IPL ANALYSIS (2008-2017)

20MIA1066 ---- PIYALI SAHA, Vellore Institute of Technology, Chennai, India

20MIA1147 ---- SHASHANK PANDEY, Vellore Institute of Technology, Chennai, India

ABSTRACT

Cricket is one of the famous outdoor sports that contain a large set of statistical data in real world. As IPL games rise in popularity, it is necessary to examine the possible predictors that affect the outcome of the matches. The project aims at analysing the IPL cricket match results from the dataset collected (2008-2017) . Analysis of structured data has seen tremendous success in the past. However, analysis of large scale unstructured data to perform predictions remains a challenging area. The Indian Premier League (IPL) is a T20 cricket league tournament held in India contested during April and May of every year where top players from all over the world take part. The IPL is the most-attended cricket league in the world and ranks sixth among all sports leagues. The idea is to analyse the IPL data hosted by Kaggle to come up with something interesting and useful. We have used various graphs and plots for doing this analysis. The project utilizes the IPL datasets that allows analyst to incorporate functions that are used by IPL application to fetch and view information. This project uses python, Python packages are: numpy, pandas, matplotlib, and seaborn and tableau, K-means clustering to extract the meaningful output which can be used by the management for analysis.

KEYWORD

Analysis , prediction , accuracy , data visualisation, graphs , dashboards, K-means clustering .

INTRODUCTION

The Indian Premier League, created by the Board of Control for Cricket in India (BCCI) on September 14, 2007 and sanctioned by the International Cricket Council (ICC), is a Twenty20 cricket competition. Indian premier league is considered to be One of the finest 20-20 game in the world of Cricket. Based on the lines of the English Premier League (EPL) and the National Basketball League (NBA), the IPL is

said to be the brainchild of BCCI vice-president Lalit Modi and is modelled along the lines of club football in Europe, which is unlike anything cricket has known in the past. The idea was first floated in 1996 but was shot down as the board felt it would go against the zonal system of domestic cricket. The project moved into top gear when the Zee group launched a rival Indian Cricket League on similar lines in April 2007. The best players from around the world will not play according to their nationality but as per the market forces. Later some politicians and prominent individuals found the scope of this process and initiated IPL. The 46-year-old Modi created the IPL, the most lucrative event in the history of Indian entertainment. The Board earned a profit of around Rs 300 Crore each of the three seasons of the IPL. But it turned out to be a Pyrrhic victory. It has very valuable 10 teams and has taken Indian cricket to a very high level. Billions of dollars being transacted in this event and every cricket lover love this event. Lots of money is involved in IPL with big corporate investing in this product. Lalit Modi has done a great job to bring IPL to this level. In 2008, Lalit Modi was instrumental in launching the Indian Premier League (IPL), a league based around Twenty20 cricket, where each team is limited to batting for a maximum of 20 overs. He also engineered the Indian Premier League's move to South Africa in 2009 after the dates of the tournament clashed with the Indian general election and the Union Minister of Home Affairs, P. Chidambaram, could not commit to the security of the tournament. The IPL has since grown into one of the world's biggest sports, worth over US$4 billion. The commercial success of the Indian Premier League and Modi's control of the league has led to him being compared to Don King and Bernie Ecclestone.

LITERATURE SURVEY

Parag shah [1] in this describes about significant challenges that we face for accurate prediction including the various parameters which affect the outcome of the match. The ball movement gets changed from every over, so it is considered being important to predicting the outcome of each match on every ball. Here they had developed a model that predicts the match result of every ball played. Using Duckworth-Lewis formula the outcome of the match will be predicted for live match. Probability is calculated and figure is plotted for each ball bowled .This model and the probability figure will be very useful for betting industry to decide which team will won the match. Using Par score concept given by Duckworth &Lewis, probability has been calculated by considering the balls faced, balls left, runs scored, runs left, wicket, wickets left.

H.Ahmad [2] in paper explains about the concept of identifying rising stars in cricket domain by using different techniques. Rising stars can be predicted by both bats as well as bowling teams. Distinct features like concept of co-players, team and opposite teams are presented with their mathematical formulation. High accuracy is demonstrated for both robust and statically significant cases. At last the top ranking list of ten rising crickets is compared with International cricket council ranking based on weighted average, performance, evolution and the rising stars scores. Measures are explicitly adopted for rising star prediction in bat and bowling domains. Finally, ranking lists of

rising stars based on weighted average, performance evolution and rising star score are presented both domain.

FEASIBILITY STUDY

The analysis parameters for IPL Analysis could be of any form: score, batsman, bowler, venue related to the matches .Data Science , data visualisation, analysis is all about finding valuable information from the given dataset. Using this data, we tried to find out the following information from IPL DataFrame.

1. How many matches have been played in each season?
2. Who won the maximum matches?
3. Who won the match by maximum runs?
4. Match runs by the maximum wickets
5. The Most valuable player in all the IPL season?
6. Match Won by minimum wickets?
7. The most successful team in all the season?

ABOUT THE DATASET

We have used two IPL datasets :-

- Matches dataset - https://www.kaggle.com/josephgpinto/ipl-data-analysis/data?select=matches.csv
- Deliveries dataset – https://www.kaggle.com/josephgpinto/ipl-data-analysis/data

  In Matches dataset we have season(numerical), city(categorical), date(numerical), teams (categorical), toss winner(categorical), toss decision(categorical), result (categorical), winner(categorical), win by runs and wickets(numerical), player of match(categorical), venue(categorical)
  And in deliveries dataset we have match id (numerical), inning , ball , over , super over, wide runs, no ball run, penalty run, batsman run, extra run, total run and wicket(numerical), batting and bowling team(categorical), batsman and bowler (categorical), non striker and player dismissed (categorical) and dismissal kind (categorical)

FEATURE COMPONENTS

K-means

Kmeans algorithm is an iterative algorithm that tries to partition the dataset into K pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible.

The way kmeans algorithm works is as follows:

1. Specify number of clusters K.

2. Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.

3. Keep iterating until there is no change to the centroids.

4. Compute the sum of the squared distance between data points and all centroids.

5. Assign each data point to the closest cluster (centroid).

IMPLEMENTATION

MATCHES DATASET

```
print (df.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
None
```

```
df.describe()
```

	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	3.373642	NaN
std	183.741666	2.773026	0.156725	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

Total Matches played at each Venue

```
df["city"].value_counts()
#result : The maximum number of matches played in a particular city is mumbai
#with 85 matches overall.
```

```
Mumbai            65
Bangalore         64
Kolkata           61
Delhi             60
Hyderabad         49
Chennai           48
Chandigarh        44
Jaipur            33
Pune              32
Durban            15
Centurion         12
Ahmedabad         12
Visakhapatnam     11
Rajkot            10
Dharamsala         9
Johannesburg       8
Ranchi             7
Cape Town          7
Port Elizabeth     7
Abu Dhabi          7
Cuttack            7
Sharjah            6
Raipur             6
Indore             5
Kochi              5
Kanpur             4
East London        3
Kimberley          3
Nagpur             3
Bloemfontein       2
Name: city, dtype: int64
```

How many seasons in the dataset?

```
df['season'].unique()
```

```
array([2017, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016],
      dtype=int64)
```

Toss Decision

```
df['toss_decision'].value_counts()
#result :we can say that teams prefer to field first rather than batting first i.e prefer to ch
```

```
field    363
bat      273
Name: toss_decision, dtype: int64
```

Which Team had won by maximum runs?

```
df.iloc[df['win_by_runs'].idxmax()]
#Result : MI has won by 146 runs
```

```
id                              44
season                        2017
city                         Delhi
date                       5/6/2017
team1               Mumbai Indians
team2            Delhi Daredevils
toss_winner      Delhi Daredevils
toss_decision              field
result                    normal
dl_applied                     0
winner              Mumbai Indians
win_by_runs                  146
win_by_wickets                 0
player_of_match      LMP Simmons
venue            Feroz Shah Kotla
umpire1              Nitin Menon
umpire2               CK Nandan
umpire3                     NaN
Name: 43, dtype: object
```

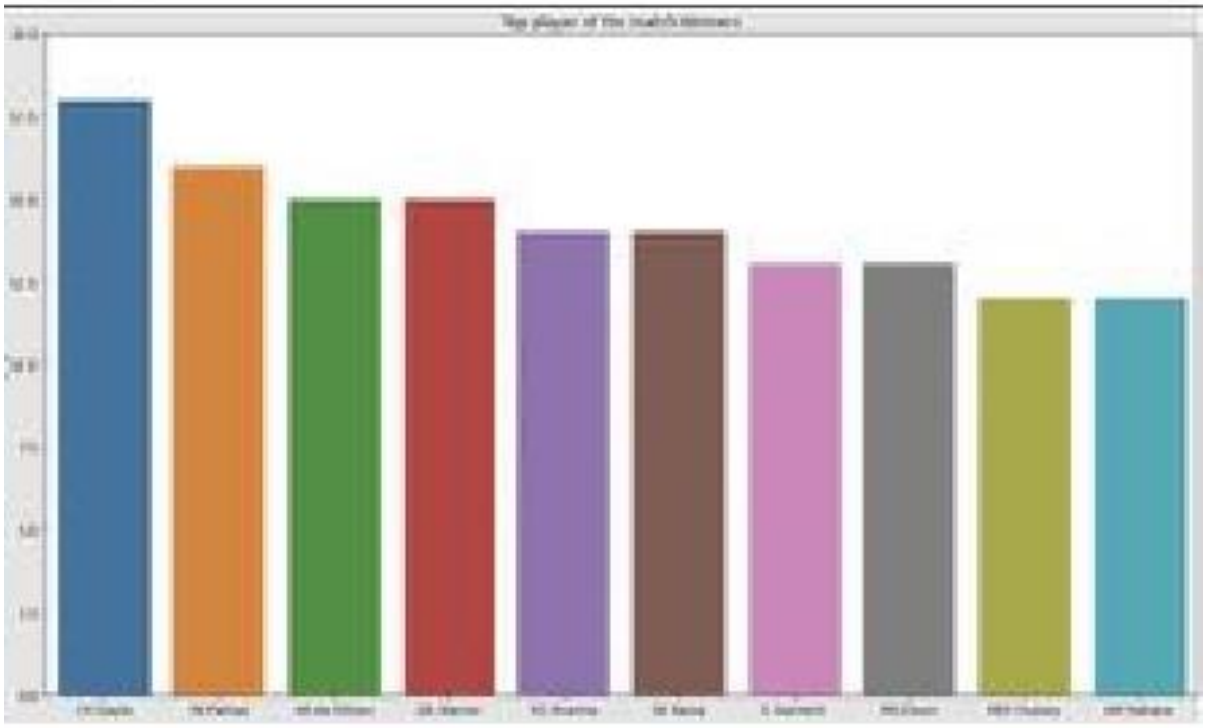
Which Team had won by maximum wickets?

```
df.iloc[df['win_by_wickets'].idxmax()]
#Result : KXP won by 10 wickets
```

```
id                                                3
season                                         2017
city                                         Rajkot
date                                       4/7/2017
team1                                Gujarat Lions
team2                        Kolkata Knight Riders
toss_winner                  Kolkata Knight Riders
toss_decision                               field
result                                     normal
dl_applied                                      0
winner                       Kolkata Knight Riders
win_by_runs                                     0
win_by_wickets                                 10
player_of_match                         CA Lynn
venue            Saurashtra Cricket Association Stadium
umpire1                                Nitin Menon
umpire2                                 CK Nandan
umpire3                                       NaN
Name: 2, dtype: object
```

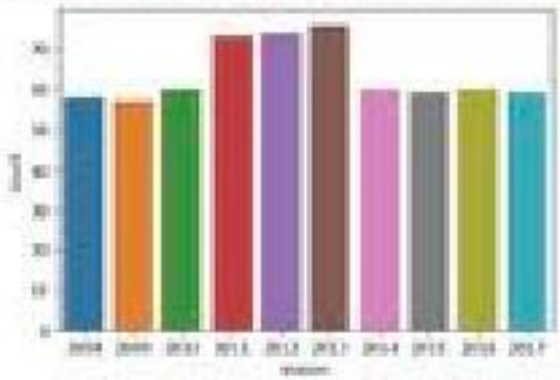
Top players of winning team

```
top_players = df.player_of_match.value_counts()[:10]
fig, ax = plt.subplots(figsize=(17,10))
ax.set_ylim([0,20])
ax.set_ylabel("Count")
ax.set_title("Top player of the match Winners")
#top_players.plot.bar()
sns.barplot(x = top_players.index, y = top_players, orient='v');
plt.show()
```



