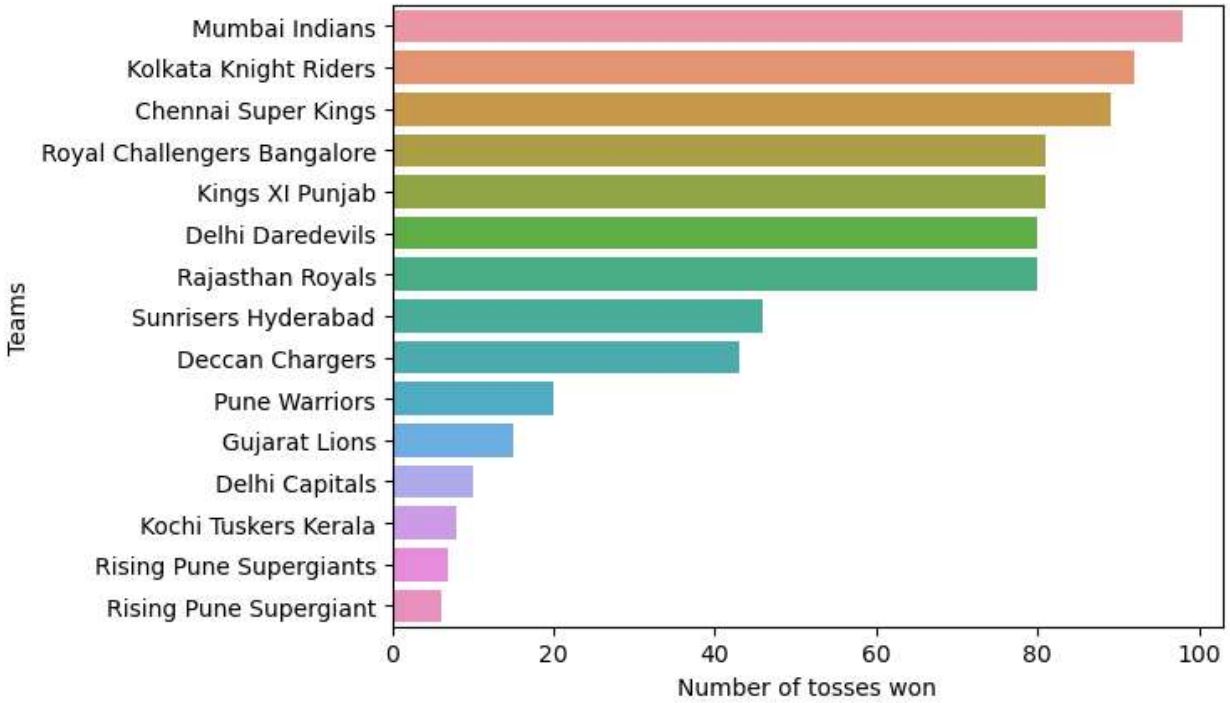
5 rows × 21 columns

Data Visualization

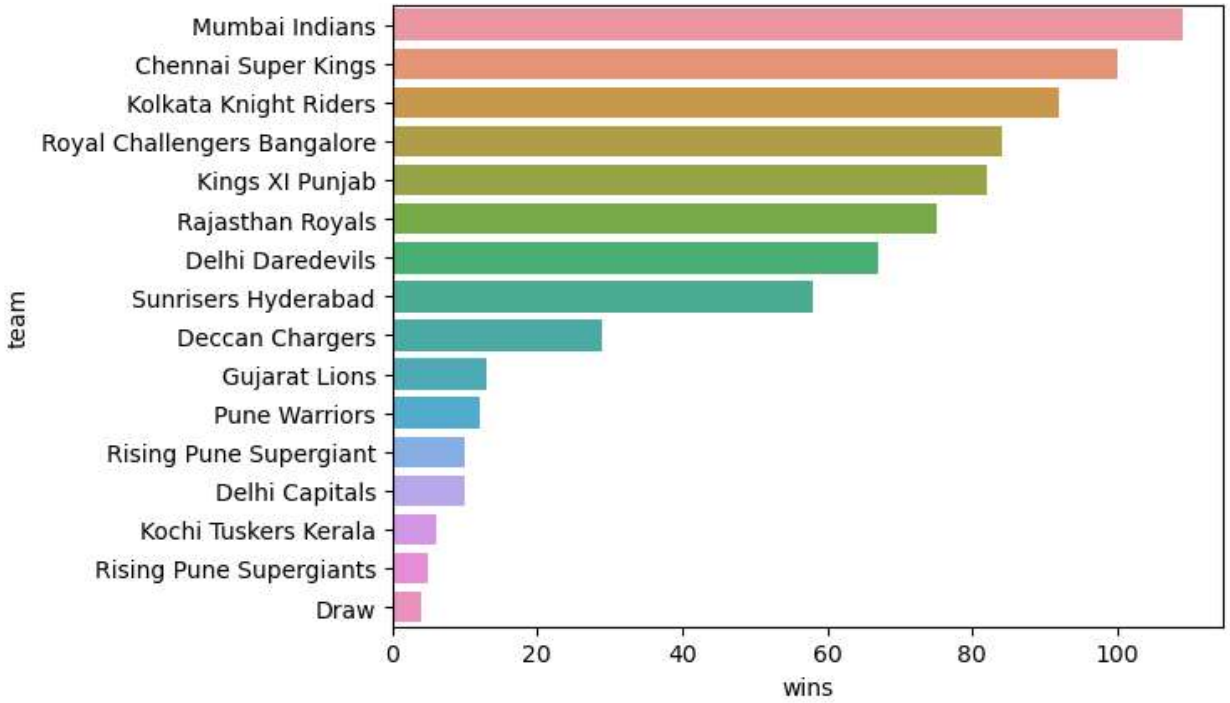
In [39]:
```
#Total Runs Scored in Each Session

season=season_data.groupby(['season'])['total_runs'].sum()
sns.lineplot(data=season)
plt.title("Total runs in each season: ",fontsize=12,fontweight="bold")
plt.show()
```



