IPL DATA ANALYSIS

Abstract

The Indian Premier League (IPL) dataset is examined in this study to examine team performance, player awards, and match results. Match outcomes are predicted using logistic regression using historical data. The results improve decision-making during games and offer insightful information to IPL stakeholders.

MATHEMATICS FOR ENGINEERS II PROJECT

PROJECT NO:

A Project Submitted

In Partial Fulfilment for the

Degree of

BACHELOR Of Technology

In

Computer Science

SUBMITTED BY

GROUP NO:

SHEENA MITTAL 220356

MANASVI ARORA 220362

DAKSH GOEL 220559

BML MUNJAL UNIVERSITY™

SCHOOL OF ENGINEERING AND TECHNOLOGY

BML MUNJAL UNIVERSITY GURGAON

2023

ACKNOWLEDGEMENT

I would like to extend my sincere gratitude to everyone who helped to complete this report, which is completely original.

I want to start by expressing my gratitude to my supervisor Mr. Ranjib Banerjee who has offered me invaluable advice and support throughout the entire process. Their knowledge, helpful criticism, and encouragement were crucial to the success of this report.

I also want to express my gratitude to the staff and professors who have guided and taught me throughout my academic career. They have instilled in me a strong sense of academic integrity and ethical behaviour thanks to their dedication and commitment to excellence.

I also want to express my gratitude to my co-workers and classmates who have given me insightful criticism and recommendations. Their encouraging words and helpful criticism have assisted me in raising the calibre of my work.

Finally, I want to express my gratitude to my family and friends for their unwavering support and inspiration. My motivation has been sustained throughout the writing process by their support and love.

I would like to once more extend my sincere gratitude to everyone who helped to complete this report.

TABLE OF CONTENT

PROBLEM STATEMENT

The current issue is to analyse and investigate the Indian Premier League (IPL) dataset, which contains data on games and ball-by-ball information from 2008 to 2022. The goal is to create a predictive model that uses historical match and delivery data to forecast cricket matches in the IPL. By taking into account various factors like the current score, runs left, balls left, wickets, and run rate, the goal is to develop a logistic regression model that accurately predicts whether a team will win or lose a match. Insights into team performance, player awards, match results, season-long trends, and other pertinent factors are also sought after by the analysis.

The analysis aims to address the following questions:

- ❖ Can we accurately predict the outcome of IPL matches using logistic regression and historical match data?
- ❖ What are the key features or variables that influence the match outcome prediction?
- ❖ How does the current score, runs left, balls left, and wickets impact the prediction of a team's chances of winning or losing a match?
- ❖ What is the accuracy of the logistic regression model in predicting match outcomes?
- ❖ Which teams have the highest number of wins in the IPL, and how is the distribution of wins among different teams?
- ❖ Which players have received the most "Player of the Match" awards, and can we identify the top 10 players in terms of awards received?
- ❖ How many matches were played in each IPL season, and can we visualize the season-wise number of matches?
- ❖ What are the different ways in which teams have won matches, and can we analyse and count the number of matches won by runs and by wickets?
- ❖ Which cities have hosted the most matches, and which teams have the highest number of wins in each city?

INTRODUCTION

The Indian Premier League (IPL) has revolutionized the world of cricket, captivating audiences with its thrilling matches, star players, and electrifying atmosphere. As a professional Twenty20 cricket league, the IPL has not only provided entertainment but has also generated a wealth of data that holds immense potential for analysis and insights. This report serves as a comprehensive exploration of the IPL dataset, encompassing match details and ball-by-ball information from 2008 to 2022.

The objective of this analysis is to delve into various aspects of the IPL and uncover meaningful patterns, trends, and relationships within the data. By leveraging statistical techniques, data visualization, and predictive modelling, we aim to extract valuable insights that can enhance our understanding of team performance, player contributions, match outcomes, and season-wise dynamics.

One of the primary goals is to develop a predictive model using logistic regression to forecast the outcome of IPL matches. By utilizing historical match data and relevant features such as current score, runs left, balls left, wickets, and run rate, we seek to ascertain the factors that significantly impact a team's chances of winning or losing. The accuracy and efficacy of the predictive model will be evaluated, providing valuable information on the predictive power of logistic regression in the context of IPL matches.

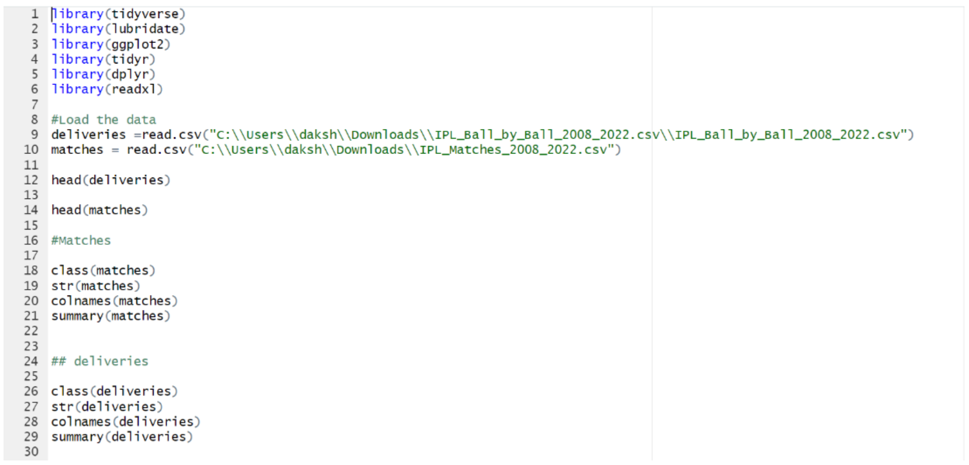
Additionally, this analysis explores the distribution of wins among IPL teams, identifying the top-performing teams with the highest number of victories. We also examine the "Player of the Match" awards, recognizing the players who have consistently stood out and made significant contributions to their teams' success. Furthermore, season-wise trends in the number of matches played and different modes of match outcomes (runs and wickets) are investigated to identify patterns and variations across IPL seasons.

The analysis employs the powerful capabilities of the tidyverse and ggplot2 libraries in R, facilitating efficient data manipulation, visualization, and summarization. By presenting the findings in a visually appealing and informative manner, this report aims to provide IPL stakeholders, teams, players, and enthusiasts with valuable insights into the tournament's dynamics, performance metrics, and strategic decision-making.

