VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Win and Score Predictor for IPL

A PROJECT REPORT

Submitted by

Hritwik Bhaumik – 17BCE0548

Course Code: CSE3020

Course Title: Data Visualization

In partial fulfilment for the award of the degree of

B. Tech

in

Computer Science and Engineering

Under the guidance of

Dr. Rajkumar R

VIT, Vellore.

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely privileged to have got this all along the completion of our project. All that we have done is only due to such supervision and assistance and we would not forget to thank them.

We must

first of all, express our heartiest gratitude to respected **Prof.**

Rajkumar R (SCOPE) for providing us all

guidance to complete project.

Last but not the least we express our sincere thanks to the institute VIT, Vellore

for providing such a platform for implementing the ideas in our mind.

Vellore Institute of Technology, Vellore

Certificate

This is to certify that the project report entitled **Win and Score Predictor for IPL**,

submitted to the Department of Computer Science and Engineering, Vellore Institute of Technology, Vellore in partial

fulfillment for the award of the degree of **Bachelor of Technology in Computer Science** and

Engineering, is a record of bona fide work carried out by **Hritwik Bhaumik – 17BCE0548** under my supervision and guidance.

All help received by them from various sources have been duly acknowledged.

Rajkumar R.

Place: Vellore

Date: 6 Nov, 2020

Abstract:

Data Visualization is a used a lot in major sports to determine the outcome of the matches and is also used in fantasy sports, betting and simulation. However with cricket technology has not been utilized as much as in other major sports. This is an attempt to predict the matches of this sport in its shortest and most popular format – T20. This can be used for analytics, fantasy teams, simulation etc.

INDEX

List of figures

Chapter-1

Introduction

Data Visualization is a used a lot in major sports to determine the outcome of the matches and is also used in fantasy sports, betting and simulation. However with cricket technology has not been utilized as much as in other major sports. This is an attempt to predict the matches of this sport in its shortest and most popular format – T20. This can be used for analytics, fantasy teams, simulation etc.

With this project we aim to achieve

- A way to predict cricket matches instantly at that point of the games using previously collected data.
- Aim of making cricket more technologically accessible and encouraging more research in this field.
- Helping teams and analyst to get the prediction of the games beforehand.

Chapter-2

Review of literature

The reference paper which we are taking takes into account ODI games and predicts the result of these games. These days t20 cricket is quickly becoming the new format and is widely spreading everywhere with over 110 teams by International Cricket Council being recognized as full t20 teams(as of 2019).

We believe our prediction and analysis will be suitable to this format of the game as well with minor modifications to the ODI format because of the frenetic and faster pace of the t20 game. We'll take the example of the biggest domestic t20 league in cricket the Indian Premier League (IPL) which has been running now for 11 years and is continuing onto its 12^{th} season.

Chapter-3

Methodology

Data Collection – Data was initially downloaded from cricsheet.org, converted into csv and then scrapped.

Data Pre-processing – Null values were found out and replaced with default values – e.g. – Cities which were "Null" were replaced by Dubai.

Packages used – We have used the Python language with Jupyter Notebook.
Some important packages used were :-

Sci-kit learn package

Seaborn package

Matplotlib

Pandas

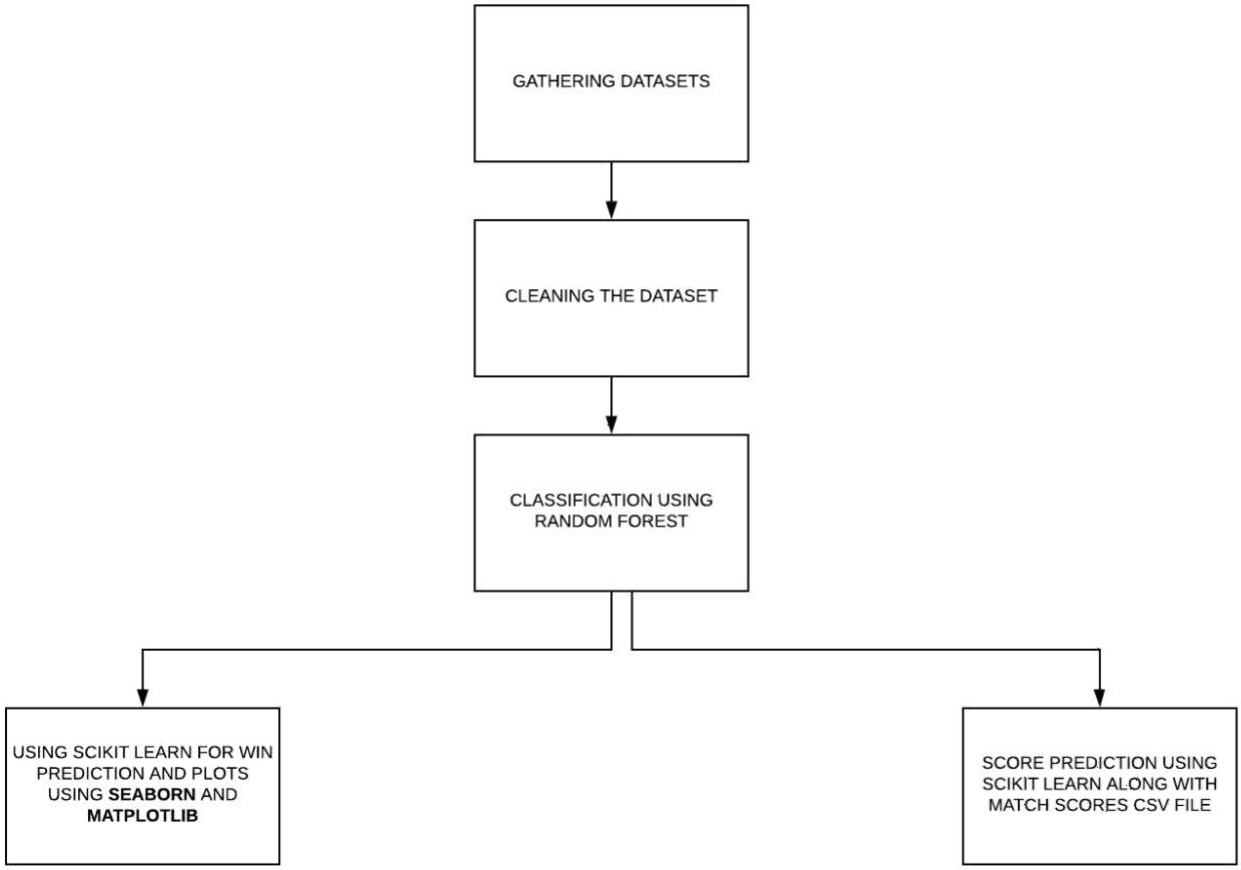
The user will enter the team names, the team entered first will be the home team.