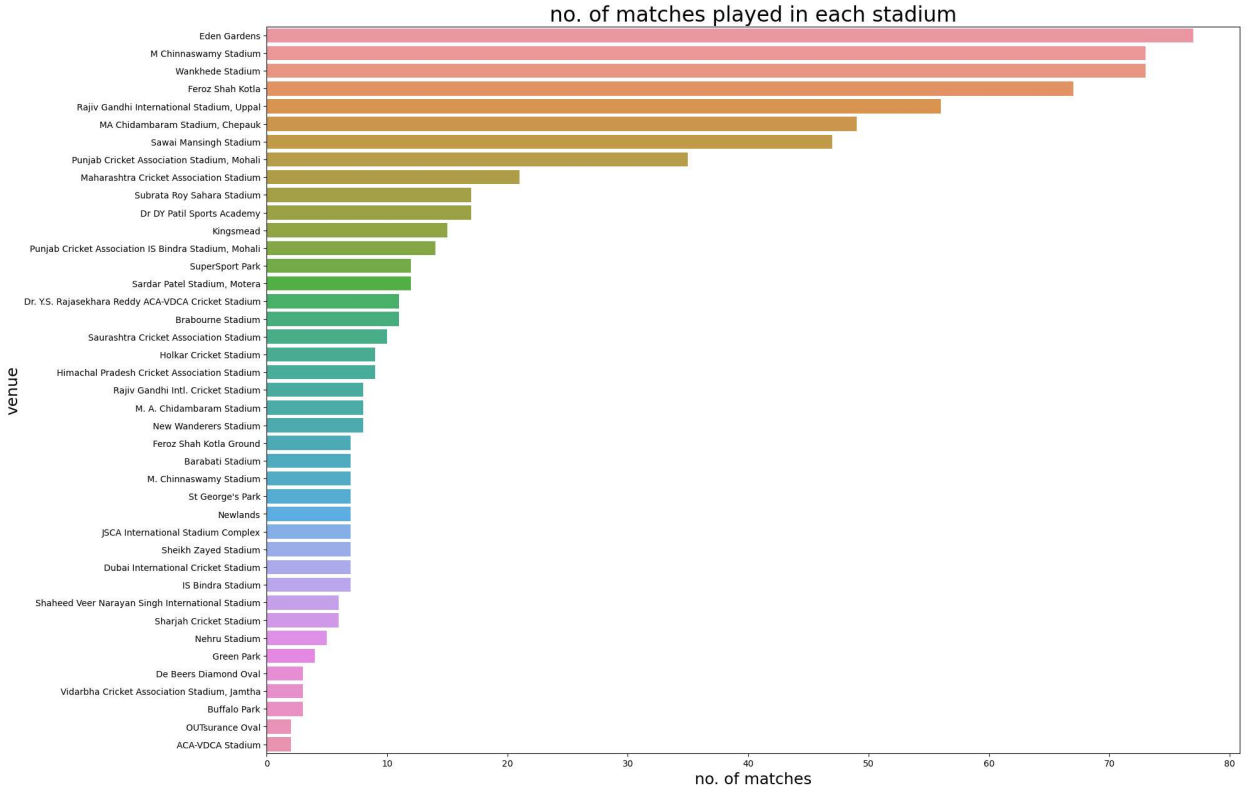
In [40]:
```
# No. of times each team won toss

toss=match["toss_winner"].value_counts()
sns.barplot(y=toss.index,x=toss)
plt.xlabel("Number of tosses won")
plt.ylabel("Teams")
plt.show()
```