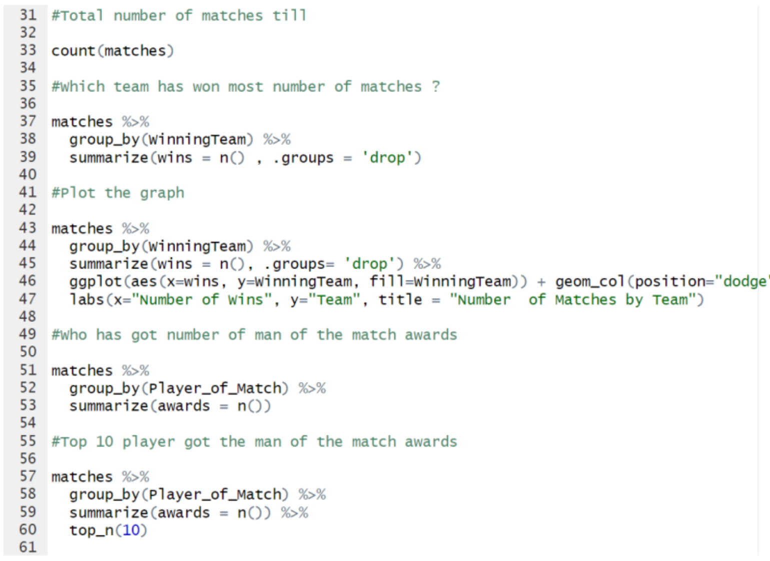
Overall, this analysis endeavours to unravel the rich tapestry of IPL data, shedding light on the factors that drive success in the tournament and showcasing the potential of data-driven approaches in cricket. The knowledge gained from this analysis can inform team strategies, player selections, and fan engagement, contributing to a deeper appreciation and understanding of the IPL's captivating journey.

Results of Simulations

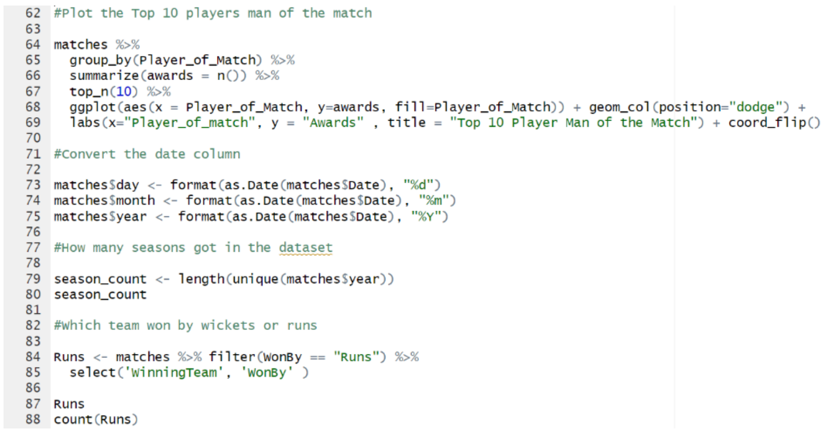
Result 1 - IPL Data Analysis



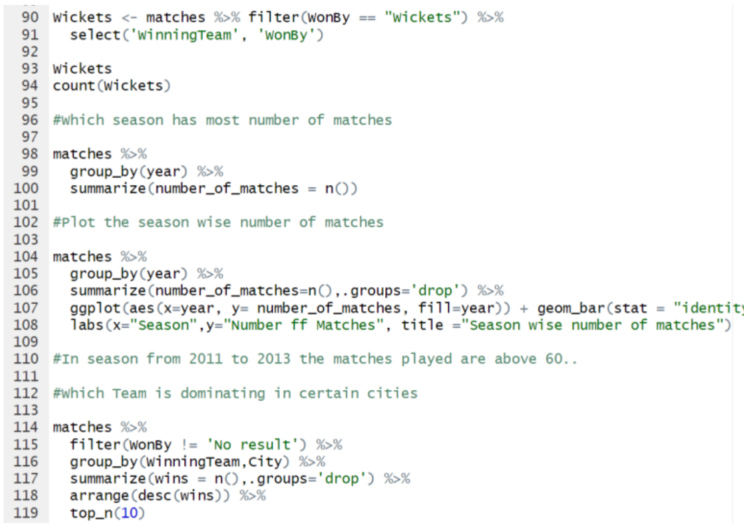
```r
library(tidyverse)
library(lubridate)
library(ggplot2)
library(tidyr)
library(dplyr)
library(readxl)

#Load the data
deliveries =read.csv("C:\\Users\\daksh\\Downloads\\IPL_Ball_by_Ball_2008_2022.csv\\IPL_Ball_by_Ball_2008_2022.csv")
matches = read.csv("C:\\Users\\daksh\\Downloads\\IPL_Matches_2008_2022.csv")

head(deliveries)

head(matches)

#Matches

class(matches)
str(matches)
colnames(matches)
summary(matches)


## deliveries

class(deliveries)
str(deliveries)
colnames(deliveries)
summary(deliveries)

```



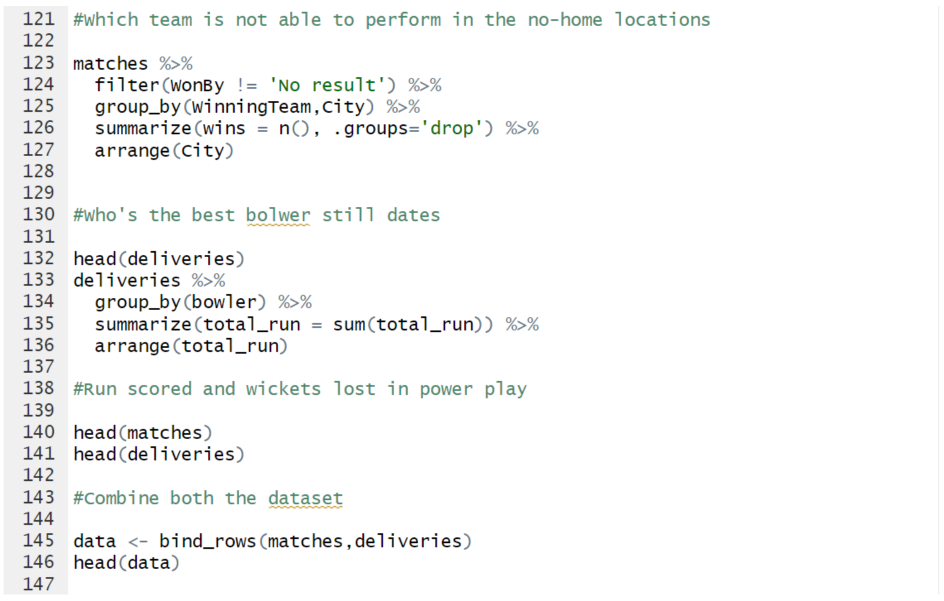
```r
#Total number of matches till

count(matches)

#Which team has won most number of matches ?

matches %>%
  group_by(WinningTeam) %>%
  summarize(wins = n() , .groups = 'drop')

#Plot the graph

matches %>%
  group_by(WinningTeam) %>%
  summarize(wins = n(), .groups= 'drop') %>%
  ggplot(aes(x=wins, y=WinningTeam, fill=WinningTeam)) + geom_col(position="dodge") +
  labs(x="Number of Wins", y="Team", title = "Number  of Matches by Team")

#Who has got number of man of the match awards

matches %>%
  group_by(Player_of_Match) %>%
  summarize(awards = n())

#Top 10 player got the man of the match awards

matches %>%
  group_by(Player_of_Match) %>%
  summarize(awards = n()) %>%
  top_n(10)