The user will also enter the toss decision and venue if needed.

Using data visualization we will predict the winner of the match using the following methods –

a) First pre-process and clean the data set
b) Assign initials to the team full names
c) Find out the ratio of toss won and match won
d) Find out the correlation between home team and venue win percentage
e) Use labelencoder() to mark the labels needed for the three main columns.
f) Use decision tree and random forest to predict the winner

Following figure shows the methodology used for IPL-WASP analysis



Chapter-4

Implementation Technique

Building the model –

- Using the scikitlearn library.
- Labeling the objects using LabelEncoder()
- Using other important modules like - :
  I. Logistic Regression II.
  Kfold cross validation III.
  RandomForest Classiifier IV.
  Decision Tree

- **Logistic Regression** - Like all **regression** analyses, the **logistic regression** is a predictive analysis. **Logistic regression** is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

  In simple linear regression, we predict the dependent variable value by using the one or more independent variables. The variable we are predicting is called the standard variable and is mentioned to as Y. The inconstant we are founding our predictions on is called the predictor variable and is stated to as X. When we are predicting only one variable this is called simple linear regression. In simple linear regression, the predictions of Y depends on the values of X. form a straight line. Line a regression consists of verdict the best-fitting conventional line through the points. The best fitting line is called a regression line.

- **K fold cross validation** - Cross-validation is a statistical method used to estimate the skill of machine learning models.
  It is commonly used in applied machine learning to compare and select a model for a given predictive modelling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods.

- **Random Forest Classifier** -A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default).

  This classifier fits n number of decision tree classifiers on many sub-samples of the dataset, controls the over-fitting of data and advances predictive precision by averaging. This classifier has been used to rise a attired precision over singular meaning trees obtained. If y is an attribute to be forecast and X are the attributes used for prediction

  Random Forest pseudocode:

  1. Randomly select "k" features from total "m" features.

     . Where k << m
  2. Calculate node "d" among the "k" features by using best split.
  3. Divide the node into daughter nodes by using the best division.
  4. Repeat the steps from 1 to 3 until we got "l" number of nodes.
  5. By repeating 1 to 4 steps build the forest for "n" number times to create "n" number of trees.

- **Decision Tree** - A **decision tree** is a **decision** support tool that uses a **tree**-like model of **decisions** and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

  The goal of this classifier is to make a model that forecasts the value of a target variable by the information of simple decision rules related from the data features. A deeper tree designates complex decision rules and that makes a improved fitted model. If y is an attribute to be predicted and X are the attributes used for approximation

  ❖ Step-1 A decision tree is built top-down from a root node and comprises dividing the data into subsets that comprise cases with similar values.

  ❖ Step-2 Standard Deviation Reduction

The standard deviation reduction is based on the reduction in standard deviation after a dataset is s on an attribute

(1) We measure the standard deviation of the target.

(2) The dataset is then split on the dissimilar attributes. We measure standard deviation for each branch. The consequent standard deviation is deducted from the standard deviation previously. The value we got after deduction is standard deviation

- Step-3 The node with highest standard deviation referred as decision node
- Step 4: By help of the calculated values we differentiate the several datasets. This procedure is run recursively on the non-leaf branches, until and unless all the data is handled.

Chapter-5

Dataset Collection

Dataset was taken from cricsheet.org

Files were initially in YAML format, they were converted to .csv for data scrapping.

There are two datasets

Matches.csv – Has the data of each match played in the IPL, team names, winner, venue and toss decision.

Deliviries.csv – Has the ball by ball information of each match in the IPL up to 2016.