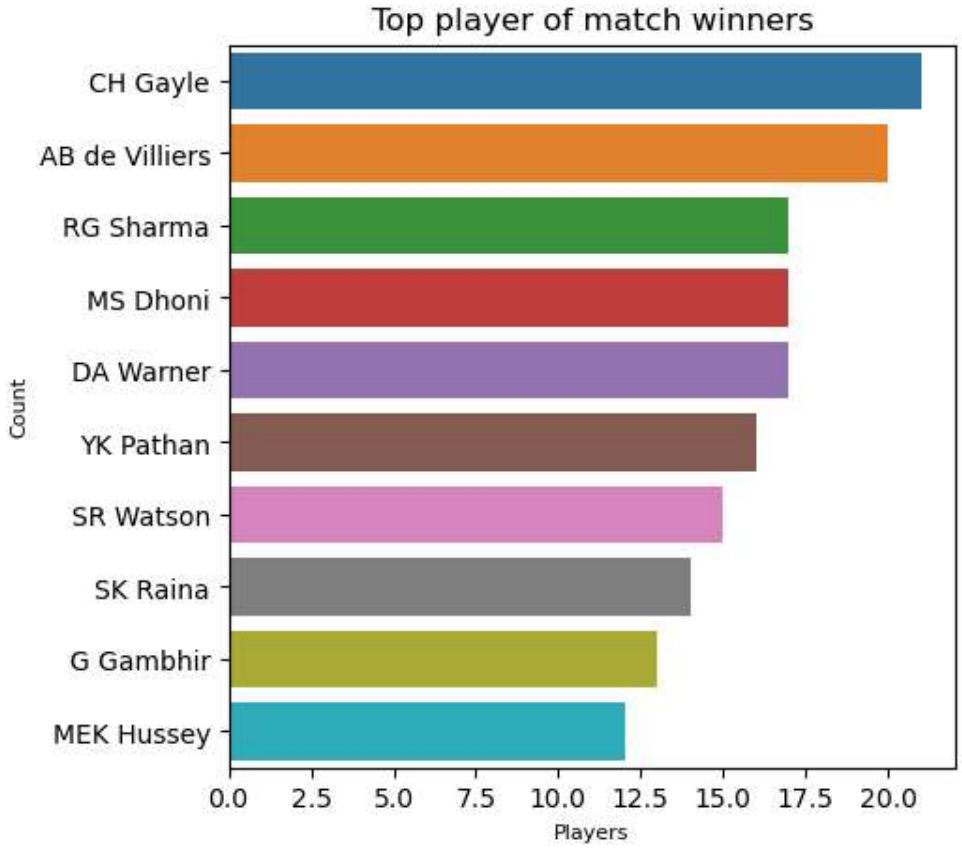
In [41]:
```
#team which won max no of matches

win=match['winner'].value_counts()
sns.barplot(y=win.index,x=win)
plt.xlabel('wins')
plt.ylabel('team')
plt.show()
```



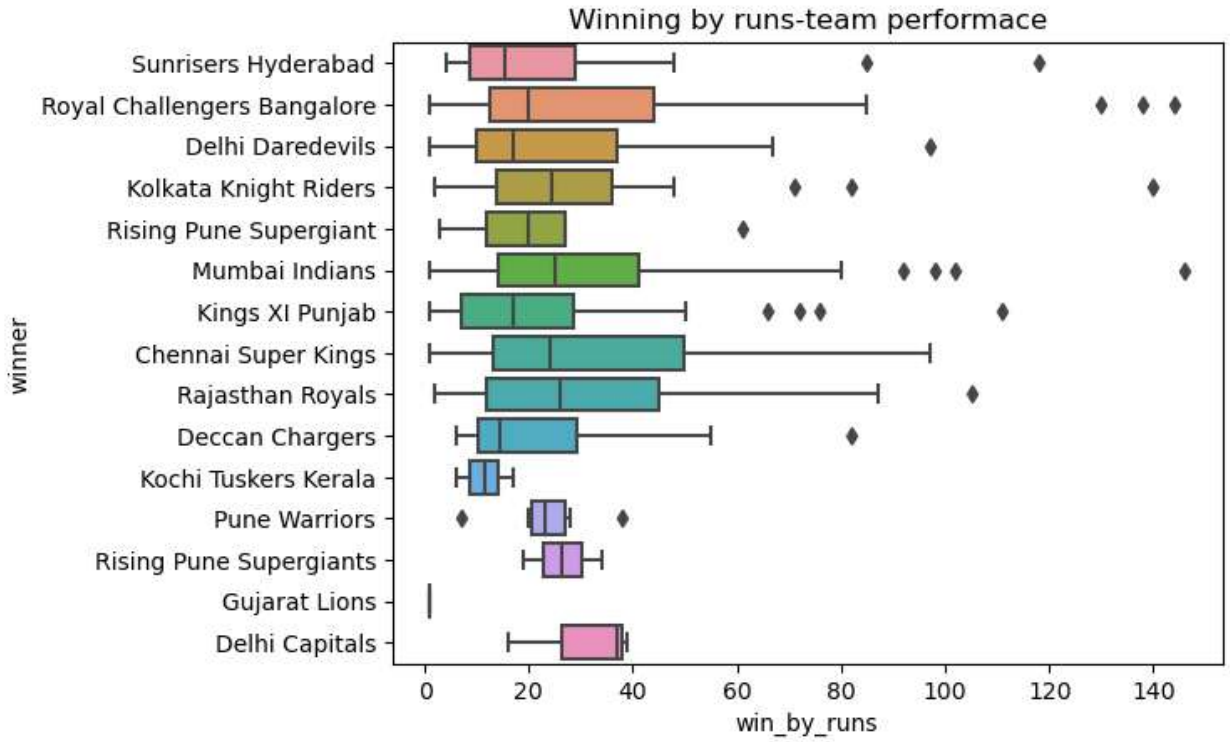
In [42]:
```
#total matches played in each stadium

plt.figure(figsize=(20,15))
stdm=match['venue'].value_counts()
sns.barplot(y=stdm.index,x=stdm)
plt.xlabel("no. of matches",fontsize=18)
plt.ylabel("venue",fontsize=18)
plt.title("no. of matches played in each stadium",fontsize=24)
plt.show()
```