```



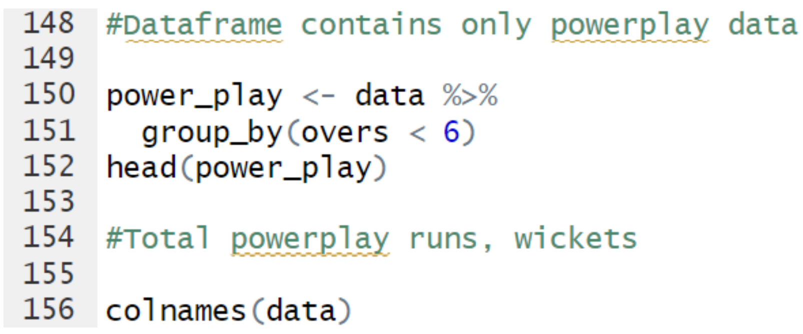
```r
#Plot the Top 10 players man of the match

matches %>%
  group_by(Player_of_Match) %>%
  summarize(awards = n()) %>%
  top_n(10) %>%
  ggplot(aes(x = Player_of_Match, y=awards, fill=Player_of_Match)) + geom_col(position="dodge") +
  labs(x="Player_of_match", y = "Awards" , title = "Top 10 Player Man of the Match") + coord_flip()

#Convert the date column

matches$day <- format(as.Date(matches$Date), "%d")
matches$month <- format(as.Date(matches$Date), "%m")
matches$year <- format(as.Date(matches$Date), "%Y")

#How many seasons got in the dataset

season_count <- length(unique(matches$year))
season_count

#Which team won by wickets or runs

Runs <- matches %>% filter(WonBy == "Runs") %>%
  select('WinningTeam', 'WonBy' )

Runs
count(Runs)
```



```r
Wickets <- matches %>% filter(WonBy == "Wickets") %>%
  select('WinningTeam', 'WonBy')

Wickets
count(Wickets)

#Which season has most number of matches

matches %>%
  group_by(year) %>%
  summarize(number_of_matches = n())

#Plot the season wise number of matches

matches %>%
  group_by(year) %>%
  summarize(number_of_matches=n(),.groups='drop') %>%
  ggplot(aes(x=year, y= number_of_matches, fill=year)) + geom_bar(stat = "identity") +
  labs(x="Season",y="Number ff Matches", title ="Season wise number of matches")

#In season from 2011 to 2013 the matches played are above 60..

#Which Team is dominating in certain cities

matches %>%
  filter(WonBy != 'No result') %>%
  group_by(WinningTeam,City) %>%
  summarize(wins = n(),.groups='drop') %>%
  arrange(desc(wins)) %>%
  top_n(10)
```

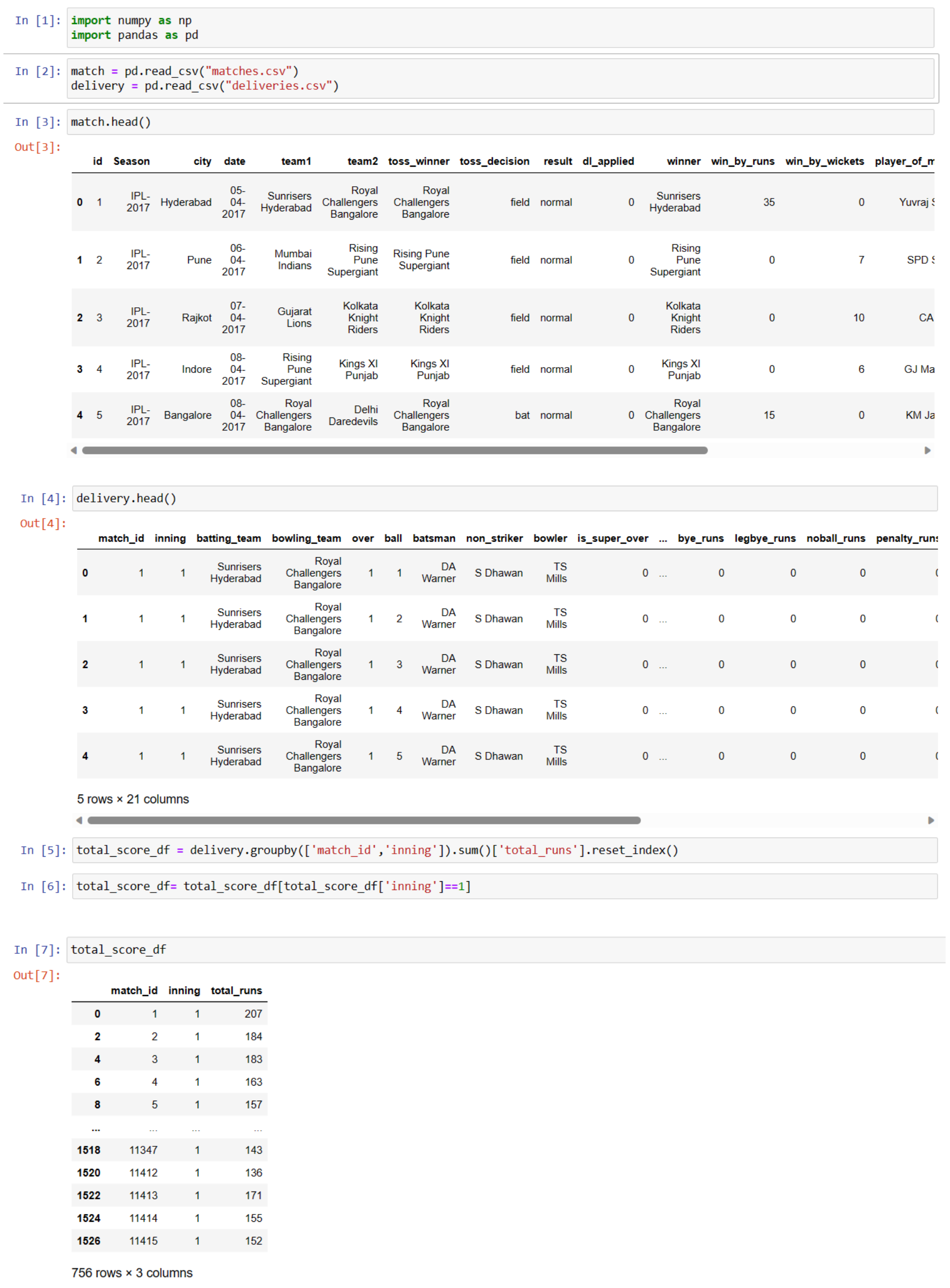


```
#Which team is not able to perform in the no-home locations

matches %>%
  filter(WonBy != 'No result') %>%
  group_by(WinningTeam,City) %>%
  summarize(wins = n(), .groups='drop') %>%
  arrange(City)


#Who's the best bolwer still dates

head(deliveries)
deliveries %>%
  group_by(bowler) %>%
  summarize(total_run = sum(total_run)) %>%
  arrange(total_run)

#Run scored and wickets lost in power play

head(matches)
head(deliveries)

#Combine both the dataset

data <- bind_rows(matches,deliveries)
head(data)