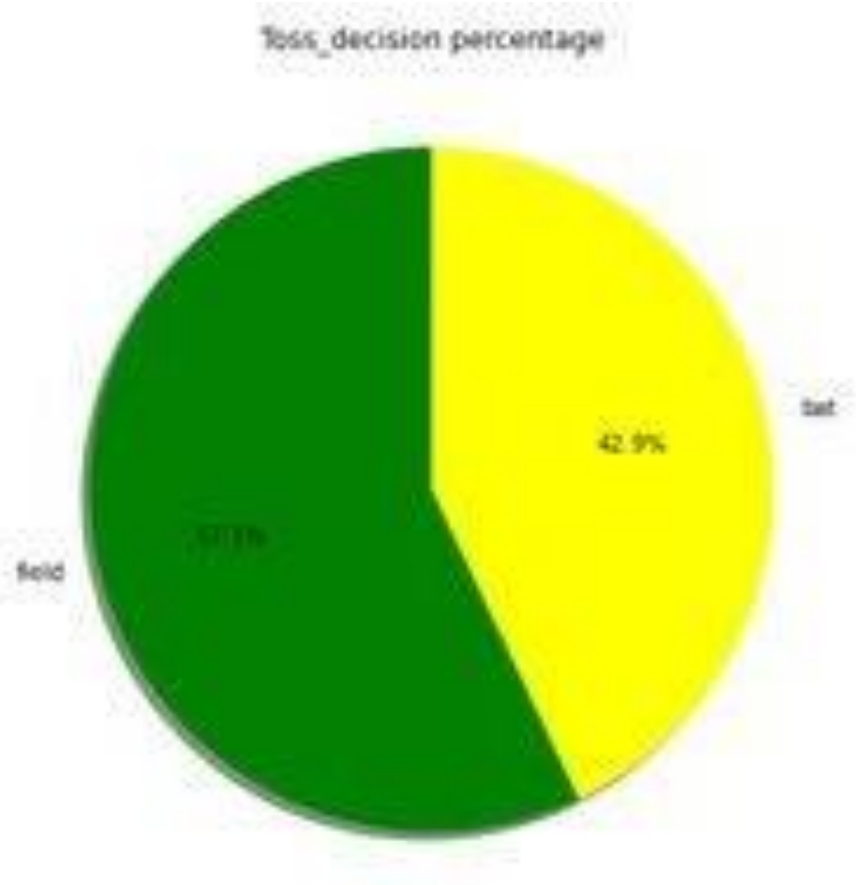
Which season had most number of matches?

```
sns.countplot(x='season', data=df)
plt.show()
#Season 2013 had the most number of matches
```



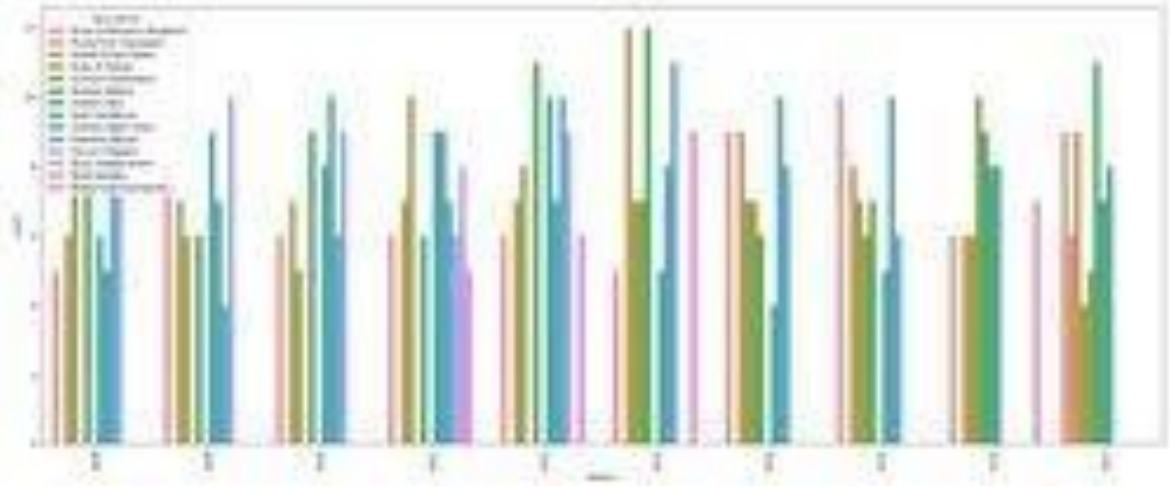
Toss_decision

```
plt.figure(figsize=(18,7))
temp_series = df.toss_decision.value_counts()
labels = (np.array(temp_series.index))
sizes = (np.array((temp_series / temp_series.sum())*100))
colors = ['green', 'yellow']
plt.pie(sizes, labels=labels, colors=colors,
        autopct='%1.1f%%', shadow=True, startangle=90)
plt.title("Toss_decision percentage")
plt.show()
```



per season toss_winner

```
plt.figure(figsize=(15,10))
sns.countplot(x='season', hue='toss_winner', data=df)
plt.xticks(rotation='vertical')
plt.show()
```



The team with the most number of wins per season.

```
teamsper_season = df.groupby('season')['winner'].value_counts()
teamsper_season
```

```
season  winner
2008    Rajasthan Royals                13
        Kings XI Punjab                 10
        Chennai Super Kings              9
        Delhi Daredevils                 7
        Mumbai Indians                   7
                                        ..
2017    Sunrisers Hyderabad              8
        Kings XI Punjab                  7
        Delhi Daredevils                 6
        Gujarat Lions                    4
        Royal Challengers Bangalore      3
Name: winner, Length: 84, dtype: int64
```

```
winnerper_season_df
```

	year	team	wins
0	2008	Rajasthan Royals	13
0	2009	Delhi Daredevils	10
0	2010	Mumbai Indians	11
0	2011	Chennai Super Kings	11
0	2012	Kolkata Knight Riders	12
0	2013	Mumbai Indians	13
0	2014	Kings XI Punjab	12
0	2015	Chennai Super Kings	10
0	2016	Sunrisers Hyderabad	11
0	2017	Mumbai Indians	12

```
plt.figure(figsize=(15,10))
sns.barplot('wins', 'team', hue='year', data=winnerpar_season_df, palette='Paired');
plt.show()

#Mumbai Indians has secured the most wins in four seasons(2020, 2013, 2015, and 2019).
```

```
C:\Cherry\CSEEK\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the
following variables as keyword args: x, y. From version 0.12, the only valid positional argum
ent will be `data`, and passing other arguments without an explicit keyword will result in an
error or misinterpretation.
  warnings.warn(
```

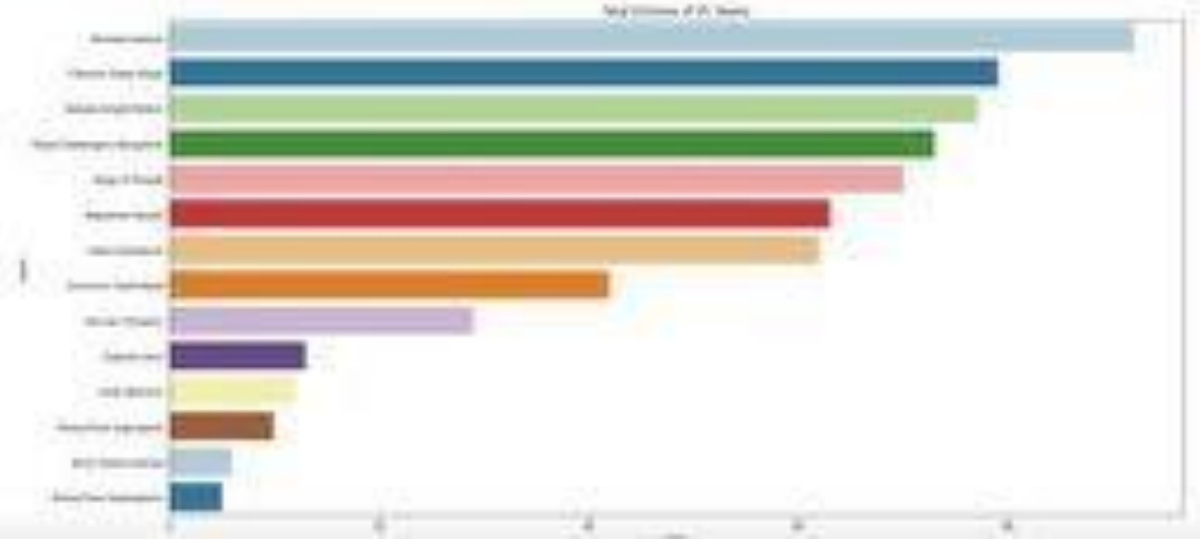
The most successful IPL team

```
team_wins_ser = df['winner'].value_counts()

team_wins_df = pd.DataFrame(columns=["team", "wins"])
for items in team_wins_ser.iteritems():
    temp_df1 = pd.DataFrame({
        'team':[items[0]],
        'wins':[items[1]]
    })
    team_wins_df = team_wins_df.append(temp_df1, ignore_index=True)
```

```
team_wins_df
```

	team	wins
0	Mumbai Indians	92
1	Chennai Super Kings	79
2	Kolkata Knight Riders	77
3	Royal Challengers Bangalore	73
4	Kings XI Punjab	70
5	Rajasthan Royals	63
6	Delhi Daredevils	62
7	Sunrisers Hyderabad	42
8	Deccan Chargers	29
9	Gujarat Lions	13
10	Pune Warriors	12
11	Rising Pune Supergiant	10
12	Kochi Tuskers Kerala	6
13	Rising Pune Supergiants	5

```
plt.figure(figsize=(20,15))
plt.title('Total Victories of IPL Teams')
sns.barplot(x='wins', y='team', data=team_wins_df, palette='Paired')
plt.show()

Mumbai Indians is the most successful team(in this term) with the maximum number of 92 matches (+13%)
Followed by Chennai Super Kings and Kolkata Knight Riders.
```



DELIVERIES DATASET

```
d_df=pd.read_csv("deliveries.csv")
```

```
d_df.head()
```

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

5 rows × 23 columns

```
print(d_df.info())
```

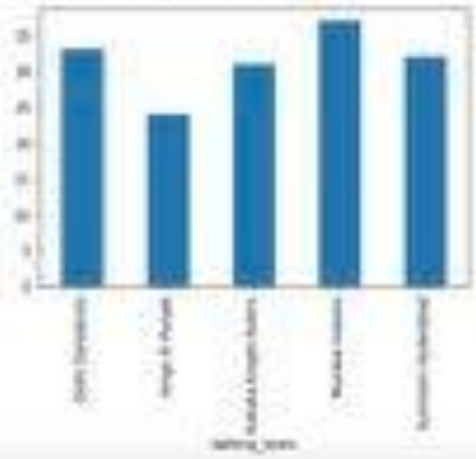
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150460 entries, 0 to 150459
Data columns (total 21 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   match_id          150460 non-null  int64
 1   inning            150460 non-null  int64
 2   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  id_wicket         150460 non-null  int64
 19  id_wicket_kind    150460 non-null  bool
 20  player_dismissed  7438 non-null    object
 21  dismissal_kind    7438 non-null    object
 22  fielder           5369 non-null    object
dtypes: bool(1), int64(14), object(6)
memory usage: 23.4+ MB
None
```