Ipldataset.csv – Has details of all matches with scores

included. **Description of the dataset –**

id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by_wick	player_of_m	venue	umpire1	umpire2	umpire3
1	2017	Hyderabad	05/04/17	Sunrisers Hy	Royal Challe	Royal Challe	field	normal	0	Sunrisers Hy	35	0	Yuvraj Singh	Rajiv Gandhi	AY Dandekar	NJ Llong	
2	2017	Pune	06/04/17	Mumbai Indi	Rising Pune S	Rising Pune S	field	normal	0	Rising Pune S	0	7	SPD Smith	Maharashtra	A Nand Kisho	S Ravi	
3	2017	Rajkot	07/04/17	Gujarat Lion	Kolkata Knigl	Kolkata Knigl	field	normal	0	Kolkata Knigl	0	10	CA Lynn	Saurashtra C	Nitin Menon	CK Nandan	
4	2017	Indore	08/04/17	Rising Pune S	Kings XI Punj	Kings XI Punj	field	normal	0	Kings XI Punj	0	6	GJ Maxwell	Holkar Cricke	AK Chaudhar	C Shamshuddin	
5	2017	Bangalore	08/04/17	Royal Challe	Delhi Darede	Royal Challe	bat	normal	0	Royal Challe	15	0	KM Jadhav	M Chinnaswamy Stadium			
6	2017	Hyderabad	09/04/17	Gujarat Lion	Sunrisers Hy	Sunrisers Hy	field	normal	0	Sunrisers Hy	0	9	Rashid Khan	Rajiv Gandhi	A Deshmukh	NJ Llong	
7	2017	Mumbai	09/04/17	Kolkata Knigl	Mumbai Indi	Mumbai Indi	field	normal	0	Mumbai Indi	0	4	N Rana	Wankhede S	Nitin Menon	CK Nandan	
8	2017	Indore	10/04/17	Royal Challe	Kings XI Punj	Royal Challe	bat	normal	0	Kings XI Punj	0	8	AR Patel	Holkar Cricke	AK Chaudhar	C Shamshuddin	
9	2017	Pune	11/04/17	Delhi Darede	Rising Pune S	Rising Pune S	field	normal	0	Delhi Darede	97	0	SV Samson	Maharashtra	AY Dandekar	S Ravi	
10	2017	Mumbai	12/04/17	Sunrisers Hy	Mumbai Indi	Mumbai Indi	field	normal	0	Mumbai Indi	0	4	JJ Bumrah	Wankhede S	Nitin Menon	CK Nandan	
11	2017	Kolkata	13/04/17	Kings XI Punj	Kolkata Knigl	Kolkata Knigl	field	normal	0	Kolkata Knigl	0	8	SP Narine	Eden Garden	A Deshmukh	NJ Llong	
12	2017	Bangalore	14/04/17	Royal Challe	Mumbai Indi	Mumbai Indi	field	normal	0	Mumbai Indi	0	4	KA Pollard	M Chinnaswa	KN Ananthap	AK Chaudhary	
13	2017	Rajkot	14/04/17	Rising Pune S	Gujarat Lion	Gujarat Lion	field	normal	0	Gujarat Lion	0	7	AJ Tye	Saurashtra C	A Nand Kisho	S Ravi	
14	2017	Kolkata	15/04/17	Kolkata Knigl	Sunrisers Hy	Sunrisers Hy	field	normal	0	Kolkata Knigl	17	0	RV Uthappa	Eden Garden	AY Dandekar	NJ Llong	
15	2017	Delhi	15/04/17	Delhi Darede	Kings XI Punj	Delhi Darede	bat	normal	0	Delhi Darede	51	0	CJ Anderson	Feroz Shah K	YC Barde	Nitin Menon	
16	2017	Mumbai	16/04/17	Gujarat Lion	Mumbai Indi	Mumbai Indi	field	normal	0	Mumbai Indi	0	6	N Rana	Wankhede S	A Nand Kisho	S Ravi	
17	2017	Bangalore	16/04/17	Rising Pune S	Royal Challe	Royal Challe	field	normal	0	Rising Pune S	27	0	BA Stokes	M Chinnaswa	KN Ananthap	C Shamshuddin	
18	2017	Delhi	17/04/17	Delhi Darede	Kolkata Knigl	Delhi Darede	bat	normal	0	Kolkata Knigl	0	4	NM Coulter-I	Feroz Shah K	Nitin Menon	CK Nandan	
19	2017	Hyderabad	17/04/17	Sunrisers Hy	Kings XI Punj	Kings XI Punj	field	normal	0	Sunrisers Hy	5	0	B Kumar	Rajiv Gandhi	AY Dandekar	A Deshmukh	
20	2017	Rajkot	18/04/17	Royal Challe	Gujarat Lion	Gujarat Lion	field	normal	0	Royal Challe	21	0	CH Gayle	Saurashtra C	S Ravi	VK Sharma	
21	2017	Hyderabad	19/04/17	Sunrisers Hy	Delhi Darede	Sunrisers Hy	bat	normal	0	Sunrisers Hy	15	0	KS Williamso	Rajiv Gandhi	CB Gaffaney	NJ Llong	
22	2017	Indore	20/04/17	Kings XI Punj	Mumbai Indi	Mumbai Indi	field	normal	0	Mumbai Indi	0	8	JC Buttler	Holkar Cricke	M Erasmus	C Shamshuddin	
23	2017	Kolkata	21/04/17	Kolkata Knigl	Gujarat Lion	Gujarat Lion	field	normal	0	Gujarat Lion	0	4	SK Raina	Eden Garden	CB Gaffaney	Nitin Menon	
24	2017	Mumbai	22/04/17	Mumbai Indi	Delhi Darede	Delhi Darede	field	normal	0	Mumbai Indi	14	0	MJ McClenag	Wankhede S	A Nand Kisho	S Ravi	
25	2017	Pune	22/04/17	Sunrisers Hy	Rising Pune S	Rising Pune S	field	normal	0	Rising Pune S	0	6	MS Dhoni	Maharashtra	AY Dandekar	A Deshmukh	
26	2017	Rajkot	23/04/17	Kings XI Punj	Gujarat Lion	Gujarat Lion	field	normal	0	Kings XI Punj	26	0	HM Amla	Saurashtra C	AK Chaudhar	M Erasmus	
27	2017	Kolkata	23/04/17	Kolkata Knigl	Royal Challe	Royal Challe	field	normal	0	Kolkata Knigl	82	0	NM Coulter-I	Eden Garden	CB Gaffaney	CK Nandan	
28	2017	Mumbai	24/04/17	Rising Pune S	Mumbai Indi	Mumbai Indi	field	normal	0	Rising Pune S	3	0	BA Stokes	Wankhede S	A Nand Kisho	S Ravi	
29	2017	Pune	26/04/17	Rising Pune S	Kolkata Knigl	Kolkata Knigl	field	normal	0	Kolkata Knigl	0	7	RV Uthappa	Maharashtra	AY Dandekar	NJ Llong	
30	2017	Bangalore	27/04/17	Royal Challe	Gujarat Lion	Gujarat Lion	field	normal	0	Gujarat Lion	0	7	AJ Tye	M Chinnaswa	AK Chaudhar	C Shamshuddin	
31	2017	Kolkata	28/04/17	Delhi Darede	Kolkata Knigl	Kolkata Knigl	field	normal	0	Kolkata Knigl	0	7	G Gambhir	Eden Garden	NJ Llong	S Ravi	
32	2017	Chandigarh	28/04/17	Sunrisers Hy	Kings XI Punj	Kings XI Punj	field	normal	0	Sunrisers Hy	26	0	Rashid Khan	Punjab Cricke	Nitin Menon	CK Nandan	