```



```
#Dataframe contains only powerplay data

power_play <- data %>%
  group_by(overs < 6)
head(power_play)

#Total powerplay runs, wickets

colnames(data)
```

<u>Result 2</u> – IPL Prediction



```
In [1]: import numpy as np
        import pandas as pd
```

```
In [2]: match = pd.read_csv("matches.csv")
        delivery = pd.read_csv("deliveries.csv")
```

```
In [3]: match.head()
Out[3]:
```

	id	Season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by_wickets	player_of_m
0	1	IPL-2017	Hyderabad	05-04-2017	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad	35	0	Yuvraj S
1	2	IPL-2017	Pune	06-04-2017	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0	Rising Pune Supergiant	0	7	SPD S
2	3	IPL-2017	Rajkot	07-04-2017	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	0	Kolkata Knight Riders	0	10	CA
3	4	IPL-2017	Indore	08-04-2017	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	0	Kings XI Punjab	0	6	GJ Ma
4	5	IPL-2017	Bangalore	08-04-2017	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	normal	0	Royal Challengers Bangalore	15	0	KM Ja

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

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

5 rows × 21 columns

```
In [5]: total_score_df = delivery.groupby(['match_id','inning']).sum()['total_runs'].reset_index()
```

```
In [6]: total_score_df= total_score_df[total_score_df['inning']==1]
```

```
In [7]: total_score_df
Out[7]:
```

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

Out[8]:

	id	Season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by_wickets	pl
0	1	IPL-2017	Hyderabad	05-04-2017	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad	35	0	
1	2	IPL-2017	Pune	06-04-2017	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0	Rising Pune Supergiant	0	7	
2	3	IPL-2017	Rajkot	07-04-2017	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	0	Kolkata Knight Riders	0	10	
3	4	IPL-2017	Indore	08-04-2017	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	0	Kings XI Punjab	0	6	
4	5	IPL-2017	Bangalore	08-04-2017	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	normal	0	Royal Challengers Bangalore	15	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
751	11347	IPL-2019	Mumbai	05-05-2019	Kolkata Knight Riders	Mumbai Indians	Mumbai Indians	field	normal	0	Mumbai Indians	0	9	
752	11412	IPL-2019	Chennai	07-05-2019	Chennai Super Kings	Mumbai Indians	Chennai Super Kings	bat	normal	0	Mumbai Indians	0	6	
753	11413	IPL-2019	Visakhapatnam	08-05-2019	Sunrisers Hyderabad	Delhi Capitals	Delhi Capitals	field	normal	0	Delhi Capitals	0	2	
754	11414	IPL-2019	Visakhapatnam	10-05-2019	Delhi Capitals	Chennai Super Kings	Chennai Super Kings	field	normal	0	Chennai Super Kings	0	6	

```
In [9]: match_df['team1'].unique()
```

```
Out[9]: 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 [10]: teams = [
             'Sunrisers Hyderabad',
             'Mumbai Indians',
             'Royal Challengers Bangalore',
             'Kolkata Knight Riders',
             'Kings XI Punjab',
             'Chennai Super Kings',
             'Rajasthan Royals',
             'Delhi Capitals'
         ]
```

```
In [11]: 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 [12]: match_df = match_df[match_df['team1'].isin(teams)]
         match_df = match_df[match_df['team2'].isin(teams)]
```

```
In [13]: match_df.shape
Out[13]: (641, 20)
```

```
In [14]: match_df = match_df[match_df['dl_applied'] == 0]
```

```
In [15]: match_df = match_df[['match_id','city','winner','total_runs']]
```

```
In [16]: delivery_df = match_df.merge(delivery,on='match_id')
```

```
In [17]: delivery_df = delivery_df[delivery_df['inning'] == 2]
```

```
In [18]: delivery_df
```

Out[18]:

	match_id	city	winner	total_runs_x	inning	batting_team	bowling_team	over	ball	batsman	...	bye_runs	legbye_runs	noball_runs	penalt
125	1	Hyderabad	Sunrisers Hyderabad	207	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	1	CH Gayle	...	0	0	0	
126	1	Hyderabad	Sunrisers Hyderabad	207	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	2	Mandeep Singh	...	0	0	0	
127	1	Hyderabad	Sunrisers Hyderabad	207	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	3	Mandeep Singh	...	0	0	0	
128	1	Hyderabad	Sunrisers Hyderabad	207	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	4	Mandeep Singh	...	0	0	0	
129	1	Hyderabad	Sunrisers Hyderabad	207	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	5	Mandeep Singh	...	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
149573	11415	Hyderabad	Mumbai Indians	152	2	Chennai Super Kings	Mumbai Indians	20	2	RA Jadeja	...	0	0	0	
149574	11415	Hyderabad	Mumbai Indians	152	2	Chennai Super Kings	Mumbai Indians	20	3	SR Watson	...	0	0	0	
149575	11415	Hyderabad	Mumbai Indians	152	2	Chennai Super Kings	Mumbai Indians	20	4	SR Watson	...	0	0	0	
149576	11415	Hyderabad	Mumbai	152	2	Chennai	Mumbai	20	5	SN		0	0	0	

```
In [19]: delivery_df['current_score'] = delivery_df.groupby('match_id').cumsum()['total_runs_y']
```

```
In [20]: delivery_df['runs_left'] = delivery_df['total_runs_x'] - delivery_df['current_score']
```

```
In [21]: delivery_df['balls_left'] = 126 - (delivery_df['over']*6 + delivery_df['ball'])
```

```
In [22]: delivery_df
```

Out[22]:

	match_id	city	winner	total_runs_x	inning	batting_team	bowling_team	over	ball	batsman	...	penalty_runs	batsman_runs	extra_runs	tc
125	1	Hyderabad	Sunrisers Hyderabad	207	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	1	CH Gayle	...	0	1	0	
126	1	Hyderabad	Sunrisers Hyderabad	207	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	2	Mandeep Singh	...	0	0	0	
127	1	Hyderabad	Sunrisers Hyderabad	207	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	3	Mandeep Singh	...	0	0	0	
128	1	Hyderabad	Sunrisers Hyderabad	207	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	4	Mandeep Singh	...	0	2	0	
129	1	Hyderabad	Sunrisers Hyderabad	207	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	5	Mandeep Singh	...	0	4	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
149573	11415	Hyderabad	Mumbai Indians	152	2	Chennai Super Kings	Mumbai Indians	20	2	RA Jadeja	...	0	1	0	
149574	11415	Hyderabad	Mumbai Indians	152	2	Chennai Super Kings	Mumbai Indians	20	3	SR Watson	...	0	2	0	
149575	11415	Hyderabad	Mumbai Indians	152	2	Chennai Super Kings	Mumbai Indians	20	4	SR Watson	...	0	1	0	

```
In [23]: delivery_df['player_dismissed'] = pd.to_numeric(delivery_df['player_dismissed'], errors='coerce')
         delivery_df['player_dismissed'] = delivery_df['player_dismissed'].fillna(0)
         delivery_df['player_dismissed'] = delivery_df['player_dismissed'].astype(int)
         delivery_df['player_dismissed'] = np.where(delivery_df['player_dismissed'] != 0, 1, 0)

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

Out[23]:

	match_id	city	winner	total_runs_x	inning	batting_team	bowling_team	over	ball	batsman	...	batsman_runs	extra_runs	total_runs_y	pl
125	1	Hyderabad	Sunrisers Hyderabad	207	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	1	CH Gayle	...	1	0	1	
126	1	Hyderabad	Sunrisers Hyderabad	207	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	2	Mandeep Singh	...	0	0	0	
127	1	Hyderabad	Sunrisers Hyderabad	207	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	3	Mandeep Singh	...	0	0	0	
128	1	Hyderabad	Sunrisers Hyderabad	207	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	4	Mandeep Singh	...	2	0	2	
129	1	Hyderabad	Sunrisers Hyderabad	207	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	5	Mandeep Singh	...	4	0	4	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
149573	11415	Hyderabad	Mumbai Indians	152	2	Chennai Super Kings	Mumbai Indians	20	2	RA Jadeja	...	1	0	1	
149574	11415	Hyderabad	Mumbai Indians	152	2	Chennai Super Kings	Mumbai Indians	20	3	SR Watson	...	2	0	2	
149575	11415	Hyderabad	Mumbai Indians	152	2	Chennai Super Kings	Mumbai Indians	20	4	SR Watson	...	1	0	1	
149576	11415	Hyderabad	Mumbai Indians	152	2	Chennai Super Kings	Mumbai Indians	20	5	SN Thakur	...	2	0	2	
149577	11415	Hyderabad	Mumbai Indians	152	2	Chennai Super Kings	Mumbai Indians	20	6	SN Thakur	...	0	0	0	

```
In [24]: delivery_df.head()
```

Out[24]:

	match_id	city	winner	total_runs_x	inning	batting_team	bowling_team	over	ball	batsman	...	batsman_runs	extra_runs	total_runs_y	playe
125	1	Hyderabad	Sunrisers Hyderabad	207	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	1	CH Gayle	...	1	0	1	
126	1	Hyderabad	Sunrisers Hyderabad	207	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	2	Mandeep Singh	...	0	0	0	
127	1	Hyderabad	Sunrisers Hyderabad	207	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	3	Mandeep Singh	...	0	0	0	
128	1	Hyderabad	Sunrisers Hyderabad	207	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	4	Mandeep Singh	...	2	0	2	
129	1	Hyderabad	Sunrisers Hyderabad	207	2	Royal Challengers Bangalore	Sunrisers Hyderabad	1	5	Mandeep Singh	...	4	0	4	

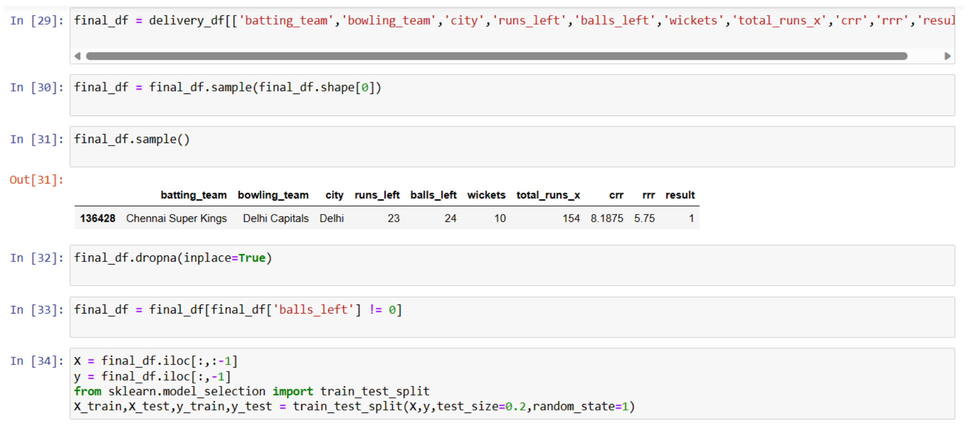
5 rows × 28 columns

```
In [25]: # crr = runs/overs
         delivery_df['crr'] = (delivery_df['current_score']*6)/(120 - delivery_df['balls_left'])
```

```
In [26]: delivery_df['rrr'] = (delivery_df['runs_left']*6)/delivery_df['balls_left']
```

```
In [27]: def result(row):
             return 1 if row['batting_team'] == row['winner'] else 0
```

```
In [28]: delivery_df['result'] = delivery_df.apply(result,axis=1)
```



```
In [29]: final_df = delivery_df[['batting_team','bowling_team','city','runs_left','balls_left','wickets','total_runs_x','crr','rrr','resul
```

```
In [30]: final_df = final_df.sample(final_df.shape[0])
```

```
In [31]: final_df.sample()
```

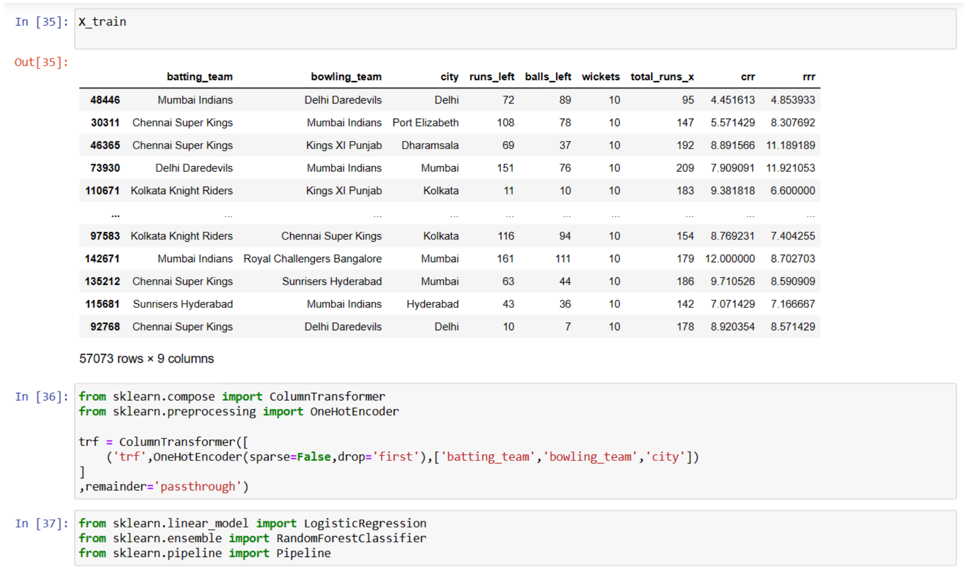
Out[31]:

	batting_team	bowling_team	city	runs_left	balls_left	wickets	total_runs_x	crr	rrr	result
136428	Chennai Super Kings	Delhi Capitals	Delhi	23	24	10	154	8.1875	5.75	1