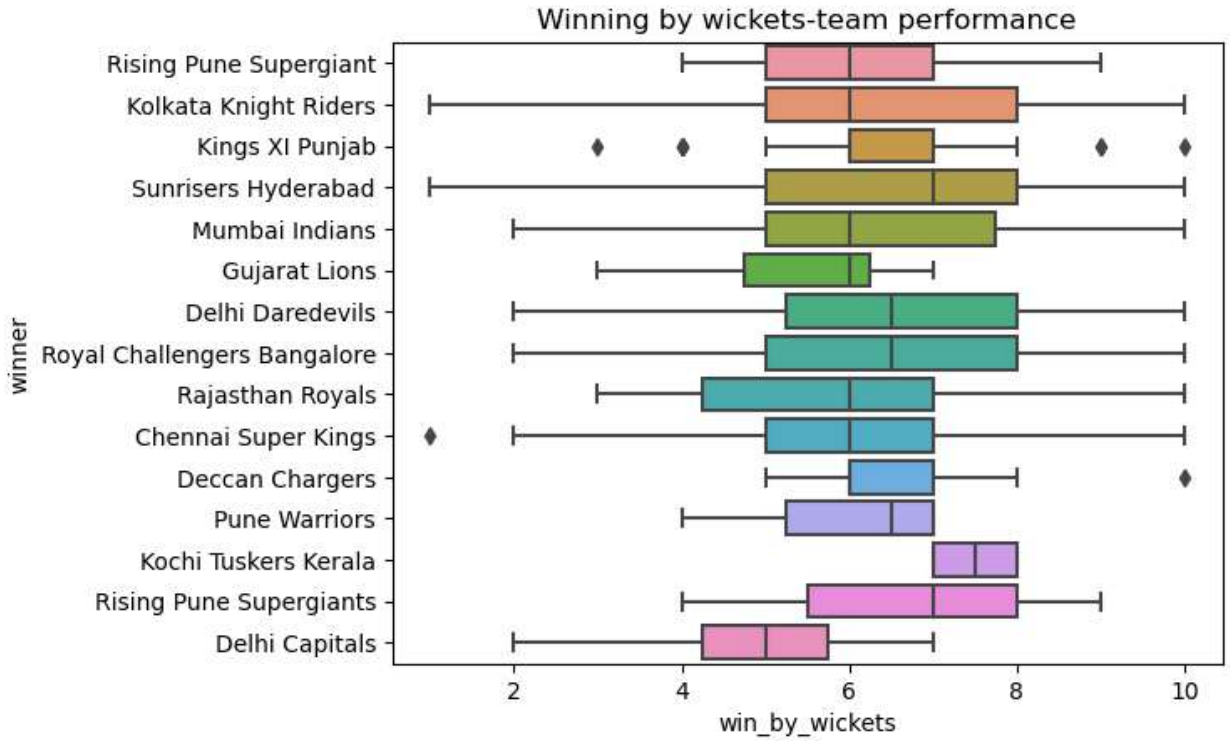
In [43]:
```
plt.figure(figsize=(5,5))
play=match['player_of_match'].value_counts()[:10]
sns.barplot(y=play.index,x=play)
plt.ylabel("Count",fontsize=8)
plt.xlabel("Players",fontsize=8)
plt.title("Top player of match winners",fontsize=12)
plt.show()
```



In [45]:
```
# Winning by runs-team performance.

fig,ax=plt.subplots()
ax.set_title("Winning by runs-team performace")
sns.boxplot(y='winner',x='win_by_runs',data=match[match['win_by_runs']>0],orient='h');
plt.show()
```



In [47]:
```
fig,ax=plt.subplots()
ax.set_title("Winning by wickets-team performance")
sns.boxplot(y='winner', x='win_by_wickets',data=match[match['win_by_wickets']>0],orient
plt.show()
```



Merge Two dataframes : Match and total_score_df

In [48]:
```
match_df=match.merge(total_score_df[['match_id', 'total_runs']], left_on='id', right_on
match_df
```

Out[48]:

	id	Season	city	date	team1	team2	toss_winner	toss_decision	result	dl_a
0	1	IPL-2017	Hyderabad	05-04-2017	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	
1	2	IPL-2017	Pune	06-04-2017	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	
2	3	IPL-2017	Rajkot	07-04-2017	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	
3	4	IPL-2017	Indore	08-04-2017	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	
4	5	IPL-2017	Bangalore	08-04-2017	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	normal	
...	...	...	...	...	...	...	...	...	...	
751	11347	IPL-2019	Mumbai	05-05-2019	Kolkata Knight Riders	Mumbai Indians	Mumbai Indians	field	normal	
752	11412	IPL-2019	Chennai	07-05-2019	Chennai Super Kings	Mumbai Indians	Chennai Super Kings	bat	normal	
753	11413	IPL-2019	Visakhapatnam	08-05-2019	Sunrisers Hyderabad	Delhi Capitals	Delhi Capitals	field	normal	
754	11414	IPL-2019	Visakhapatnam	10-05-2019	Delhi Capitals	Chennai Super Kings	Chennai Super Kings	field	normal	
755	11415	IPL-2019	Hyderabad	12-05-2019	Mumbai Indians	Chennai Super Kings	Mumbai Indians	bat	normal	