Matches.csv

match_id	series_id	match_detail	result	Team1	Score1	Team2	Score2	date	venue	round	home	away	winner
335982	313494	1st match:RC	KKR won by	KKR	222	RCB	82	18/04/08	Bangalore	1st match	RCB	KKR	KKR
335983	313494	2nd match:K	CSK won by	CSK	240	KXIP	207	19/04/08	Mohali	2nd match	KXIP	CSK	CSK
335984	313494	3rd match:D	DD won by 9	RR	129	DD	132	19/04/08	Delhi	3rd match	DD	RR	DD
335986	313494	4th match:KI	KKR won by	DC	110	KKR	112	20/04/08	Kolkata	4th match	KKR	DC	KKR
335985	313494	5th match:M	RCB won by	MI	165	RCB	166	20/04/08	Mumbai	5th match	MI	RCB	RCB
335987	313494	6th match:RI	RR won by 6	KXIP	166	RR	168	21/04/08	Jaipur	6th match	RR	KXIP	RR
335988	313494	7th match:D	DD won by 9	DC	142	DD	143	22/04/08	Hyderabad (	7th match	DC	DD	DD
335989	313494	8th match:C	CSK won by	CSK	208	MI	202	23/04/08	Chennai	8th match	CSK	MI	CSK
335990	313494	9th match:D	RR won by 3	DC	214	RR	217	24/04/08	Hyderabad (	9th match	DC	RR	RR
335991	313494	10th match:I	KXIP won by	KXIP	182	MI	116	25/04/08	Mohali	10th match	KXIP	MI	KXIP
335993	313494	11th match:(	CSK won by	KKR	147	CSK	152	26/04/08	Chennai	11th match	CSK	KKR	CSK
335992	313494	12th match:I	RR won by 7	RCB	135	RR	138	26/04/08	Bangalore	12th match	RCB	RR	RR
335995	313494	13th match:I	KXIP won by	DD	158	KXIP	162	27/04/08	Mohali	13th match	KXIP	DD	KXIP
335994	313494	14th match:I	DC won by 1	MI	154	DC	155	27/04/08	Mumbai	14th match	MI	DC	DC
335996	313494	15th match:I	CSK won by	CSK	178	RCB	165	28/04/08	Bangalore	15th match	RCB	CSK	CSK
335997	313494	16th match:I	MI won by 7	KKR	137	MI	138	29/04/08	Kolkata	16th match	KKR	MI	MI
335998	313494	17th match:I	DD won by 1	DD	191	RCB	181	30/04/08	Delhi	17th match	DD	RCB	DD
336000	313494	18th match:I	RR won by 4	RR	196	KKR	151	01/05/08	Jaipur	18th match	RR	KKR	RR
335999	313494	19th match:I	KXIP won by	DC	164	KXIP	167	01/05/08	Hyderabad (	19th match	DC	KXIP	KXIP
336001	313494	20th match:(	DD won by 8	CSK	169	DD	172	02/05/08	Chennai	20th match	CSK	DD	DD
336034	313494	21st match:F	RCB won by	RCB	156	DC	153	03/05/08	Bangalore	21st match	RCB	DC	RCB