```
In [32]: final_df.dropna(inplace=True)
```

```
In [33]: final_df = final_df[final_df['balls_left'] != 0]
```

```
In [34]: X = final_df.iloc[:,:-1]
         y = final_df.iloc[:,-1]
         from sklearn.model_selection import train_test_split
         X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=1)
```



```
In [35]: X_train
```

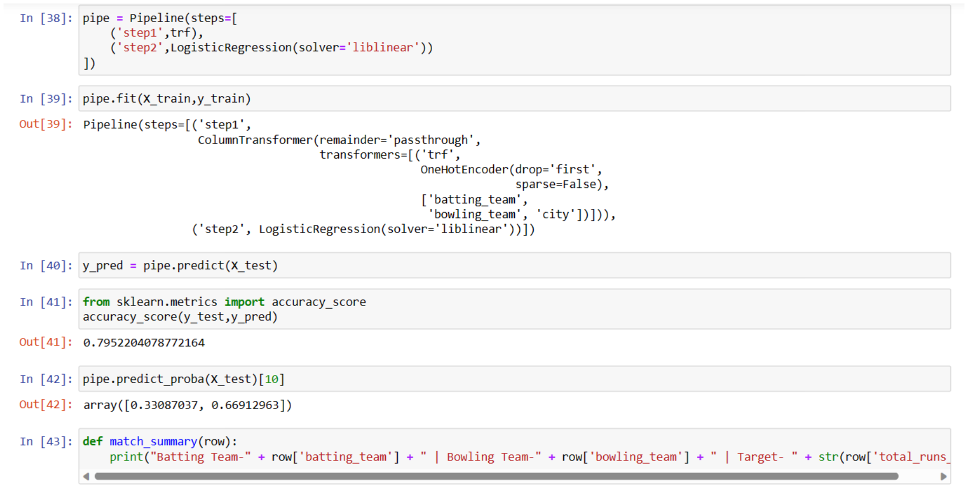
Out[35]:

	batting_team	bowling_team	city	runs_left	balls_left	wickets	total_runs_x	crr	rrr
48446	Mumbai Indians	Delhi Daredevils	Delhi	72	89	10	95	4.451613	4.853933
30311	Chennai Super Kings	Mumbai Indians	Port Elizabeth	108	78	10	147	5.571429	8.307692
46365	Chennai Super Kings	Kings XI Punjab	Dharamsala	69	37	10	192	8.891566	11.189189
73930	Delhi Daredevils	Mumbai Indians	Mumbai	151	76	10	209	7.909091	11.921053
110671	Kolkata Knight Riders	Kings XI Punjab	Kolkata	11	10	10	183	9.381818	6.600000
...	...	...	...	...	...	...	...	...	...
97583	Kolkata Knight Riders	Chennai Super Kings	Kolkata	116	94	10	154	8.769231	7.404255
142671	Mumbai Indians	Royal Challengers Bangalore	Mumbai	161	111	10	179	12.000000	8.702703
135212	Chennai Super Kings	Sunrisers Hyderabad	Mumbai	63	44	10	186	9.710526	8.590909
115681	Sunrisers Hyderabad	Mumbai Indians	Hyderabad	43	36	10	142	7.071429	7.166667
92768	Chennai Super Kings	Delhi Daredevils	Delhi	10	7	10	178	8.920354	8.571429

57073 rows × 9 columns

```
In [36]: 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 [37]: from sklearn.linear_model import LogisticRegression
         from sklearn.ensemble import RandomForestClassifier
         from sklearn.pipeline import Pipeline
```



```
In [38]: pipe = Pipeline(steps=[
             ('step1',trf),
             ('step2',LogisticRegression(solver='liblinear'))
         ])
```

```
In [39]: pipe.fit(X_train,y_train)

Out[39]: Pipeline(steps=[('step1',
                          ColumnTransformer(remainder='passthrough',
                                            transformers=[('trf',
                                                           OneHotEncoder(drop='first',
                                                                         sparse=False),
                                                           ['batting_team',
                                                            'bowling_team', 'city'])])),
                         ('step2', LogisticRegression(solver='liblinear'))])
```

```
In [40]: y_pred = pipe.predict(X_test)
```

```
In [41]: from sklearn.metrics import accuracy_score
         accuracy_score(y_test,y_pred)

Out[41]: 0.7952204078772164
```

```
In [42]: pipe.predict_proba(X_test)[10]

Out[42]: array([0.33087037, 0.66912963])
```

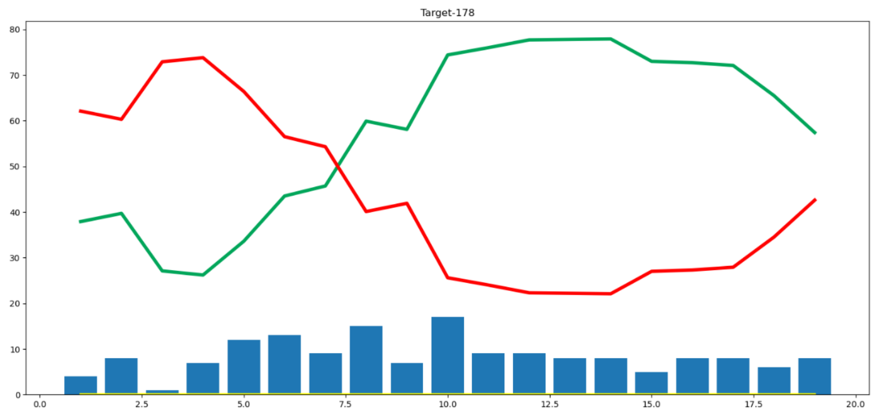
```
In [43]: def match_summary(row):
             print("Batting Team-" + row['batting_team'] + " | Bowling Team-" + row['bowling_team'] + " | Target- " + str(row['total_runs_
In [44]: def match_progression(x_df,match_id,pipe):
             match = x_df[x_df['match_id'] == match_id]
             match = match[(match['ball'] == 6)]
             temp_df = match[['batting_team','bowling_team','city','runs_left','balls_left','wickets','total_runs_x','crr','rrr']].dropna(
             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
```

```
In [45]: temp_df,target = match_progression(delivery_df,74,pipe)
         temp_df