756 rows × 20 columns

In [49]:
```
match_df['team1'].unique()
```

Out[49]:
```
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',
       'Delhi Capitals'], dtype=object)
```

In [50]:
```
match_df['team1']=match_df['team1'].str.replace('Delhi Daredevils','Delhi Capitals')
match_df['team2']=match_df['team2'].str.replace('Delhi Daredevils','Delhi Capitals')

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

In [51]:
```
match_df=match_df[match_df['team1'].isin(teams)]
match_df=match_df[match_df['team2'].isin(teams)]
```

In [52]:
```
match_df.shape
```

Out[52]: (641, 20)

In [53]:
```
match_df=match_df[match_df['dl_applied']==0]
```

In [54]:
```
match_df=match_df[['match_id','city','winner','total_runs']]
```

In [55]:
```
print(match_df.columns)
```

```
Index(['match_id', 'city', 'winner', 'total_runs'], dtype='object')
```

In [56]:
```
delivery_df=match_df.merge(delivery, left_on = 'match_id', right_on='id')
```

In [57]:
```
delivery_df=delivery_df[delivery_df['inning']==2]
```

In [58]:
```
delivery_df
```

Out[58]:

	match_id	city	winner	total_runs_x	id	inning	batting_team	bowling_team	over	bal
125	1	Hyderabad	Sunrisers Hyderabad	207	1	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	
126	1	Hyderabad	Sunrisers Hyderabad	207	1	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	
127	1	Hyderabad	Sunrisers Hyderabad	207	1	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	
128	1	Hyderabad	Sunrisers Hyderabad	207	1	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	
129	1	Hyderabad	Sunrisers Hyderabad	207	1	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	
...	...	...	...	...	...	...	...	...	...	...
149573	11415	Hyderabad	Mumbai Indians	152	11415	2	Chennai Super Kings	Mumbai Indians	20	
149574	11415	Hyderabad	Mumbai Indians	152	11415	2	Chennai Super Kings	Mumbai Indians	20	
149575	11415	Hyderabad	Mumbai Indians	152	11415	2	Chennai Super Kings	Mumbai Indians	20	
149576	11415	Hyderabad	Mumbai Indians	152	11415	2	Chennai Super Kings	Mumbai Indians	20	
149577	11415	Hyderabad	Mumbai Indians	152	11415	2	Chennai Super Kings	Mumbai Indians	20	

72413 rows × 25 columns

In [59]:
```
delivery_df['current_score']=delivery_df.groupby('match_id').cumsum()['total_runs_y']
```

```
C:\Users\sharm\AppData\Local\Temp\ipykernel_25660\1213823236.py:1: FutureWarning: The
default value of numeric_only in DataFrameGroupBy.cumsum is deprecated. In a future ve
rsion, numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  delivery_df['current_score']=delivery_df.groupby('match_id').cumsum()['total_runs_
y']
```

In [60]:
```
delivery_df['runs_left'] = delivery_df['total_runs_x'] - delivery_df['current_score']
```

In [61]:
```
delivery_df['balls_left'] = 126 - (delivery_df['over']*6 + delivery_df['ball'])
```

In [62]:
```
delivery_df
```

Out[62]:

	match_id	city	winner	total_runs_x	id	inning	batting_team	bowling_team	over	bal
125	1	Hyderabad	Sunrisers Hyderabad	207	1	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	
126	1	Hyderabad	Sunrisers Hyderabad	207	1	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	
127	1	Hyderabad	Sunrisers Hyderabad	207	1	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	
128	1	Hyderabad	Sunrisers Hyderabad	207	1	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	
129	1	Hyderabad	Sunrisers Hyderabad	207	1	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	
...	...	...	...	...	...	...	...	...	...	
149573	11415	Hyderabad	Mumbai Indians	152	11415	2	Chennai Super Kings	Mumbai Indians	20	
149574	11415	Hyderabad	Mumbai Indians	152	11415	2	Chennai Super Kings	Mumbai Indians	20	
149575	11415	Hyderabad	Mumbai Indians	152	11415	2	Chennai Super Kings	Mumbai Indians	20	
149576	11415	Hyderabad	Mumbai Indians	152	11415	2	Chennai Super Kings	Mumbai Indians	20	
149577	11415	Hyderabad	Mumbai Indians	152	11415	2	Chennai Super Kings	Mumbai Indians	20	

72413 rows × 28 columns

In [63]:
```
#fillna 0 in the column of player_dismissed

delivery_df['player_dismissed'] = delivery_df['player_dismissed'].fillna("0")

#pandas apply lambda function 0 and 1 method

delivery_df['player_dismissed'] = delivery_df['player_dismissed'].apply(lambda x:x if x

#implimenting astype as int

delivery_df['player_dismissed'] = delivery_df['player_dismissed'].astype('int')