392181	374163	1st match:CS	MI won by 1	MI	165	CSK	146	18/04/09	Cape Town	1st match	CSK	MI	MI
392182	374163	2nd match:R	RCB won by	RCB	133	RR	58	18/04/09	Cape Town	2nd match	RCB	RR	RCB
392183	374163	3rd match:D	DD won by 1	KXIP	104	DD	58	19/04/09	Cape Town	3rd match	DD	KXIP	DD
392184	374163	4th match:D	DC won by 8	KKR	101	DC	104	19/04/09	Cape Town	4th match	DC	KKR	DC
392185	374163	5th match:R	CSK won by	CSK	179	RCB	87	20/04/09	Port Elizabet	5th match	RCB	CSK	CSK
392186	374163	6th match:K	KKR won by	KXIP	158	KKR	79	21/04/09	Durban	6th match	KXIP	KKR	KKR
392187	374163	7th match:M	Match abandoned without a ball bowled					21/04/09	Durban	7th match	MI	RR	Match
392188	374163	8th match:R	DC won by 2	DC	184	RCB	160	22/04/09	Cape Town	8th match	RCB	DC	DC
392189	374163	9th match:C	DD won by 9	DD	189	CSK	180	23/04/09	Durban	9th match	CSK	DD	DD
392190	374163	10th match:I	Match tied (F	RR	150	KKR	150	23/04/09	Cape Town	10th match	KKR	RR	RR
392191	374163	11th match:I	KXIP won by	RCB	168	KXIP	173	24/04/09	Durban	11th match	RCB	KXIP	KXIP
392192	374163	12th match:I	DC won by 1	DC	168	MI	156	25/04/09	Durban	12th match	DC	MI	DC
392193	374163	13th match:(	Match abandoned without a ball bowled					25/04/09	Cape Town	13th match	CSK	KKR	Match
392194	374163	14th match:I	DD won by 6	RCB	149	DD	150	26/04/09	Port Elizabet	14th match	RCB	DD	DD
392195	374163	15th match:I	KXIP won by	KXIP	139	RR	112	26/04/09	Cape Town	15th match	KXIP	RR	KXIP
392196	374163	16th match:(	DC won by 6	CSK	165	DC	169	27/04/09	Durban	16th match	CSK	DC	DC
392197	374163	17th match:I	MI won by 9	MI	187	KKR	95	27/04/09	Port Elizabet	17th match	KKR	MI	MI

MI	165	7	20	20	CSK	146	7	20	20		
RCB	133	8	20	20	RR	58	10	15.1	20		
KXIP	104	7	12	12	DD	58	0	4.5	6	1	54
KKR	101	10	19.4	20	DC	104	2	13.1	20		
CSK	179	5	20	20	RCB	87	10	15.2	20		
KXIP	158	6	20	20	KKR	79	1	9.2	9.2	1	69
DC	184	6	20	20	RCB	160	8	20	20		
DD	189	5	20	20	CSK	180	9	20	20		
RR	150	6	20	20	KKR	150	8	20	20		
RCB	168	9	20	20	KXIP	173	3	19	20		
DC	168	9	20	20	MI	156	7	20	20		
RCB	149	7	20	20	DD	150	4	19.2	20		
KXIP	139	6	20	20	RR	112	7	20	20		
CSK	165	6	20	20	DC	169	4	19.3	20		
MI	187	6	20	20	KKR	95	10	15.2	20		

win_by_runs	win_by_wick	balls_remain	innings1	innings1_rur	innings1_wic	innings1_ove	innings1_ove	innings2	innings2_rur	innings2_wic	innings2_ove	in
140			KKR	222	3	20	20	RCB	82	10	15.1	
33			CSK	240	5	20	20	KXIP	207	4	20	
	9	29	RR	129	8	20	20	DD	132	1	15.1	
	5	6	DC	110	10	18.4	20	KKR	112	5	19	
	5	2	MI	165	6	20	20	RCB	166	5	19.4	
	6	11	KXIP	166	8	20	20	RR	168	4	18.1	
	9	42	DC	142	8	20	20	DD	143	1	13	
6			CSK	208	5	20	20	MI	202	7	20	
	3	1	DC	214	5	20	20	RR	217	7	19.5	
66			KXIP	182	10	20	20	MI	116	9	20	
	9	18	KKR	147	9	20	20	CSK	152	1	17	
	7	17	RCB	135	8	20	20	RR	138	3	17.1	
	4	3	DD	158	8	20	20	KXIP	162	6	19.3	
	10	48	MI	154	7	20	20	DC	155	0	12	
13			CSK	178	5	20	20	RCB	165	10	19.4	
	7	8	KKR	137	8	20	20	MI	138	3	18.4	
10			DD	191	5	20	20	RCB	181	5	20	
45			RR	196	7	20	20	KKR	151	10	19.1	
	7	7	DC	164	8	20	20	KXIP	167	3	18.5	
	8	6	CSK	169	6	20	20	DD	172	2	19	
3			RCB	156	8	20	20	DC	153	6	20	
9			KXIP	178	6	20	20	KKR	169	6	20	