```
d_df.describe()
```

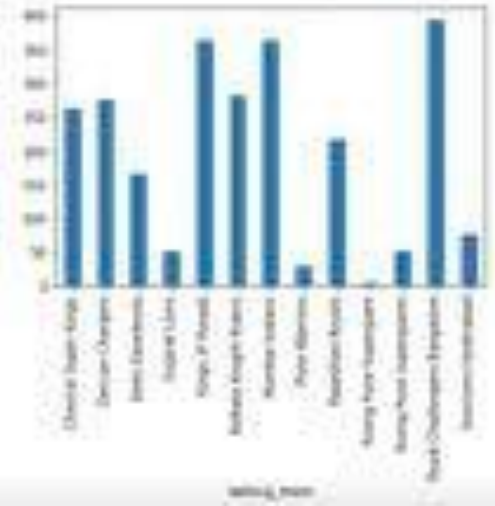
	match_id	inning	over	ball	is_super_over	wide_runs	bye_runs	legbye_runs	noi
count	150460.000000	150460.000000	150460.000000	150460.000000	150460.000000	150460.000000	150460.000000	150460.000000	15046
mean	318.281317	1.482188	10.142649	3.616483	0.000538	0.037498	0.004885	0.022232	1
std	182.955531	0.501768	5.674338	1.807698	0.023196	0.257398	0.114234	0.200104	
min	1.000000	1.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	1
25%	161.000000	1.000000	5.000000	2.000000	0.000000	0.000000	0.000000	0.000000	1
50%	319.000000	1.000000	10.000000	4.000000	0.000000	0.000000	0.000000	0.000000	1
75%	476.000000	2.000000	15.000000	5.000000	0.000000	0.000000	0.000000	0.000000	1
max	636.000000	4.000000	20.000000	9.000000	1.000000	5.000000	4.000000	5.000000	1

bowler performance from team - Royal Challengers Bangalore

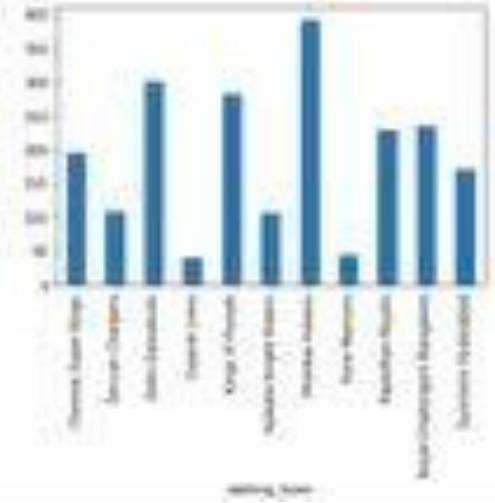
```
[illegible]
```



bowler performance from team - Sunrisers Hyderabad



bowler performance from team - Rising Pune Supergiant

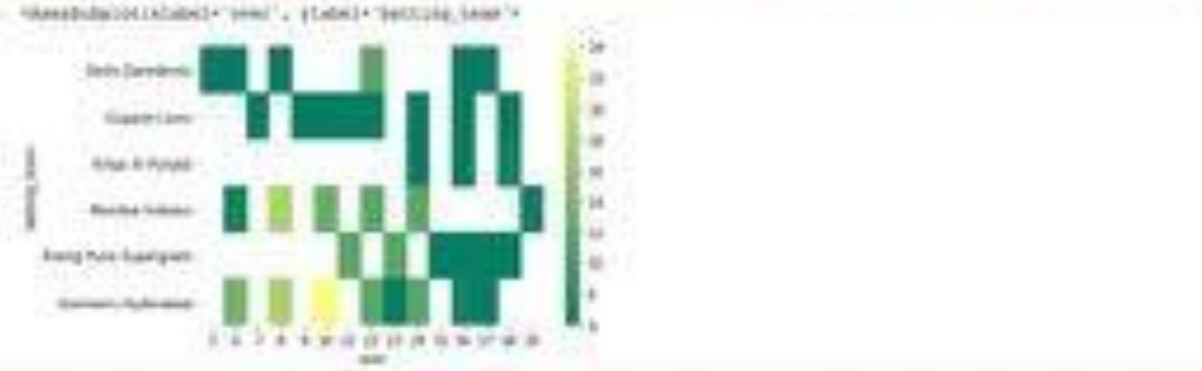


bowler performance from team - Mumbai Indians

bowler performance from team - Kolkata Knight Riders

Over-wise Batting Performance of Each Team in the IPL

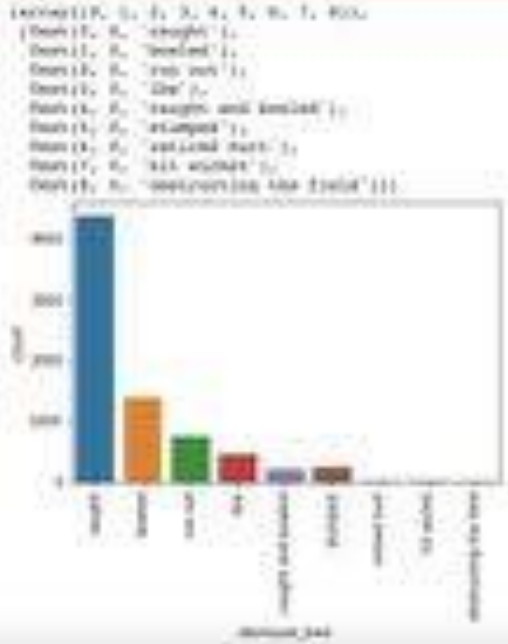
```
deliveries=d_df[mask]
deliveries=deliveries[['batting_team','over','batsman_runs']]
grouped=deliveries.groupby(['batting_team','over'],as_index=False).sum()
sns.heatmap(a, cmap='summer')
```



Dismissal kind

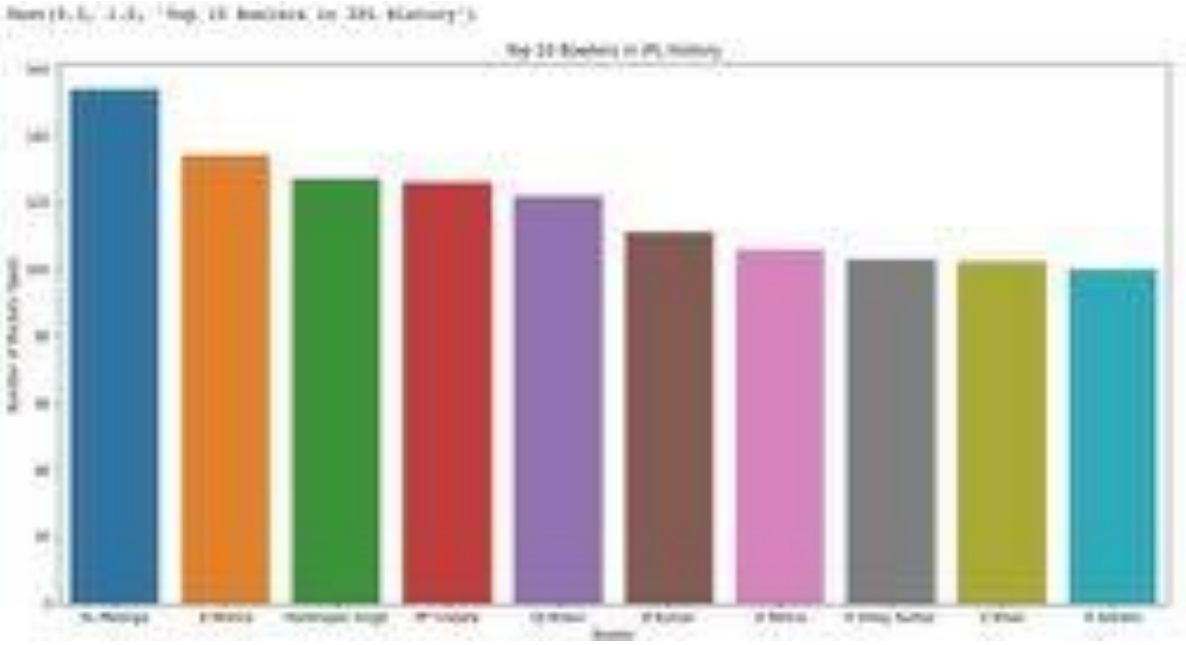
```
sns.countplot('dismissal_kind', data=d_df)
plt.xticks(rotation=90)
```

```
C:\Users\DELL\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```



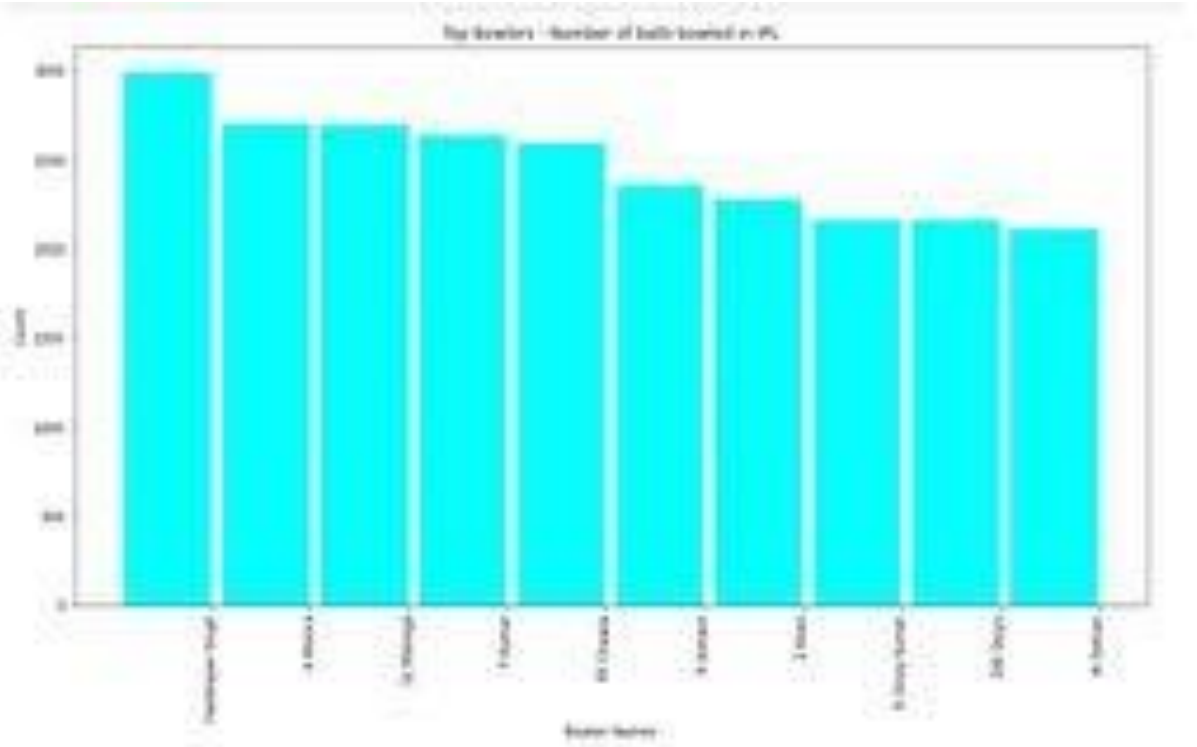
TOP 10 BOWLERS IN IPL HISTORY

```
# There are many things we investigate
# First let's consider which bowler has taken the most wickets
wickets=dismissal_df[(dismissal_df['dismissal_kind']!='retired hurt')
                     &(dismissal_df['dismissal_kind']!='obstructing the field')
                     &(dismissal_df['dismissal_kind']!='run out')].groupby(['bowler'])['dismissal_kind'].count().sor
f,ax=plt.subplots(figsize=(16,8))
sns.barplot(wickets['bowler'][:10],wickets['dismissal_kind'][:10])
plt.xlabel('Bowler')
plt.ylabel('Number of Wickets Taken')
plt.title('Top 10 Bowlers in IPL History')
```



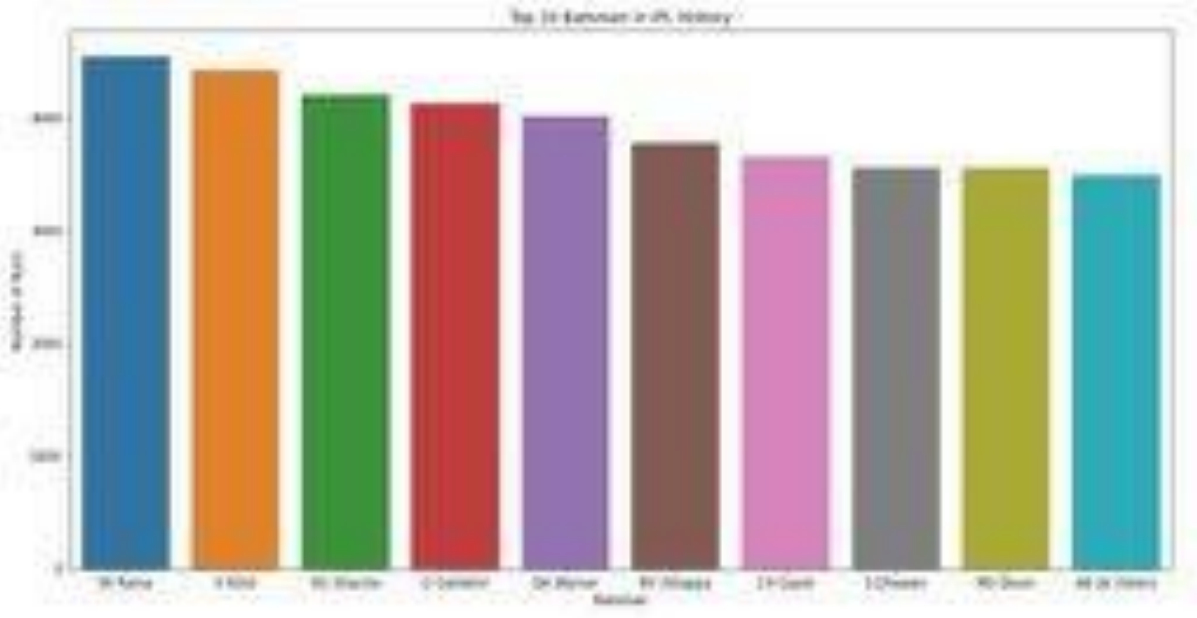
the bowlers who has bowled most number of balls in IPL

```
temp_df = d_df.groupby('bowler')['ball'].agg('count').reset_index().sort_values(by='ball', ascending=False).reset_in
temp_df = temp_df.iloc[:10,:]
labels = np.array(temp_df['bowler'])
ind = np.arange(len(labels))
width = 0.5
fig, ax = plt.subplots(figsize=(15,8))
rects = ax.bar(ind, np.array(temp_df['ball']), width=width, color='cyan')
ax.set_xticks(ind+((width)/2.))
ax.set_xticklabels(labels, rotation='vertical')
ax.set_ylabel("Count")
ax.set_title("Top Bowlers - Number of balls bowled in IPL")
ax.set_xlabel('Bowler Names')
plt.show()
#Harbhajan Singh is the the bowler with most number of balls bowled in IPL matches.
```