         Target- 178
```

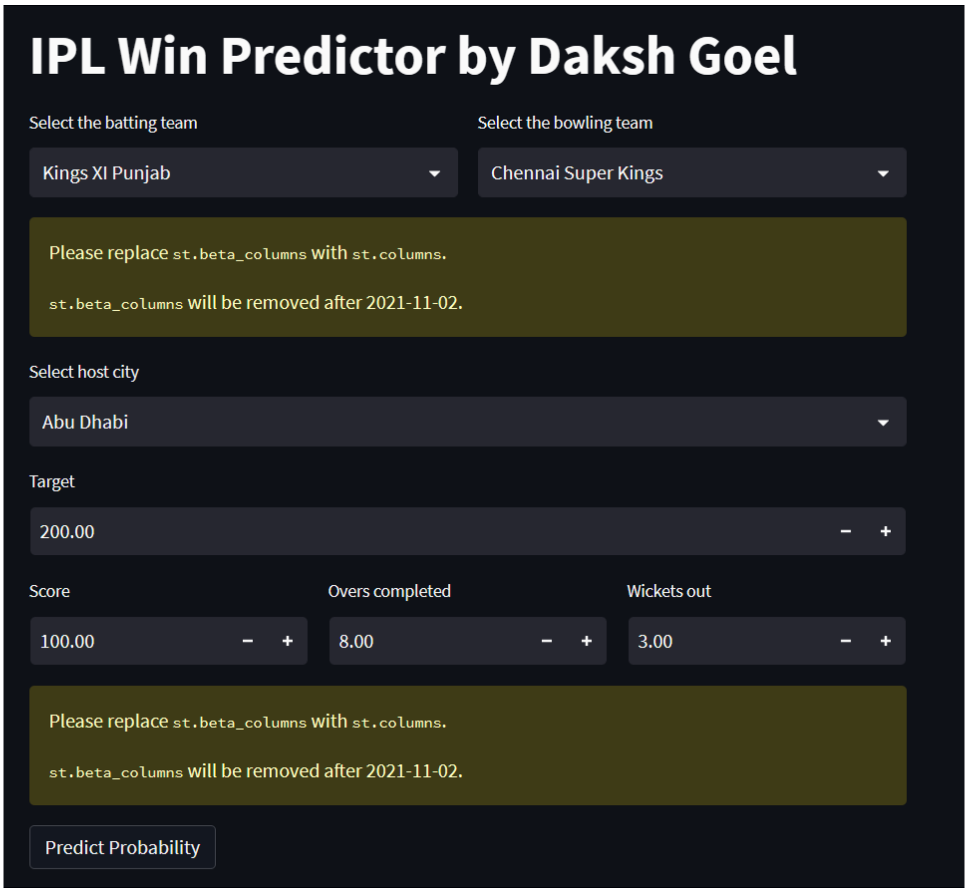
Out[45]:

	end_of_over	runs_after_over	wickets_in_over	lose	win
10459	1	4	0	62.1	37.9
10467	2	8	0	60.3	39.7
10473	3	1	0	72.9	27.1
10479	4	7	0	73.8	26.2
10485	5	12	0	66.4	33.6
10491	6	13	0	56.5	43.5
10497	7	9	0	54.3	45.7
10505	8	15	0	40.1	59.9
10511	9	7	0	41.9	58.1
10518	10	17	0	25.6	74.4
10524	11	9	0	24.0	76.0
10530	12	9	0	22.3	77.7
10536	13	8	0	22.2	77.8
10542	14	8	0	22.1	77.9
10548	15	5	0	27.0	73.0
10555	16	8	0	27.3	72.7
10561	17	8	0	27.9	72.1
10567	18	6	0	34.5	65.5
10573	19	8	0	42.6	57.4

```
In [46]: import matplotlib.pyplot as plt
         plt.figure(figsize=(18,8))
         plt.plot(temp_df['end_of_over'],temp_df['wickets_in_over'],color='yellow',linewidth=3)
         plt.plot(temp_df['end_of_over'],temp_df['win'],color='#00a65a',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[46]: Text(0.5, 1.0, 'Target-178')
```



```
In [47]: teams

Out[47]: ['Sunrisers Hyderabad',
          'Mumbai Indians',
          'Royal Challengers Bangalore',
          'Kolkata Knight Riders',
          'Kings XI Punjab',
          'Chennai Super Kings',
          'Rajasthan Royals',
          'Delhi Capitals']

In [48]: delivery_df['city'].unique()

Out[48]: array(['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', nan, 'Mohali', 'Bengaluru'], dtype=object)