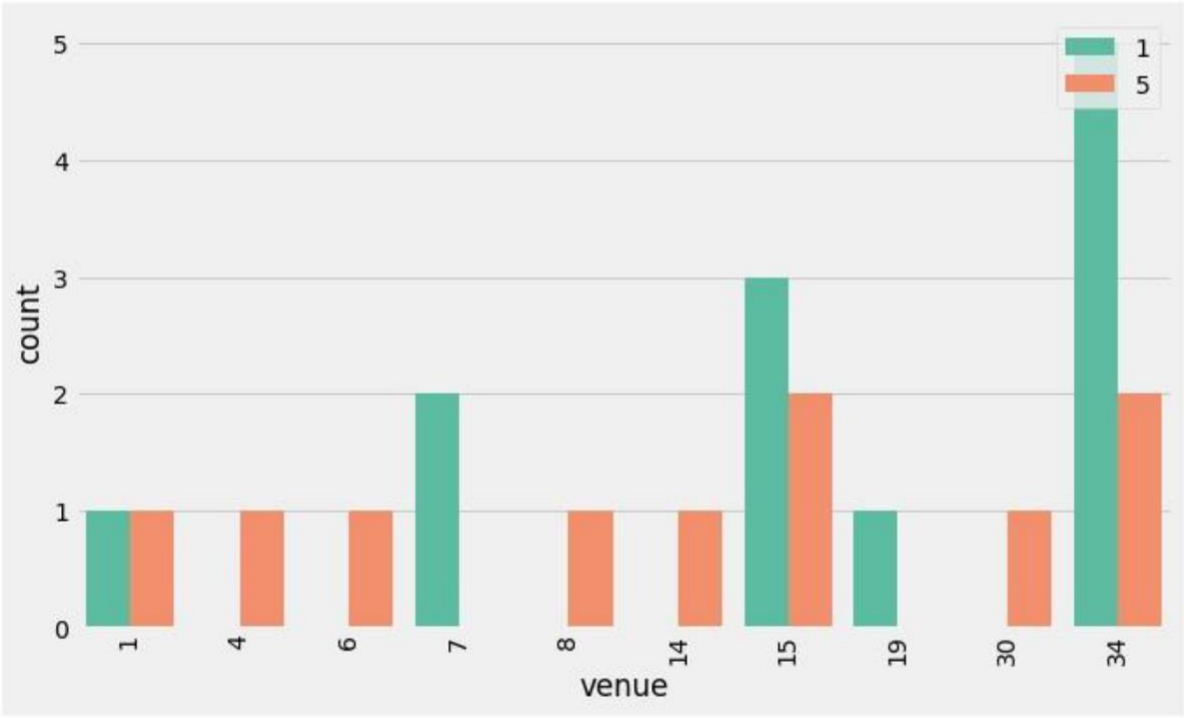
Chapter-6

Code and Explanation

Importing the packages and the datasets

```
import numpy as np
import pandas as pd
matches=pd.read_csv('matches.csv')
matches.info()
matches[pd.isnull(matches['winner'])]
matches['winner'].fillna('Draw', inplace=True)
matches.replace(['Mumbai Indians','Kolkata Knight Riders','Royal Challengers Bangalore','Deccan Ch
argers','Chennai Super Kings',
                 'Rajasthan Royals','Delhi Daredevils','Gujarat Lions','Kings XI Punjab',
                 'Sunrisers Hyderabad','Rising Pune Supergiants','Rising Pune Supergiant','Kochi T
uskers Kerala','Pune Warriors']
                ,['MI','KKR','RCB','DC','CSK','RR','DD','GL','KXIP','SRH','RPS','RPS','KTK','PW'],
inplace=True)

encode = {'team1': {'MI':1,'KKR':2,'RCB':3,'DC':4,'CSK':5,'RR':6,'DD':7,'GL':8,'KXIP':9,'SRH':10,
'RPS':11,'KTK':12,'PW':13},
          'team2': {'MI':1,'KKR':2,'RCB':3,'DC':4,'CSK':5,'RR':6,'DD':7,'GL':8,'KXIP':9,'SRH':10,
'RPS':11,'KTK':12,'PW':13},
          'toss_winner': {'MI':1,'KKR':2,'RCB':3,'DC':4,'CSK':5,'RR':6,'DD':7,'GL':8,'KXIP':9,'SR
H':10,'RPS':11,'KTK':12,'PW':13},
          'winner': {'MI':1,'KKR':2,'RCB':3,'DC':4,'CSK':5,'RR':6,'DD':7,'GL':8,'KXIP':9,'SRH':10,
'RPS':11,'KTK':12,'PW':13,'Draw':14}}
matches.replace(encode, inplace=True)
matches.head(2)
matches[pd.isnull(matches['city'])]
matches['city'].fillna('Dubai',inplace=True)
matches.describe()
```

	team1	team2	city	toss_decision	toss_winner	venue	winner
0	10	3	14	1	3	23	10
1	1	11	25	1	11	16	11
2	8	2	27	1	2	25	2
3	11	9	15	1	9	11	9
4	3	7	2	0	3	14	3
5	8	10	14	1	10	23	10
6	2	1	22	1	1	34	1




```
team1='RCB'
team2='KKR'
toss_winner='RCB'
input=[dicVal[team1],dicVal[team2],'14',dicVal[toss_winner],'2','1']
input = np.array(input).reshape((1, -1))
output=model.predict(input)
print("Possible winner : "+list(dicVal.keys())[list(dicVal.values()).index(output)])
```

```
Possible winner : KKR
```

```
In [2]: import numpy as np
        import pandas as pd
```

```
In [5]: df = pd.read_csv("ipldataset2.csv")
        df.head()
```