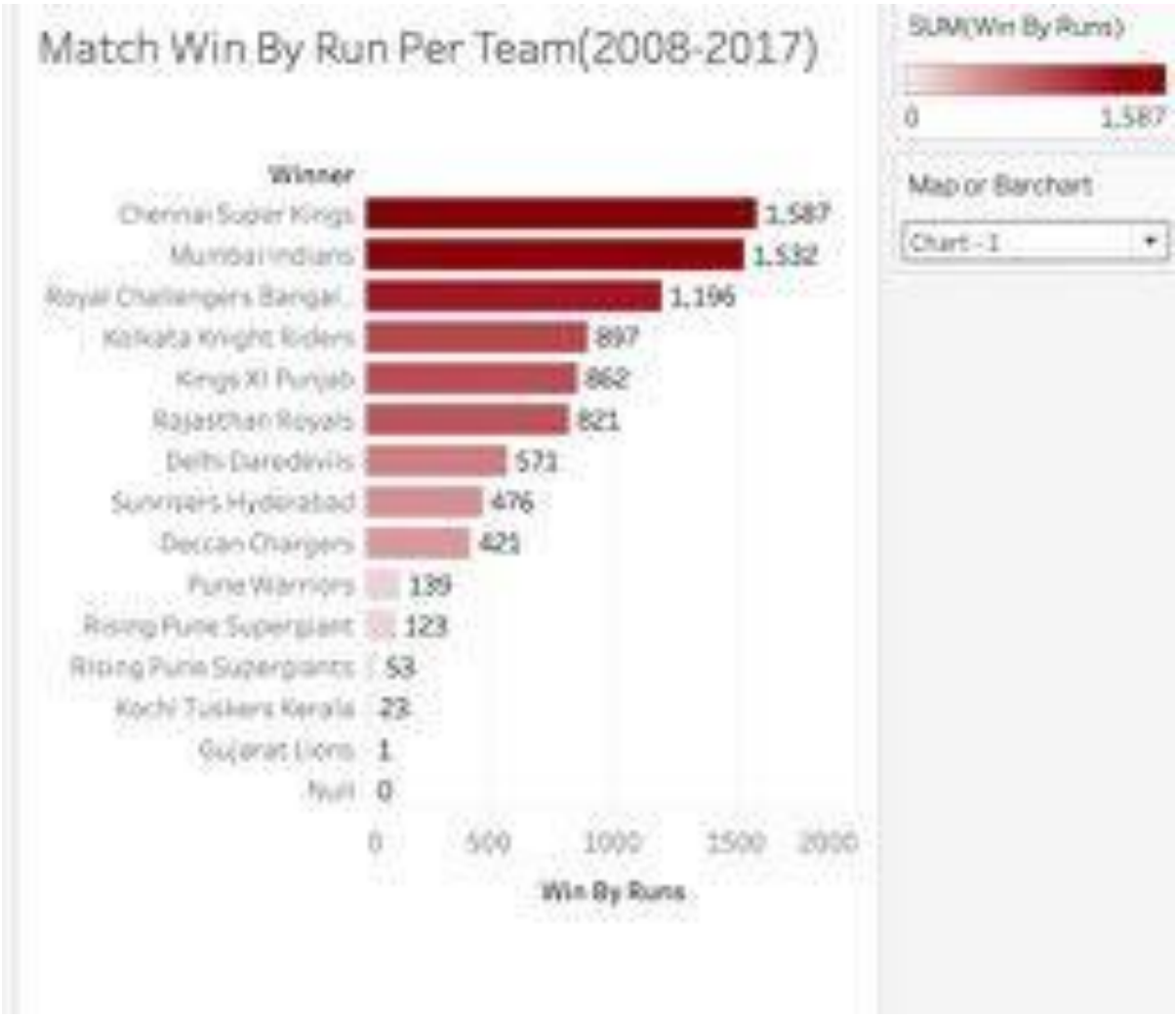


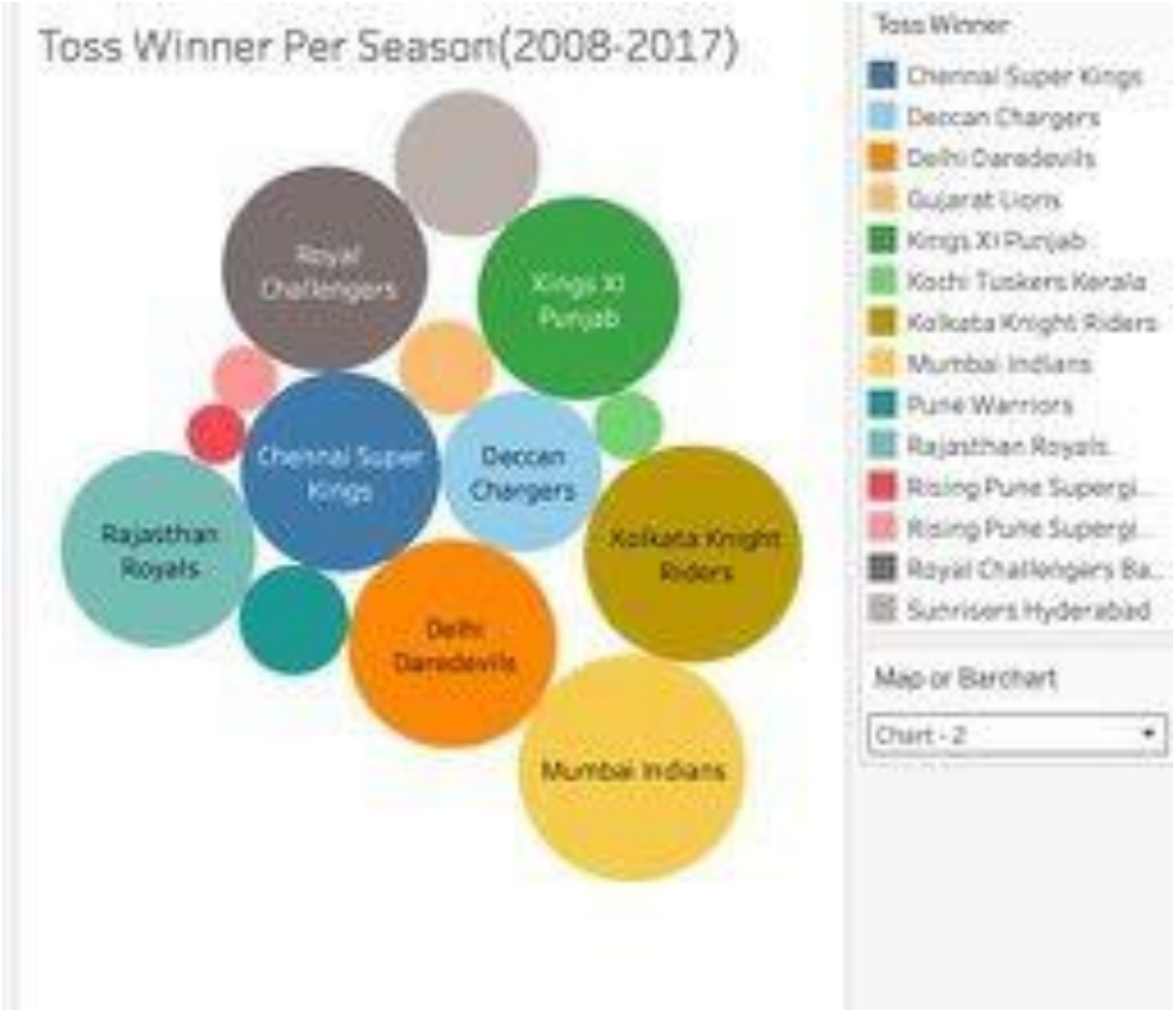
Top 10 Batsmen in IPL History

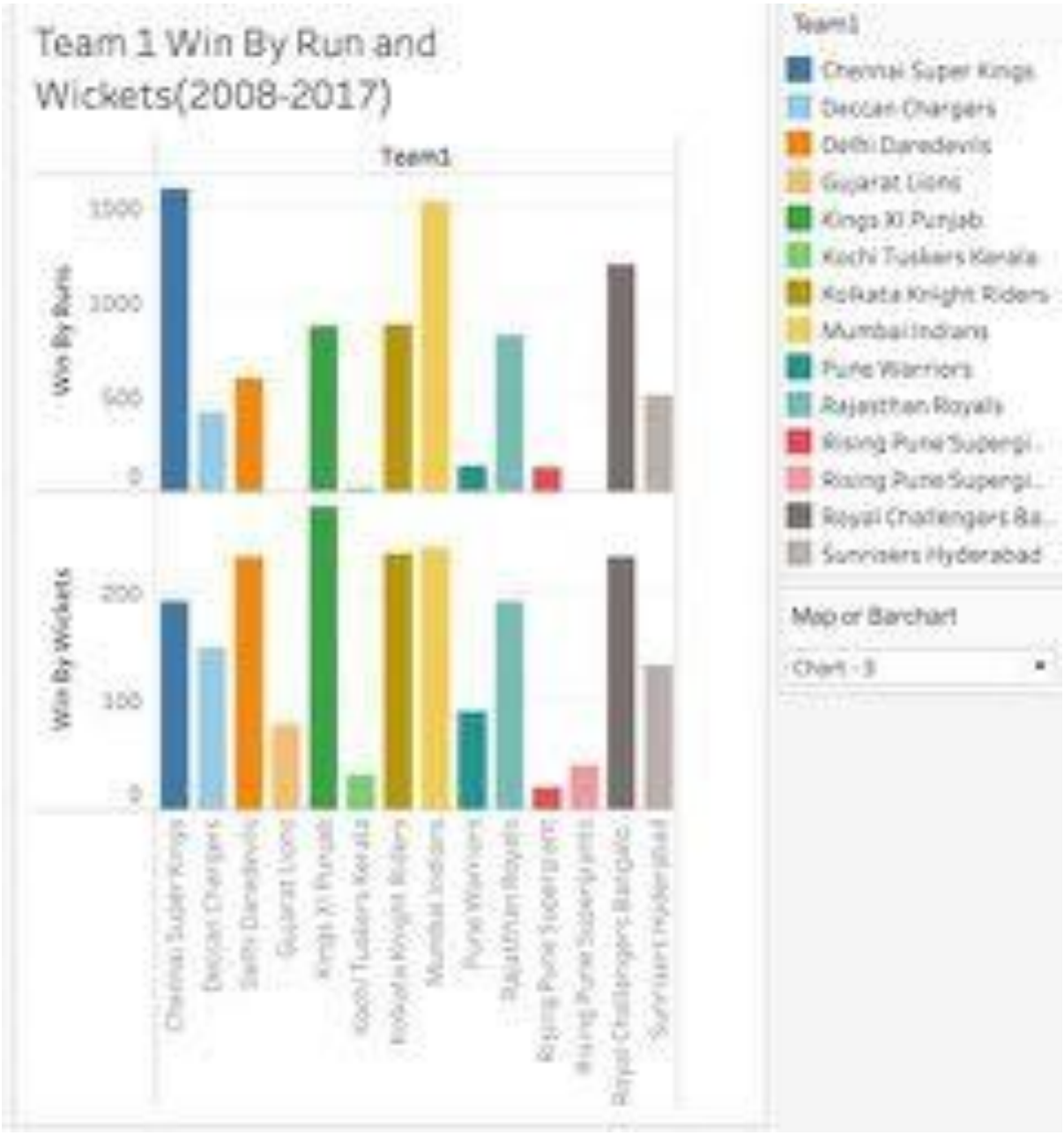
```
runs=batting_df.groupby(['batsman'])
runs=runs.agg({'batsman_runs':'sum','ball':'count','player_dismissed':'count'})
runs=runs.sort_values(by='batsman_runs',ascending=False).reset_index()
f,ax=plt.subplots(figsize=(16,8))
sns.barplot(runs['batsman'][:10],runs['batsman_runs'][:10])
plt.xlabel('Batsman')
plt.ylabel('Number of Runs')
plt.title('Top 10 Batsmen in IPL History')
```



MATCHES DATASET :

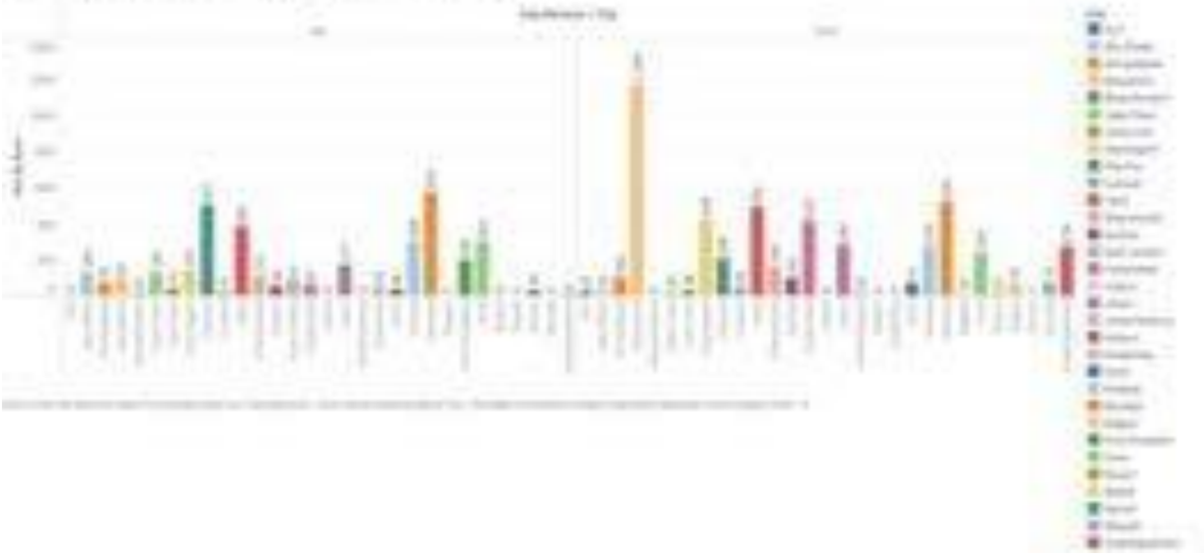



Toss Winner Per Season(2008-2017)
Royal Challengers
Kings XI Punjab
Chennai Super Kings
Deccan Chargers
Rajasthan Royals
Kolkata Knight Riders
Delhi Daredevils
Mumbai Indians
Toss Winner
Chennai Super Kings
Deccan Chargers
Delhi Daredevils
Gujarat Lions
Kings XI Punjab
Kochi Tuskers Kerala
Kolkata Knight Riders
Mumbai Indians
Pune Warriors
Rajasthan Royals
Rising Pune Supergi...
Rising Pune Supergi...
Royal Challengers Ba...
Sunrisers Hyderabad
Map or Barchart
Chart - 2