In [49]: import pickle
         pickle.dump(pipe,open('pipe.pkl','wb'))
```



IPL Win Predictor by Daksh Goel

Select the batting team: Kings XI Punjab

Select the bowling team: Chennai Super Kings

Please replace st.beta_columns with st.columns.

st.beta_columns will be removed after 2021-11-02.

Select host city: Abu Dhabi

Target: 200.00

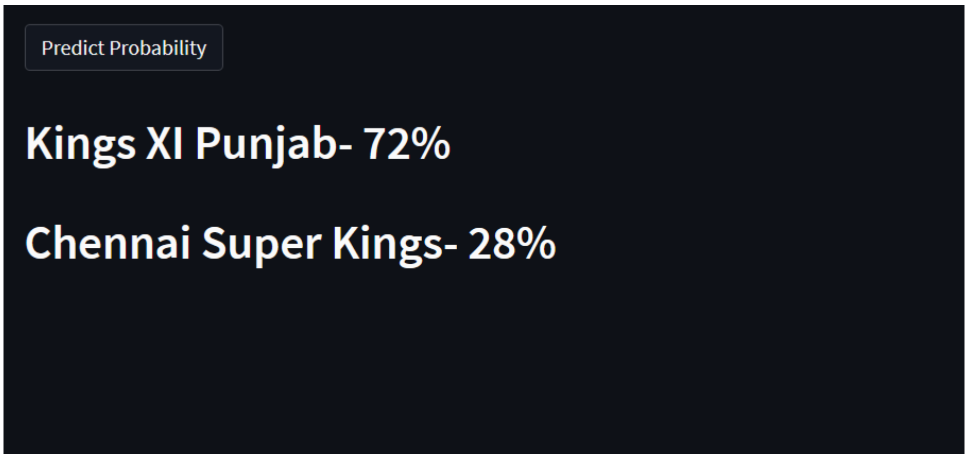
Score: 100.00

Overs completed: 8.00

Wickets out: 3.00

Please replace st.beta_columns with st.columns.

st.beta_columns will be removed after 2021-11-02.



Kings XI Punjab- 72%

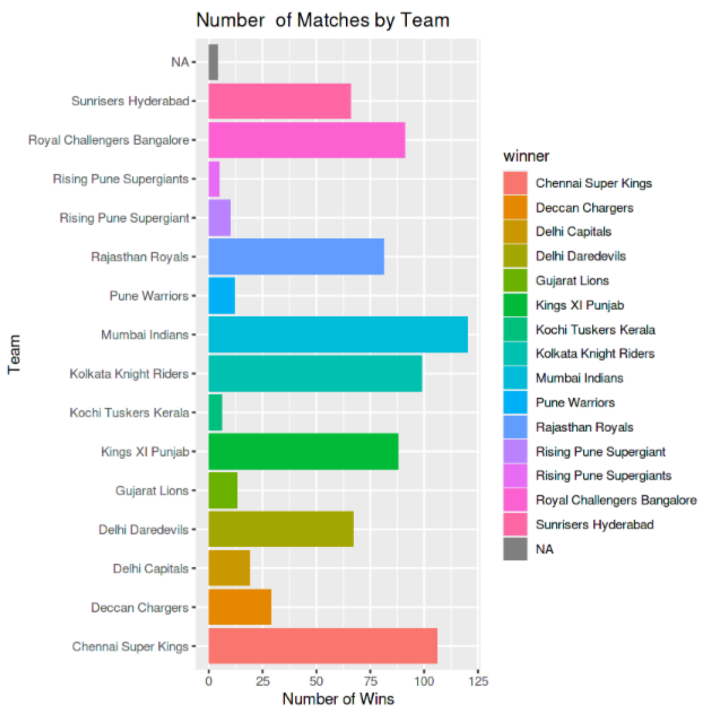
Chennai Super Kings- 28%

ANALYTICAL SOLUTION

In the first half of the code, we have done some analysis on the code to make it more understandable and readable. The two datasets matches and deliveries have been analysed and some graphs have been made from it.

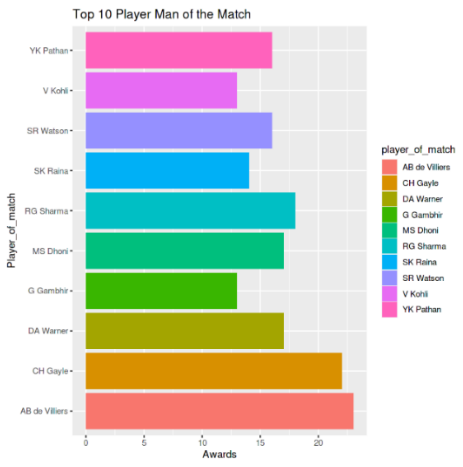
1. Which team won how many times

Winner Vs Count

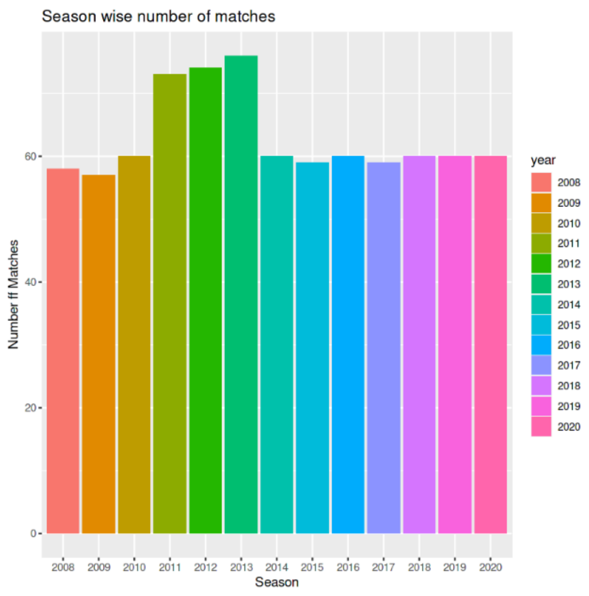


2. Which player has received man of the match most times

Player Vs Title Count



3. Season wise number of Matches



In the second part of code i.e., IPL WIN PREDICTOR some logical mathematical equations are used, the equations are:

1. Runs Left = Total runs – current score [For every ball]
2. Balls Left = 126 – (current over) *6 + ball   [As the game proceeds]
3. Cumulative sum of wickets after every ball for balls with same match id and inning
4. Wickets left = 10 – wickets

The IPL Predictor code involves several mathematical expressions used for building and training a machine learning model. Here are some of the key mathematical expressions used in this code:

1. Linear Regression Model:

The linear regression model used in this code involves the following mathematical expression:

$y = b0 + b1x1 + b2x2 + ... + bn*xn$

where,

y = dependent variable (target variable)

b0 = intercept (bias)

b1, b2, ..., bn = coefficients of independent variables (features)

x1, x2, ..., xn = independent variables (features)

2. Mean Squared Error (MSE):

   Mean Squared Error is a metric used to evaluate the performance of a regression model. It is calculated as the average of the squared differences between the actual values and the predicted values. The mathematical expression for MSE is as follows:

   $MSE = (1/n) * \sum (y - \hat{y})^2$

   where,

   n = number of data points

   y = actual value

   $\hat{y}$ = predicted value

3. Gradient Descent

   Gradient descent is an optimization algorithm used to update the coefficients of a linear regression model during the training process. It involves the following mathematical expression:

   $\beta = \beta - \alpha * (\partial J/\partial \beta)$

   where,

   β = coefficient

   α = learning rate

   J = cost function

   $\partial J/\partial \beta$ = partial derivative of cost function with respect to β

BRIEF OVERVIEW OF THE CODE FOR IPL PREDICTOR

This code is a Python program for predicting the winner of a cricket match in the Indian Premier League (IPL) based on the remaining runs, balls and wickets of the team batting in the second innings. The program imports two CSV files ('matches.csv' and 'deliveries.csv') containing information about IPL matches and ball-by-ball data of each match.

The program uses pandas library to manipulate dataframes and extract required information from the data. It then creates a new dataframe 'final_df' containing features such as the remaining runs, balls, wickets, current run rate, required run rate, and the city of the match. The 'final_df' dataframe is then split into training and testing sets using the 'train_test_split' function from the 'sklearn. model selection' module.

The 'ColumnTransformer' function from the 'sklearn. compose' module is used to one-hot encode the categorical features ('batting_team', 'bowling_team', 'city') in the training and testing sets. The transformed datasets are then passed through a logistic regression model from the 'sklearn. linear model' module using a 'Pipeline' function from the 'sklearn. pipeline' module.

The 'match_summary' function prints the details of the batting and bowling teams and the target runs to chase. The 'match_progression' function takes in the 'final_df' dataframe, match_id and the trained pipeline model as arguments, and outputs a dataframe with columns such as runs_left, balls_left++, wickets, current run rate, required run rate, and the probability of winning and losing the match after each over.

Overall, the program provides a basic framework for predicting the winner of an IPL match based on the remaining resources of the team batting in the second innings, using a logistic regression model. However, there is scope for improvement by incorporating more relevant features and using more sophisticated machine learning models.

Conclusion:

we can draw several conclusions:

- Historical Data Analysis: By examining various factors such as team performance, player statistics, pitch conditions, weather conditions, and head-to-head records, we gained valuable insights into the patterns and trends in IPL matches over the years.
- Team Performance Trends: The analysis of team performance across multiple seasons revealed distinct trends. Some teams consistently performed well, while others experienced fluctuations in their performance. Understanding these trends can provide valuable information for predicting future team performances.
- IPL Predictor: The development of an IPL predictor based on the first innings involved utilizing historical data, statistical models, and machine learning techniques. The predictor incorporated features such as team strength, player form, head-to-head records, and venue conditions to make predictions about the outcome of the first innings.

In conclusion, IPL data analysis provides valuable insights into team and player performances, enabling the identification of trends and patterns. The development of an IPL predictor based on these analyses offers a useful tool for predicting match outcomes.

Reference:

- IPL Dataset (2008-2022)
- IPL Prediction Report