Out[5]:

	match_id	series_id	match_details	result	Team1	Score1	Team2	Score2	date	venue	...	innings1_wickets
0	335982.0	313494.0	1st match:RCB v KKR at Bangalore- Apr 18, 2008	KKR won by 140 runs	KKR	222.0	RCB	82.0	18/04/08	Bangalore	...	3.0
1	335983.0	313494.0	2nd match:KXIP v CSK at Mohali- Apr 19, 2008	CSK won by 33 runs	CSK	240.0	KXIP	207.0	19/04/08	Mohali	...	5.0
2	335984.0	313494.0	3rd match:DD v RR at Delhi- Apr 19, 2008	DD won by 9 wickets (with 29 balls remaining)	RR	129.0	DD	132.0	19/04/08	Delhi	...	8.0
3	335986.0	313494.0	4th match:KKR v DC at Kolkata- Apr 20, 2008	KKR won by 5 wickets (with 6 balls remaining)	DC	110.0	KKR	112.0	20/04/08	Kolkata	...	10.0
4	335985.0	313494.0	5th match:MI v RCB at Mumbai- Apr	RCB won by 5 wickets (with 2	MI	165.0	RCB	166.0	20/04/08	Mumbai	...	6.0

In [28]:
```
data = df[['Team1','Team2','Score1','Score2','venue']]
data.head()
```

Out[28]:

	Team1	Team2	Score1	Score2	venue
0	KKR	RCB	222.0	82.0	Bangalore
1	CSK	KXIP	240.0	207.0	Mohali
2	RR	DD	129.0	132.0	Delhi
3	DC	KKR	110.0	112.0	Kolkata
4	MI	RCB	165.0	166.0	Mumbai

In [31]:
```
a = []
b= []
c = 'KKR'
d = 'RCB'
e = 'Kolkata'
for i in range(0,len(data)):
    if(data.iloc[i][0]== c and data.iloc[i][1]==d and data.iloc[i][4] == e):
        a.append(data.iloc[i][2])
        b.append(data.iloc[i][3])
print(c,'=',round(sum(a)/len(a)))
print(d,'=',round(sum(b)/len(b)))
```

```
KKR = 174.0
RCB = 162.0
```

In [33]:
```
c='DD'
d='RCB'
e='Delhi'
for i in range(0,len(data)):
    if(data.iloc[i][0]==c and data.iloc[i][1]==d and data.iloc[i][4]==e):
         a.append(data.iloc[i][2])
         b.append(data.iloc[i][3])
print(c,'=',round(sum(a)/len(a)))
print(d,'=',round(sum(b)/len(b)))
```

```
DD = 168.0
RCB = 156.0
```

In [1]:
```
import numpy as np # linear algebra
import pandas as pd # data processing
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.simplefilter(action = "ignore", category = FutureWarning)
```

In [5]:
```
ipl_matches = pd.read_csv('matches.csv')
ipl_matches["type"] = "pre-qualifier"
for year in range(2008, 2017):
    final_match_index = ipl_matches[ipl_matches['season']==year][-1:].index.values[0]
    ipl_matches = ipl_matches.set_value(final_match_index, "type", "final")
    ipl_matches = ipl_matches.set_value(final_match_index-1, "type", "qualifier-2")
    ipl_matches = ipl_matches.set_value(final_match_index-2, "type", "eliminator")
    ipl_matches = ipl_matches.set_value(final_match_index-3, "type", "qualifier-1")

ipl_matches.groupby(["type"])["id"].count()
ipl_matches.head()
```

Out[5]:

	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	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad	35	0	Yuvraj S
1	2	2017	Pune	2017-04-06	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0	Rising Pune Supergiant	0	7	SPD S
2	3	2017	Rajkot	2017-04-07	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	0	Kolkata Knight Riders	0	10	CA
3	4	2017	Indore	2017-04-08	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	0	Kings XI Punjab	0	6	GJ Ma
4	5	2017	Bangalore	2017-04-08	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	normal	0	Royal Challengers Bangalore	15	0	KM Ja

In [6]:
```
deliveries = pd.read_csv('deliveries.csv')
deliveries.head()
```

Out[6]:

	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	0
1	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	2	DA Warner	S Dhawan	TS Mills	0	...	0	0	0	0
2	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	3	DA Warner	S Dhawan	TS Mills	0	...	0	0	0	0
3	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	4	DA Warner	S Dhawan	TS Mills	0	...	0	0	0	0
4	1	1	Sunrisers Hyderabad	Royal Challengers Bangalore	1	5	DA Warner	S Dhawan	TS Mills	0	...	0	0	0	0