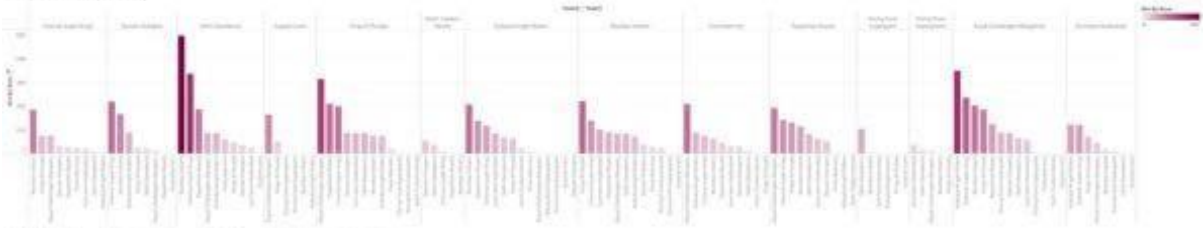

Team 1 Win By Run and Wickets(2008-2017)
Team1
Win By Runs
1500
1000
500
0
Win By Wickets
200
100
0
Chennai Super Kings
Deccan Chargers
Delhi Daredevils
Gujarat Lions
Kings XI Punjab
Kochi Tuskers Kerala
Kolkata Knight Riders
Mumbai Indians
Pune Warriors
Rajasthan Royals
Rising Pune Supergi...
Rising Pune Supergi...
Royal Challengers Ba...
Sunrisers Hyderabad
Team1
Chennai Super Kings
Deccan Chargers
Delhi Daredevils
Gujarat Lions
Kings XI Punjab
Kochi Tuskers Kerala
Kolkata Knight Riders
Mumbai Indians
Pune Warriors
Rajasthan Royals
Rising Pune Supergi...
Rising Pune Supergi...
Royal Challengers Ba...
Sunrisers Hyderabad
Map or Barchart
Chart - 3


City Wise Match Date(2008-2017)
Year of Date
2010
2015
Chandigarh
Jaipur
Nagpur
Mumbai
Chennai
City
Null
Abu Dhabi
Ahmedabad
Bangalore
Bloemfontein
Cape Town
Centurion
Chandigarh
Chennai
Cuttack
Delhi
Dharamsala
Map or Barchart
Chart - 4

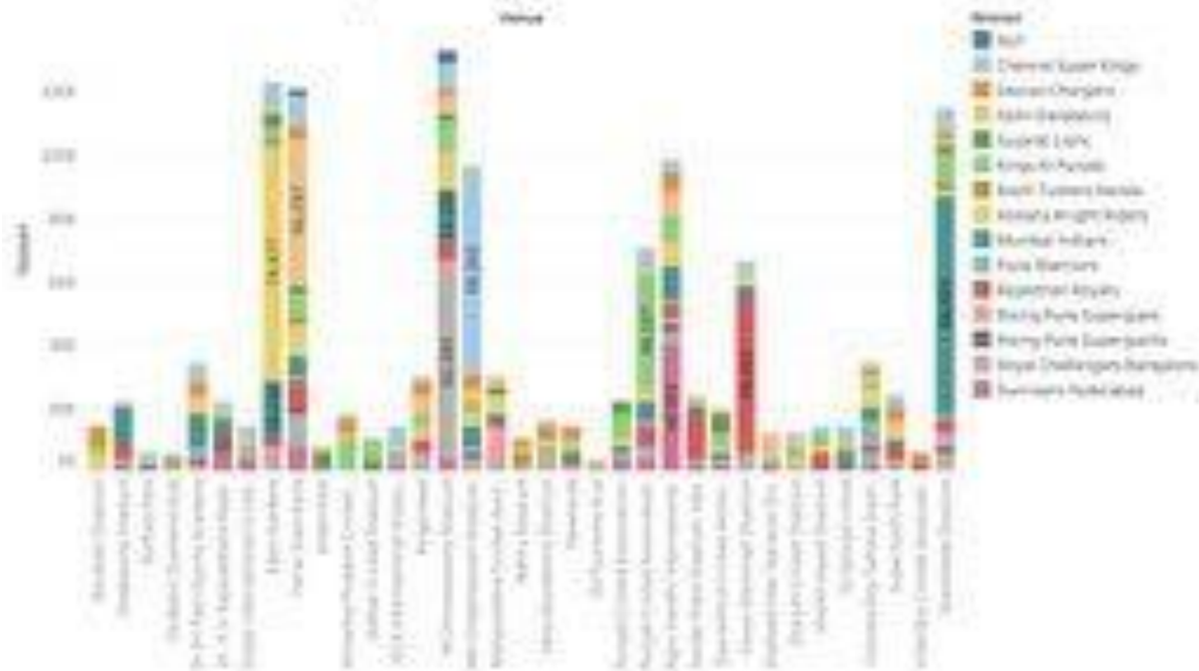
City Wise View of Team Winning in Total Seasons(2008-2017)

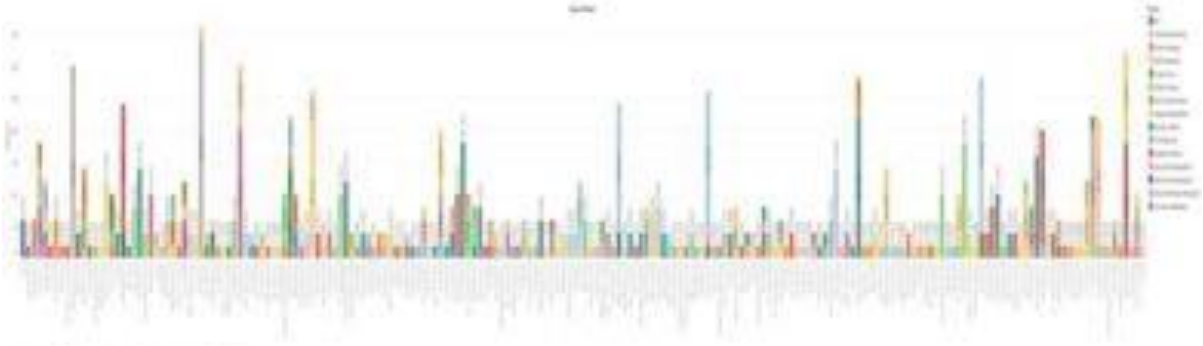


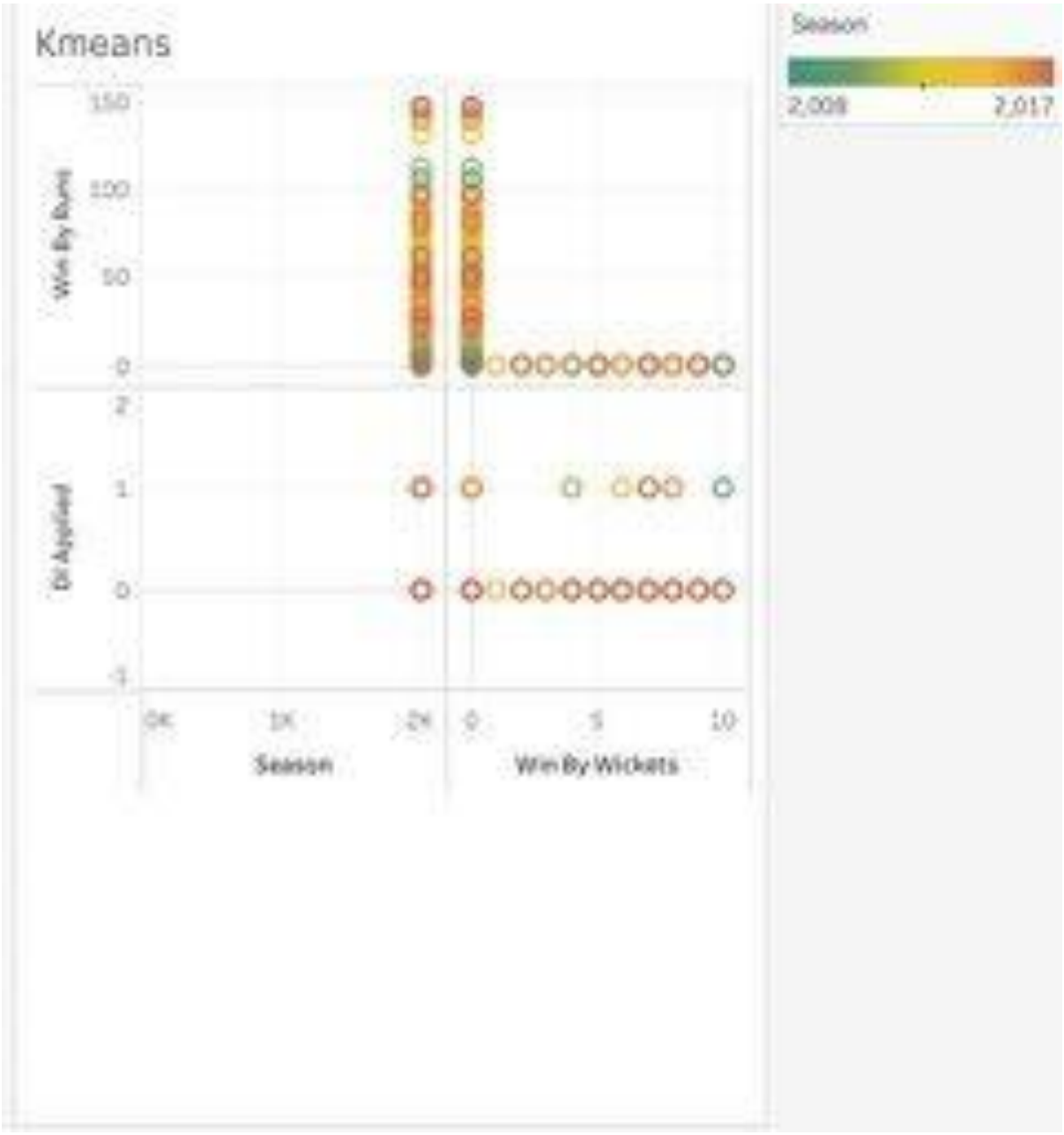
Win By Runs Per Team(2008-2017)