#Appling groupby function

wickets = delivery_df.groupby('match_id').cumsum()['player_dismissed'].values
delivery_df['wickets'] = 10 - wickets
```

```
C:\Users\sharm\AppData\Local\Temp\ipykernel_25660\879185360.py:15: FutureWarning: The
default value of numeric_only in DataFrameGroupBy.cumsum is deprecated. In a future ve
rsion, numeric_only will default to False. Either specify numeric_only or select only
columns which should be valid for the function.
  wickets = delivery_df.groupby('match_id').cumsum()['player_dismissed'].values
```

In [64]:
```
#To see the first rows of the dataset

delivery_df.head()
```

Out[64]:

	match_id	city	winner	total_runs_x	id	inning	batting_team	bowling_team	over	ball	...	l
125	1	Hyderabad	Sunrisers Hyderabad	207	1	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	1	...	
126	1	Hyderabad	Sunrisers Hyderabad	207	1	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	2	...	
127	1	Hyderabad	Sunrisers Hyderabad	207	1	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	3	...	
128	1	Hyderabad	Sunrisers Hyderabad	207	1	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	4	...	
129	1	Hyderabad	Sunrisers Hyderabad	207	1	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	5	...	

5 rows × 29 columns

In [65]:
```
# current run rate = runs/overs

delivery_df['crr'] = (delivery_df['current_score']*6)/(120 - delivery_df['balls_left'])
```

In [66]:
```
delivery_df['rrr'] = (delivery_df['runs_left']*6)/delivery_df['balls_left']
```

In [67]:
```
def result(row):
    return 1 if row['batting_team'] == row['winner'] else 0
```

In [68]:
```
delivery_df['result'] = delivery_df.apply(result,axis=1)
```

In [69]:
```
final_df = delivery_df[['batting_team','bowling_team','city','runs_left','balls_left','
```

In [70]:
```
#Chcking the shape of the final_df

final_df = final_df.sample(final_df.shape[0])
```

In [71]:
```
#To check the sample

final_df.sample()
```

Out[71]:

	batting_team	bowling_team	city	runs_left	balls_left	wickets	total_runs_x	crr	rrr	resul
19041	Kolkata Knight Riders	Kings XI Punjab	Kolkata	-1	2	3	174	8.898305	-3.0	1

In [72]:
```
final_df.dropna(inplace=True)
```

In [73]:
```
final_df = final_df[final_df['balls_left'] != 0]
```

In [74]:
```
# Dimensionality reduciton by the index value X and y