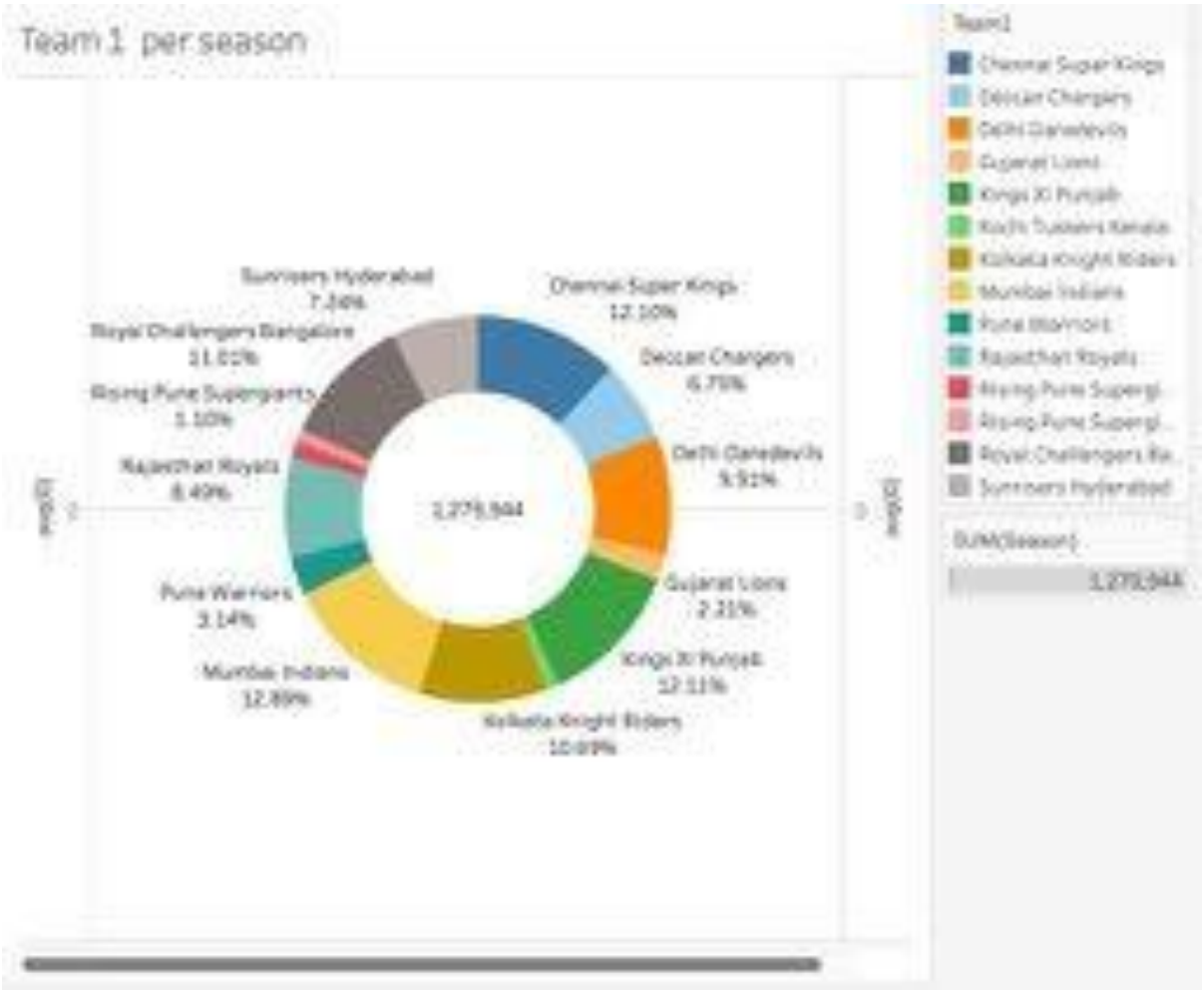


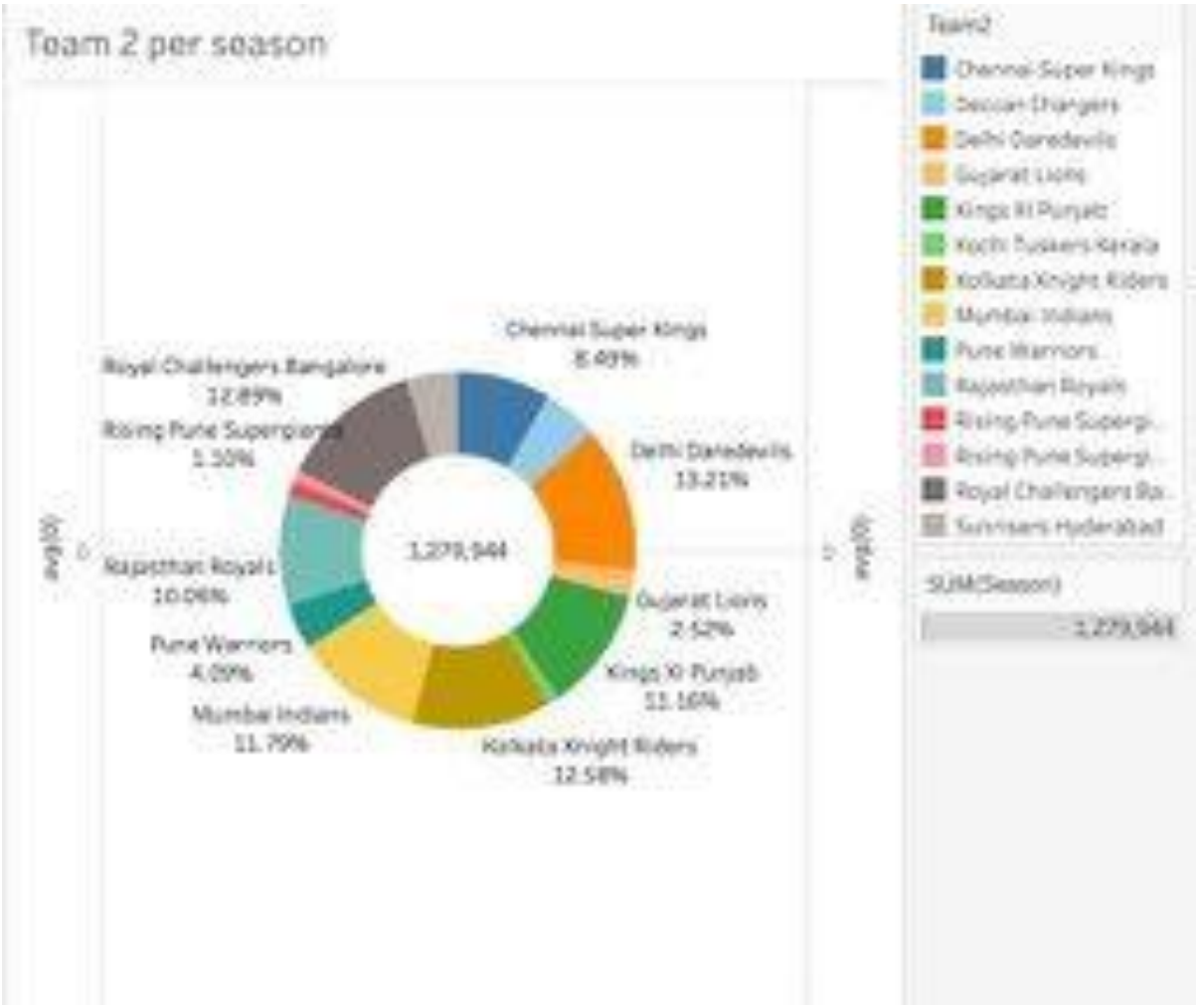
Winner Per Season and Venue(2008-2017)

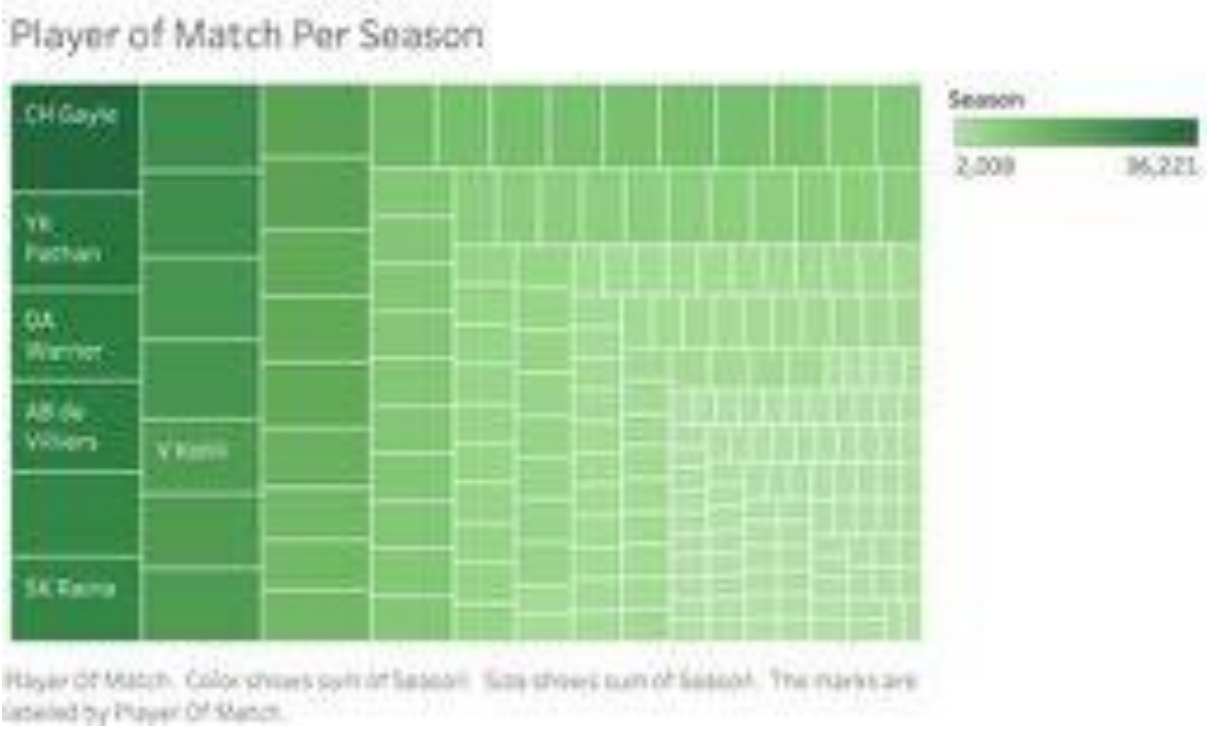




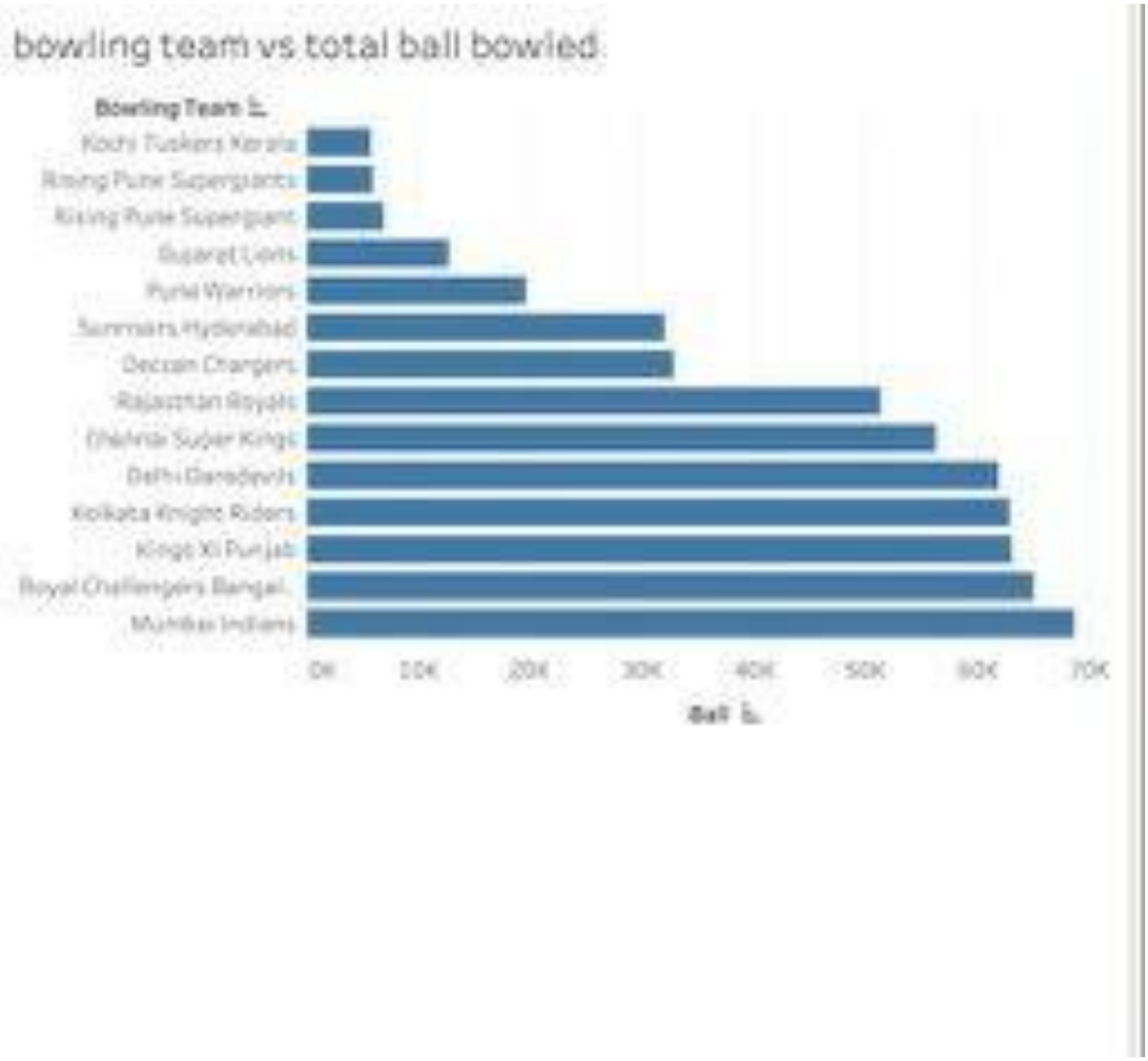

Kmeans
Season
2,008
2,017
Win By Runs
150
100
50
0
DI Applied
2
1
0
-1
0K
1K
2K
Season
0
5
10
Win By Wickets

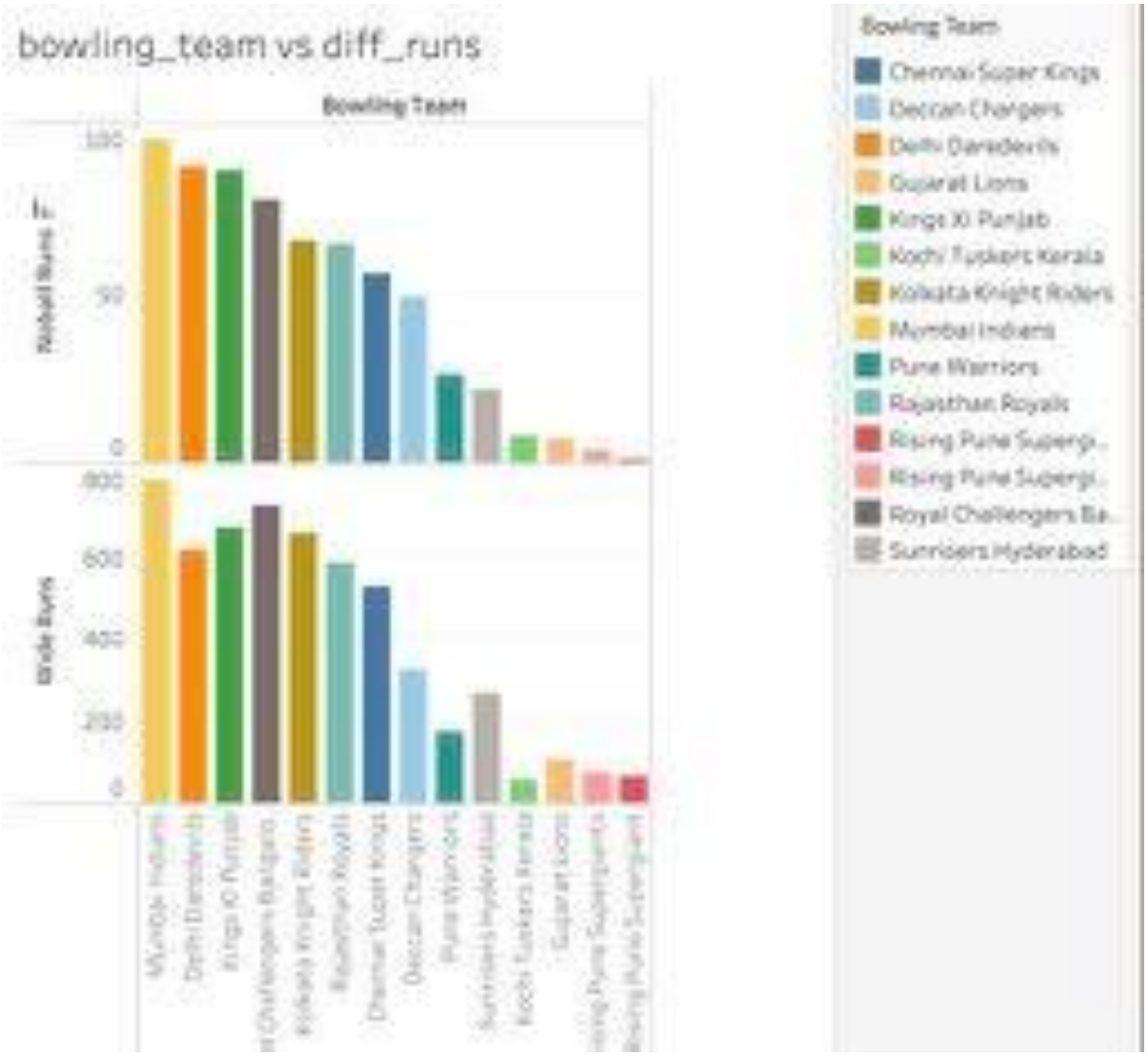

Team1 per season
Sunrisers Hyderabad
7.24%
Chennai Super Kings
12.10%
Royal Challengers Bangalore
11.01%
Deccan Chargers
6.75%
Rising Pune Supergiants
1.10%
Delhi Daredevils
9.51%
Rajasthan Royals
8.49%
1,279,944
Gujarat Lions
2.21%
Pune Warriors
3.14%
Kings XI Punjab
12.11%
Mumbai Indians
12.89%
Kolkata Knight Riders
10.09%
SUM(Season)
1,279,944

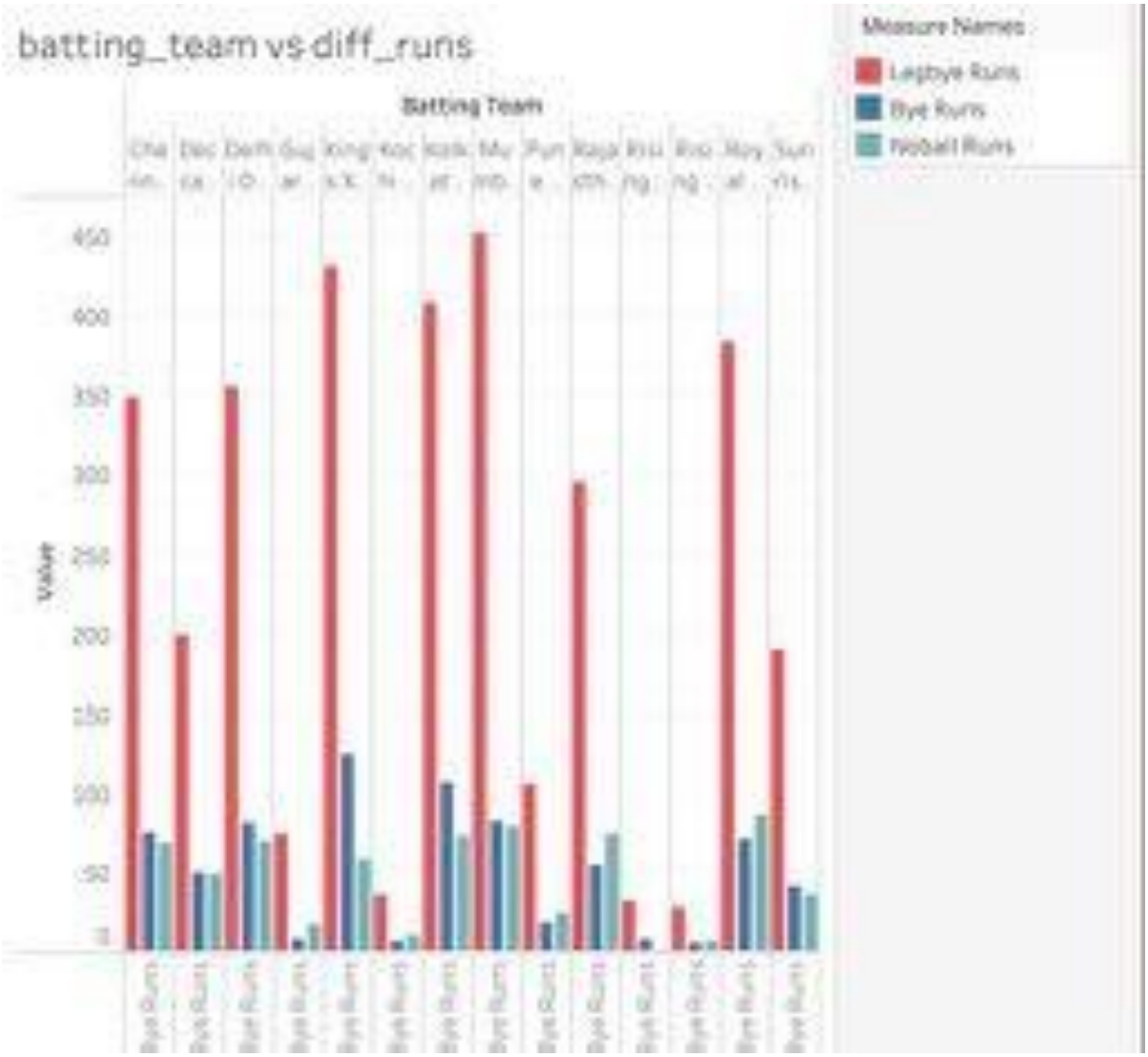

Team 2 per season
Chennai Super Kings
8.49%
Delhi Daredevils
13.21%
Gujarat Lions
2.52%
Kings XI Punjab
11.16%
Kolkata Knight Riders
12.58%
Mumbai Indians
11.79%
Pune Warriors
4.09%
Rajasthan Royals
10.06%
Rising Pune Supergiants
1.55%
Royal Challengers Bangalore
12.89%
1,279,944
avg(0)
avg(0)
Team2
Chennai Super Kings
Deccan Chargers
Delhi Daredevils
Gujarat Lions
Kings XI Punjab
Kochi Tuskers Kerala
Kolkata Knight Riders
Mumbai Indians
Pune Warriors
Rajasthan Royals
Rising Pune Supergi...
Rising Pune Supergi...
Royal Challengers Ba...
Sunrisers Hyderabad
SUM(Season)
1,279,944

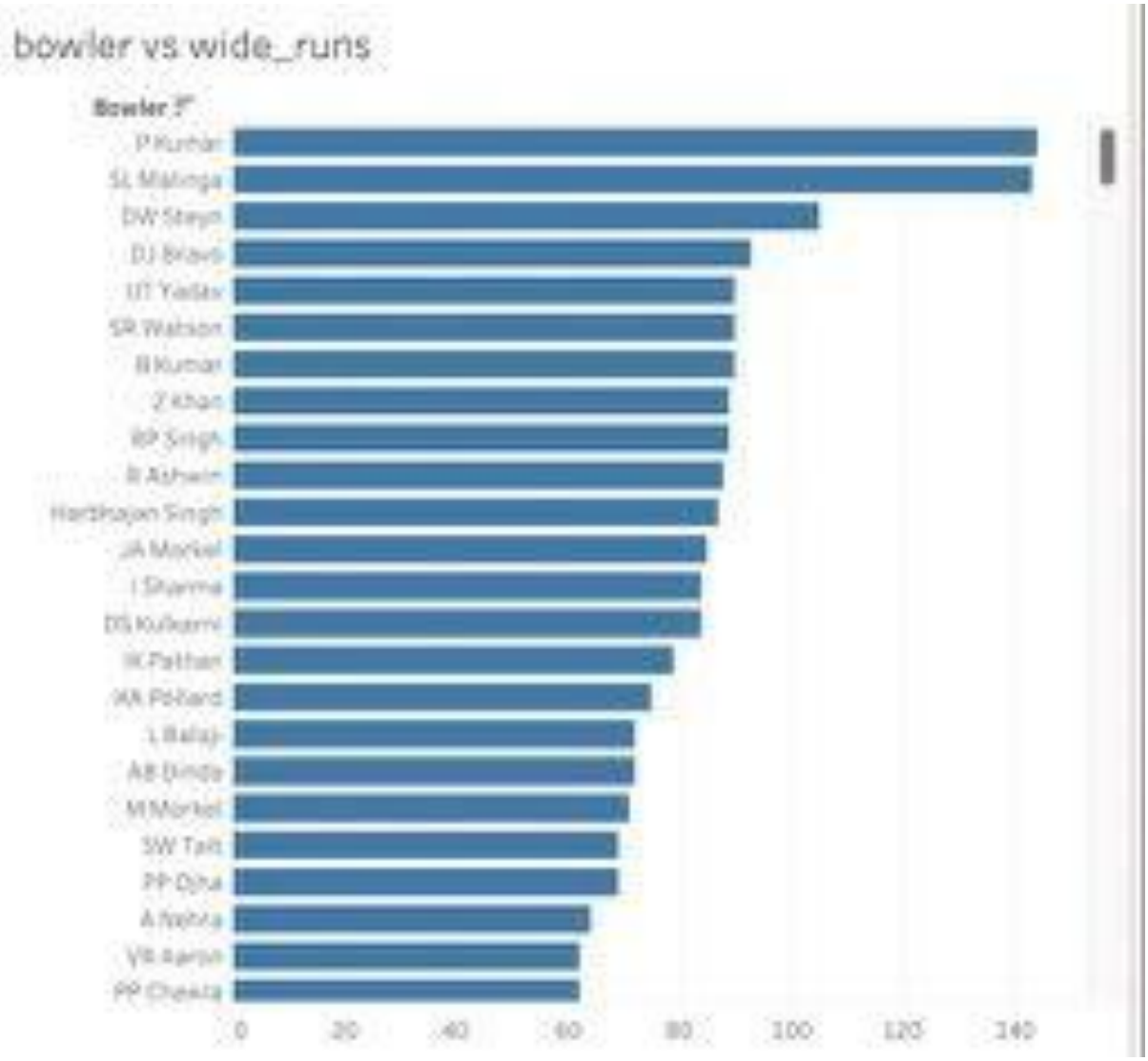


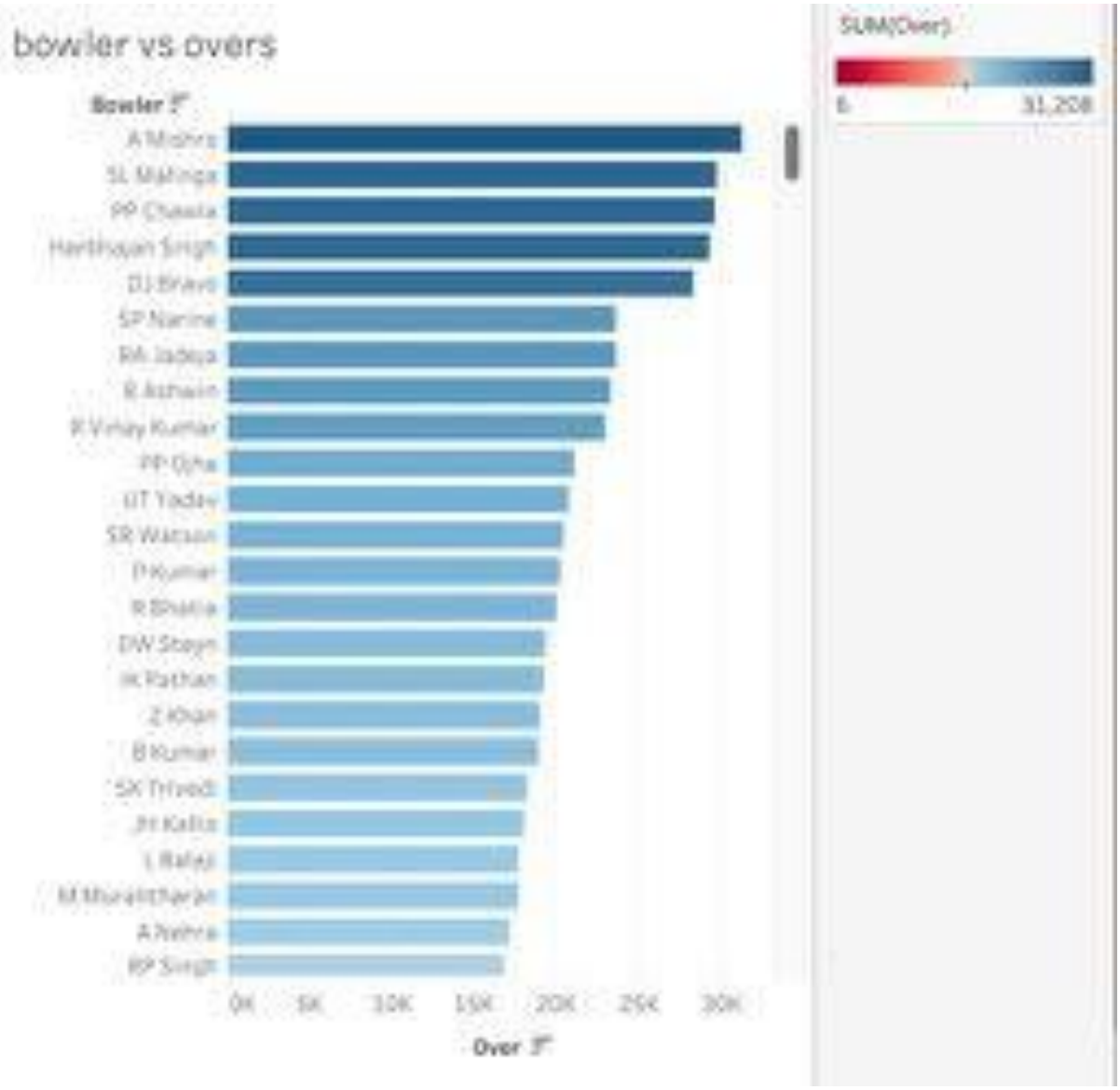
DELIVERIES DATASET :