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

In [75]:
```
#Spliting the value into training and testing dataset

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

In [76]:
```
X_train
```

Out[76]:

	batting_team	bowling_team	city	runs_left	balls_left	wickets	total_runs_x	crr	
58992	Kings XI Punjab	Rajasthan Royals	Jaipur	55	14	3	191	7.698113	23.5
48048	Kolkata Knight Riders	Chennai Super Kings	Chennai	32	17	5	153	7.048544	11.2
122200	Kolkata Knight Riders	Royal Challengers Bangalore	Kolkata	15	18	5	182	9.823529	5.0
79732	Kings XI Punjab	Mumbai Indians	Mumbai	105	71	7	174	8.448980	8.8
33065	Royal Challengers Bangalore	Chennai Super Kings	Johannesburg	18	17	6	146	7.456311	6.3
...	...	...	...	...	...	...	...	...	
109158	Royal Challengers Bangalore	Chennai Super Kings	Chennai	114	90	8	148	6.800000	7.6
77834	Delhi Daredevils	Kings XI Punjab	Dharamsala	60	23	5	171	6.865979	15.6
61847	Royal Challengers Bangalore	Rajasthan Royals	Bangalore	116	51	6	195	6.869565	13.6
139108	Sunrisers Hyderabad	Delhi Capitals	Delhi	54	68	8	134	9.230769	4.7
84199	Delhi Daredevils	Chennai Super Kings	Chennai	134	94	9	168	7.846154	8.5

57737 rows × 9 columns

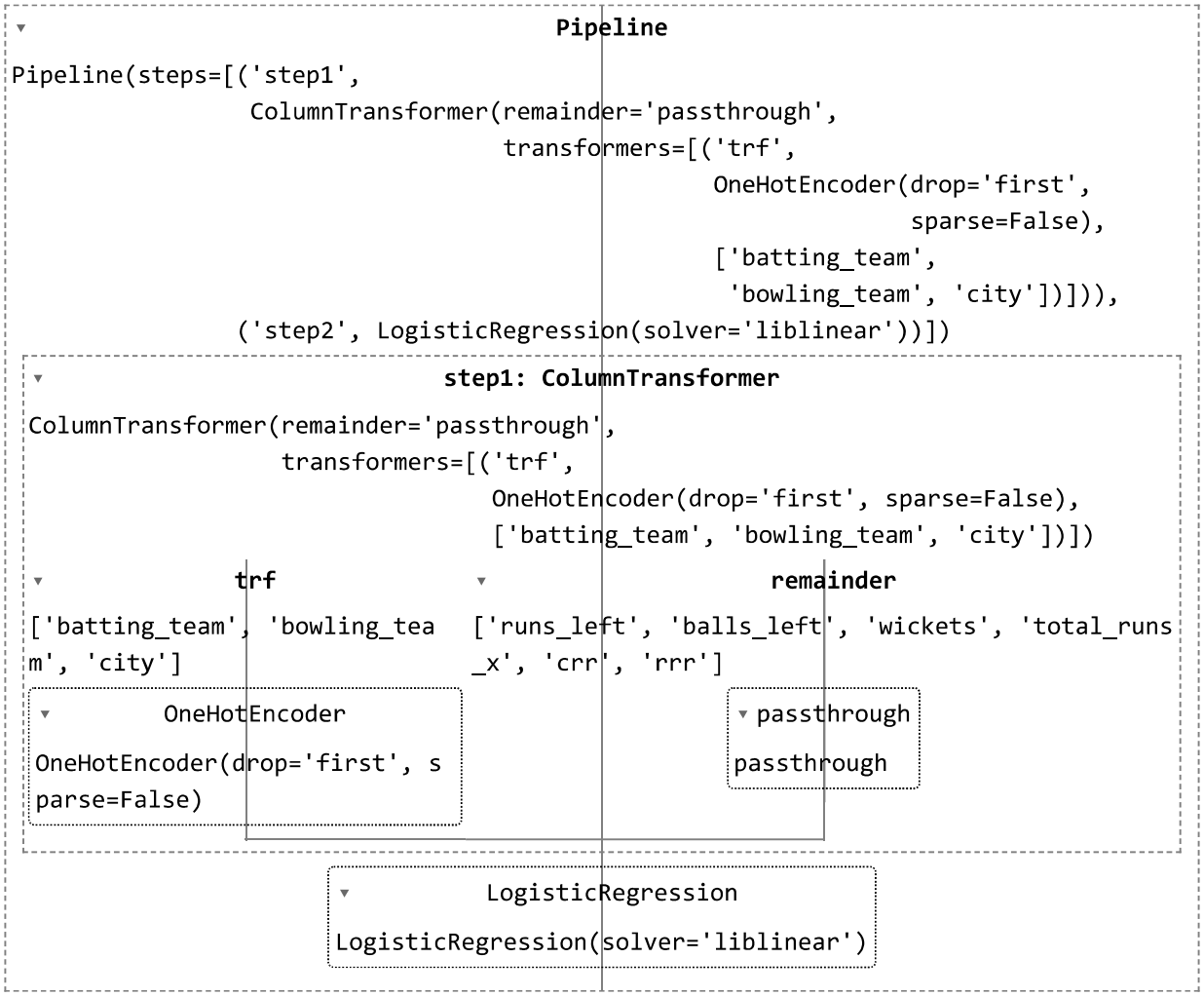
In [77]:
```
# importing column transformer and onehotencoder from sklearn

from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

trf = ColumnTransformer([
    ('trf',OneHotEncoder(sparse=False,drop='first'),['batting_team','bowling_team','city
]
,remainder='passthrough')
```

In [78]:
```
pipe = Pipeline(steps=[
    ('step1',trf),
    ('step2',LogisticRegression(solver='liblinear'))
])
```

In [79]:
```
pipe.fit(X_train,y_train)
```

```
C:\Users\sharm\anaconda3\Lib\site-packages\sklearn\preprocessing\_encoders.py:972: Fut
ureWarning: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed
in 1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

Out[79]:



In [80]:
```
#applying prediction method

y_pred = pipe.predict(X_test)   #prediction variable
```

In [81]:
```
#test and prediction accuracy

accuracy_score(y_test,y_pred)
```

Out[81]:
```
0.798198822306893
```

In [82]:
```
pipe.predict_proba(X_test)[10]
```

Out[82]:
```
array([0.97909424, 0.02090576])
```

Match Progression Analysis and Win-Loss Prediction.

In [83]:
```
def match_summary(row):
    print("Batting Team-" + row['batting_team'] + " | Bowling Team-" + row['bowling_tea
```

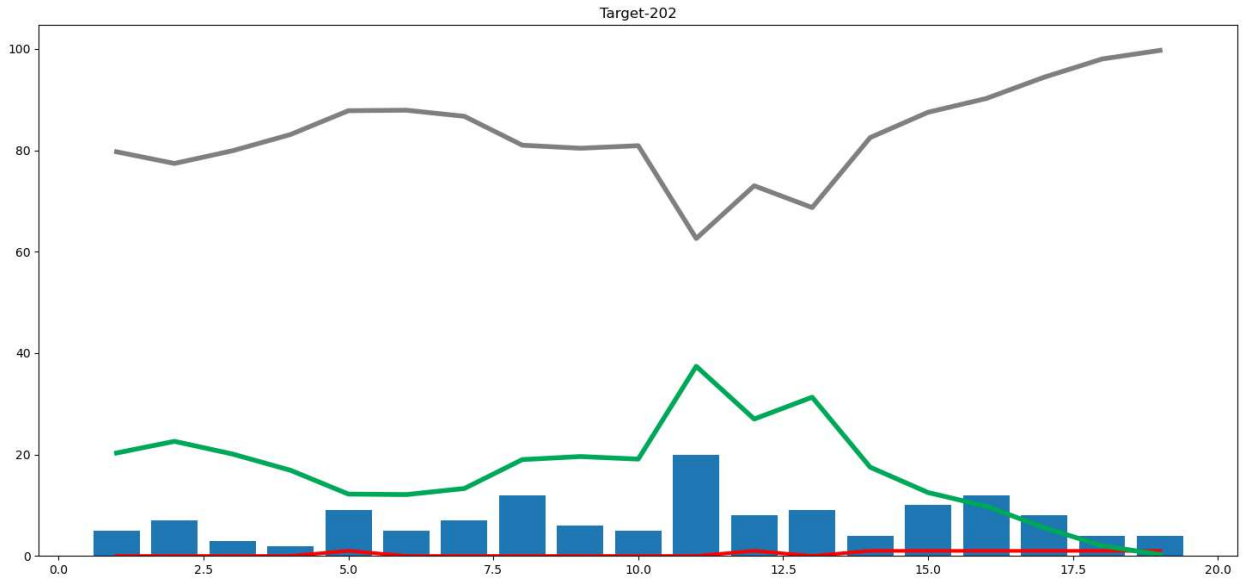
In [84]:
```
import pandas as pd
import numpy as np
import pickle

def match_progression(x_df, pipe):
    match_id = int(input("Please enter the match_id: "))  # Take match_id as input from
    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','wic
    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'].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','win']]
    return temp_df, target


temp_df, target = match_progression(delivery_df, pipe)
print(temp_df)
```

```
Please enter the match_id: 576
Target- 202
        end_of_over  runs_after_over  wickets_in_over  lose   win
113963            1                5                0  79.7  20.3
113969            2                7                0  77.4  22.6
113975            3                3                0  79.9  20.1
113981            4                2                0  83.1  16.9
113988            5                9                1  87.8  12.2
113994            6                5                0  87.9  12.1
114000            7                7                0  86.7  13.3
114006            8               12                0  81.0  19.0
114012            9                6                0  80.4  19.6
114019           10                5                0  80.9  19.1
114026           11               20                0  62.6  37.4
114032           12                8                1  73.0  27.0
114038           13                9                0  68.7  31.3
114044           14                4                1  82.5  17.5
114050           15               10                1  87.5  12.5
114056           16               12                1  90.2   9.8
114063           17                8                1  94.4   5.6
114069           18                4                1  98.0   2.0
114076           19                4                1  99.7   0.3
```

Visualizing Match Progression and Win-Loss Predictions.

In [85]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(18,8))
plt.plot(temp_df['end_of_over'],temp_df['wickets_in_over'],color='red',linewidth=3)
plt.plot(temp_df['end_of_over'],temp_df['win'],color='#00a85a',linewidth=4)
plt.plot(temp_df['end_of_over'],temp_df['lose'],color='grey',linewidth=4)
plt.bar(temp_df['end_of_over'],temp_df['runs_after_over'])
plt.title('Target-' + str(target))
```

Out[85]: Text(0.5, 1.0, 'Target-202')



IPL Match Outcome Prediction.

In [87]:
```
teams = ['Sunrisers Hyderabad', 'Mumbai Indians', 'Royal Challengers Bangalore', 'Kolka
cities = ['Hyderabad', 'Bangalore', 'Mumbai', 'Indore', 'Kolkata', 'Delhi', 'Chandigarh

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

print('IPL Win Predictor')

batting_team = input('Select the batting team: ')
bowling_team = input('Select the bowling team: ')
selected_city = input('Select host city: ')
target = int(input('Target: '))
score = int(input('Score: '))
overs = float(input('Overs completed: '))
wickets = int(input('Wickets out: '))

runs_left = target - score
balls_left = 120 - (overs*6)
wickets = 10 - wickets
crr = score/overs
rrr = (runs_left*6)/balls_left

input_df = pd.DataFrame({'batting_team':[batting_team],'bowling_team':[bowling_team],'c

result = pipe.predict_proba(input_df)
loss = result[0][0]
win = result[0][1]

print(batting_team + "- " + str(round(win*100)) + "%")
print(bowling_team + "- " + str(round(loss*100)) + "%")
```

```
IPL Win Predictor
Select the batting team: Delhi Capitals
Select the bowling team: Chennai Super Kings
Select host city: Indore
Target: 260
Score: 140
Overs completed: 12
Wickets out: 4
Delhi Capitals- 54%
Chennai Super Kings- 46%
```

Conclusion

1. The project showcases the application of machine learning in sports analytics, specifically for predicting outcomes of IPL cricket matches.

2. It provides real-time, dynamic predictions that enhance the cricket-watching experience for fans.

3. The project not only offers predictions but also visualizes match progression in an engaging and understandable manner.

4. The interactive nature of the project allows users to input match details and receive personalized predictions, making the tool user-friendly.

5. This project highlights the transformative potential of data science and machine learning in revolutionizing traditional sports viewing experiences.

6. There is great potential for future enhancements and refinements to provide even more insightful predictions and a more immersive viewing experience.

7. The project aims to cater to cricket fans, data enthusiasts, or both, offering an intriguing glimpse into the future of sports entertainment. Enjoy the game!