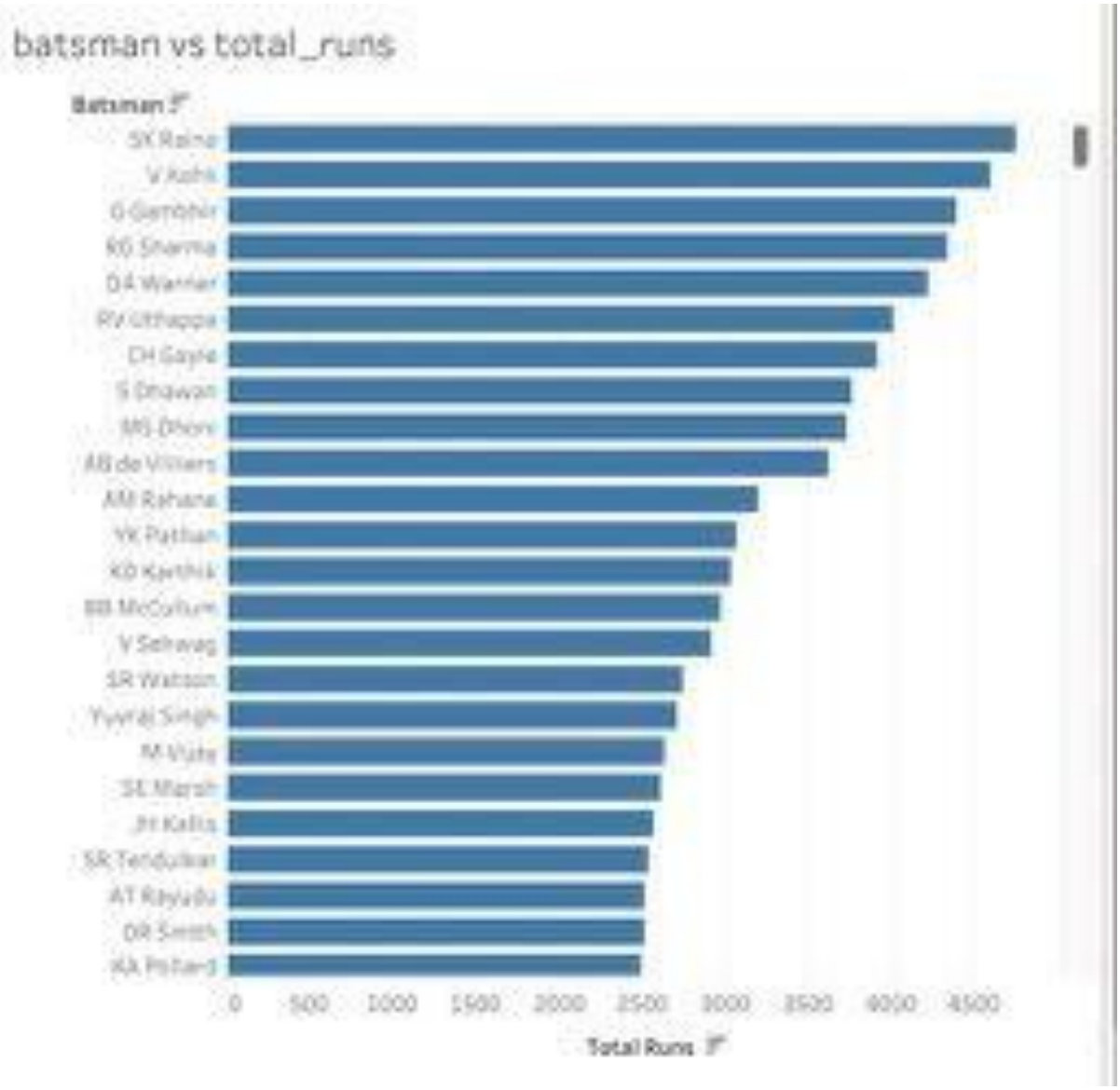

bowling team vs total ball bowled
Bowling Team
Kochi Tuskers Kerala
Rising Pune Supergiants
Rising Pune Supergiant
Gujarat Lions
Pune Warriors
Sunrisers Hyderabad
Deccan Chargers
Rajasthan Royals
Chennai Super Kings
Delhi Daredevils
Kolkata Knight Riders
Kings XI Punjab
Royal Challengers Bangal..
Mumbai Indians
0K
10K
20K
30K
40K
50K
60K
70K
Ball


bowling_team vs diff_runs
Bowling Team
Sum of Total Runs
Sum of Extra Runs
Chennai Super Kings
Deccan Chargers
Delhi Daredevils
Gujarat Lions
Kings XI Punjab
Kochi Tuskers Kerala
Kolkata Knight Riders
Mumbai Indians
Pune Warriors
Rajasthan Royals
Rising Pune Supergi..
Rising Pune Supergi..
Royal Challengers Ba..
Sunrisers Hyderabad


batting_team vs diff_runs
Batting Team
Measure Names
Legbye Runs
Bye Runs
Noball Runs
Value
450
400
350
300
250
200
150
100
50
0


bowler vs wide_runs
Bowler
P Kumar
SL Malinga
DW Steyn
DJ Bravo
UT Yadav
SR Watson
B Kumar
Z Khan
RP Singh
R Ashwin
Harbhajan Singh
JA Morkel
I Sharma
DS Kulkarni
IK Pathan
KA Pollard
L Balaji
AB Dinda
M Morkel
SW Tait
PP Ojha
A Nehra
VR Aaron
PP Chawla
0
20
40
60
80
100
120
140

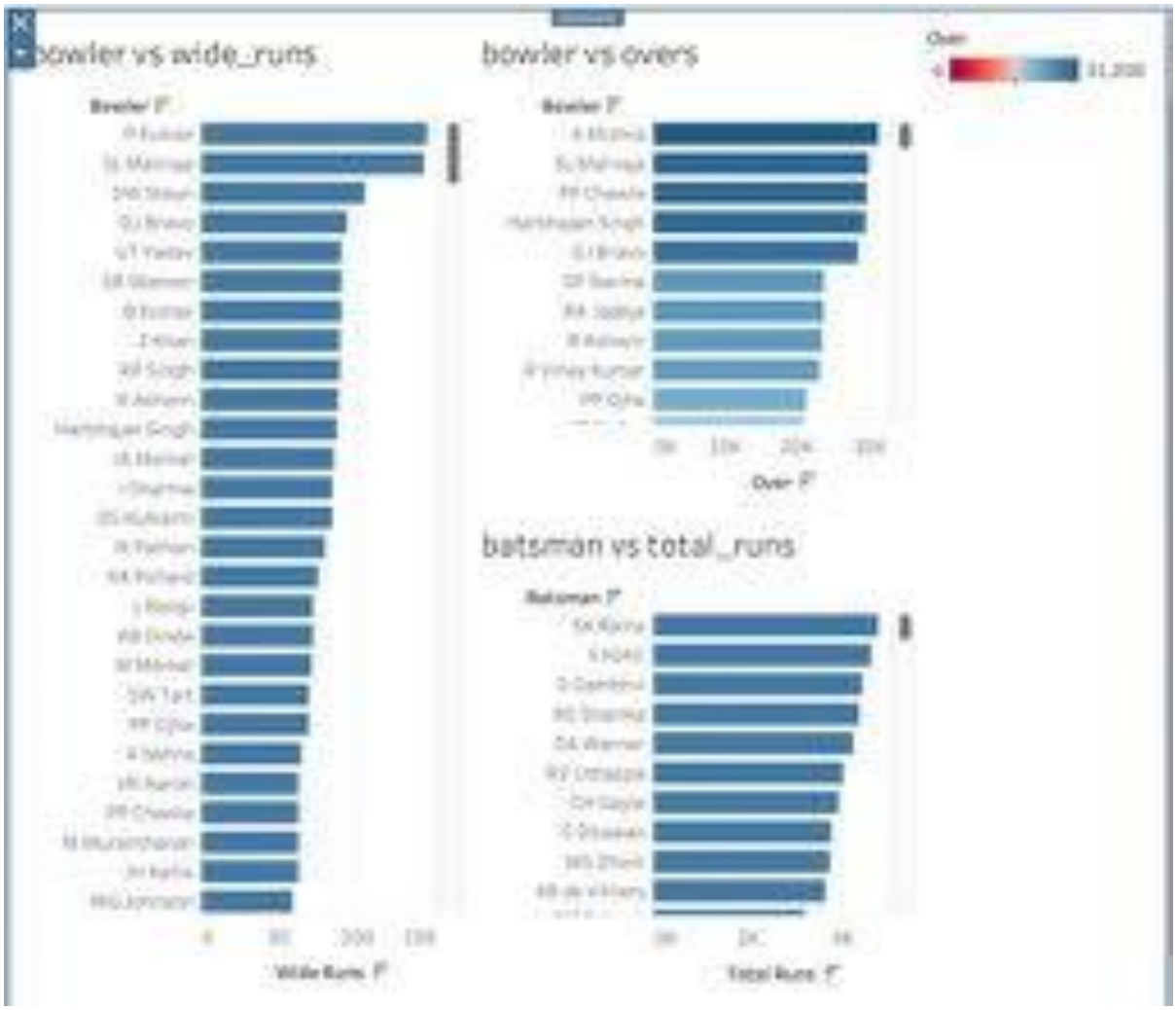

bowler vs overs
Bowler
A Mishra
SL Malinga
PP Chawla
Harbhajan Singh
DJ Bravo
SP Narine
RA Jadeja
R Ashwin
R Vinay Kumar
PP Ojha
UT Yadav
SR Watson
P Kumar
R Bhatia
DW Steyn
M Pathan
Z Khan
B Kumar
SK Trivedi
JH Kallis
L Balaji
M Muralitharan
A Nehra
RP Singh
0K
5K
10K
15K
20K
25K
30K
Over
SUM(Over)
6
31,208


batsman vs total_runs
Batsman
SK Raina
V Kohli
G Gambhir
KD Sharma
DA Warner
RV Uthappa
CH Gayle
S Dhawan
MS Dhoni
AB de Villiers
AM Rahane
YK Pathan
KD Karthik
BB McCullum
V Sehwag
SR Watson
Yuvraj Singh
M Vijay
SE Marsh
JH Kallis
SR Tendulkar
AT Rayudu
DR Smith
KA Pollard
0
500
1000
1500
2000
2500
3000
3500
4000
4500
Total Runs

DASHBOARD - 1


batting team vs total runs
bowling team vs total ball bowled
bowling_team vs diff_runs
batting_team vs diff_runs

DASHBOARD – 2



CONCLUSION :

The above work aims at understanding the dataset of past 10 years history of the IPL data. It helps to understand the visualisation concept , tableau working and k-means clustering The model classifies the data and compares the results. It takes into consideration the measures accuracy, error rate, precision, recall, sensitivity and specificity. The work focuses on exploring IPL data and presenting its insights as graphical representation and comparative analysis. By making use of this, Indian Premier League and the fan followers can take decisions on the team's performance and predict the trophy winners that will lead to success in future.

Selection of the best team for a cricket match plays a significant role for the team's victory. The main goal of this project is to analyse the IPL cricket data and predict the players' performance. The implementation tools used are Anaconda navigator and Jupyter, tableau. The knowledge gained from the insights will be used in future to predict the winning teams for the next series IPL matches. Hence using this prediction, the best team can be formed.

REFERENCES :

http://www.ijirset.com/upload/2019/april/69_Analyzing.pdf

https://jovian.ai/beast7rk/ipl-data-analysis-project

https://towardsdatascience.com/exploratory-data-analysis-of-ipl-matches-part-1-c3555b15edbb

https://www.kaggle.com/code/razamh/eda-ipl-dataset/data