5 rows × 21 columns

In [7]:
```
team_score = deliveries.groupby(['match_id', 'inning'])['total_runs'].sum().unstack().reset_index()
team_score.columns = ['match_id', 'Team1_score', 'Team2_score', 'Team1_superover_score', 'Team2_superover_score']
matches_agg = pd.merge(ipl_matches, team_score, left_on = 'id', right_on = 'match_id', how = 'outer')

team_extras = deliveries.groupby(['match_id', 'inning'])['extra_runs'].sum().unstack().reset_index()
team_extras.columns = ['match_id', 'Team1_extras', 'Team2_extras', 'Team1_superover_extras', 'Team2_superover_extras']
matches_agg = pd.merge(matches_agg, team_extras, on = 'match_id', how = 'outer')

#Reorder the columns to make the data more readable
cols = ['match_id', 'season','city','date','team1','team2', 'toss_winner', 'toss_decision', 'result', 'dl_applied', 'wi
matches_agg = matches_agg[cols]
matches_agg.head(2)
```

Out[7]:

	match_id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	...	Team1_superover_score	Team2_superover_s
0	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	...	NaN	
1	2	2017	Pune	2017-04-06	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0	...	NaN	

2 rows × 27 columns

In [9]:
```
batsman_grp = deliveries.groupby(["match_id", "inning", "batting_team", "batsman"])
batsmen = batsman_grp["batsman_runs"].sum().reset_index()

balls_faced = deliveries[deliveries["wide_runs"] == 0]
balls_faced = balls_faced.groupby(["match_id", "inning", "batsman"])["batsman_runs"].count().reset_index()
balls_faced.columns = ["match_id", "inning", "batsman", "balls_faced"]
batsmen = batsmen.merge(balls_faced, left_on=["match_id", "inning", "batsman"],
                        right_on=["match_id", "inning", "batsman"], how="left")

fours = deliveries[ deliveries["batsman_runs"] == 4]
sixes = deliveries[ deliveries["batsman_runs"] == 6]

fours_per_batsman = fours.groupby(["match_id", "inning", "batsman"])["batsman_runs"].count().reset_index()
sixes_per_batsman = sixes.groupby(["match_id", "inning", "batsman"])["batsman_runs"].count().reset_index()

fours_per_batsman.columns = ["match_id", "inning", "batsman", "4s"]
sixes_per_batsman.columns = ["match_id", "inning", "batsman", "6s"]

batsmen = batsmen.merge(fours_per_batsman, left_on=["match_id", "inning", "batsman"],
                        right_on=["match_id", "inning", "batsman"], how="left")
batsmen = batsmen.merge(sixes_per_batsman, left_on=["match_id", "inning", "batsman"],
                        right_on=["match_id", "inning", "batsman"], how="left")
batsmen['SR'] = np.round(batsmen['batsman_runs'] / batsmen['balls_faced'] * 100, 2)

for col in ["batsman_runs", "4s", "6s", "balls_faced", "SR"]:
    batsmen[col] = batsmen[col].fillna(0)

dismissals = deliveries[ pd.notnull(deliveries["player_dismissed"])]
dismissals = dismissals[["match_id", "inning", "player_dismissed", "dismissal_kind", "fielder"]]
dismissals.rename(columns={"player_dismissed": "batsman"}, inplace=True)
batsmen = batsmen.merge(dismissals, left_on=["match_id", "inning", "batsman"],
                        right_on=["match_id", "inning", "batsman"], how="left")

batsmen = ipl_matches[['id','season']].merge(batsmen, left_on = 'id', right_on = 'match_id', how = 'left').drop('id', a
batsmen.head(2)
```

Out[9]:

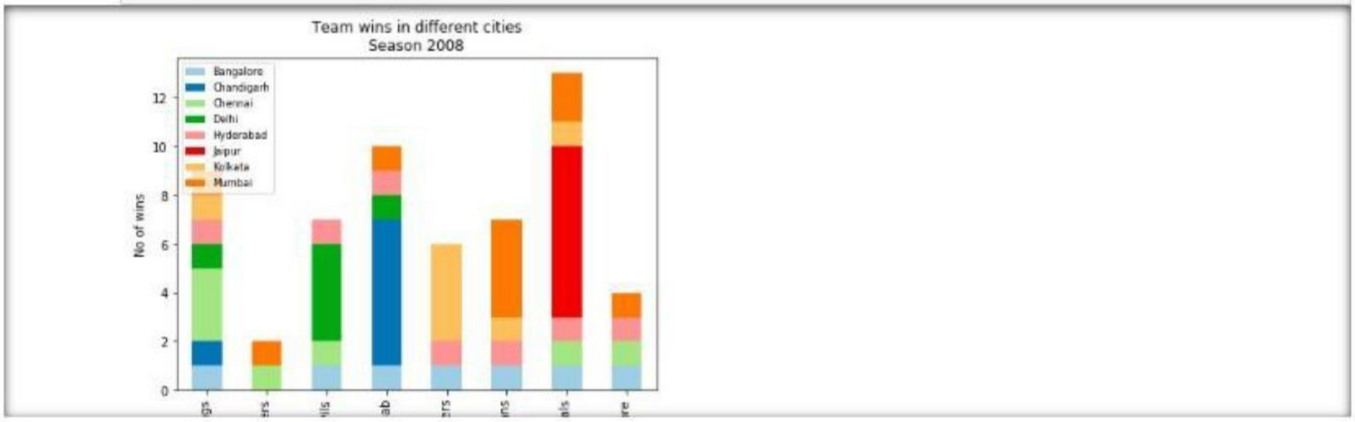
	season	match_id	inning	batting_team	batsman	batsman_runs	balls_faced	4s	6s	SR	dismissal_kind	fielder
0	2017	1	1	Sunrisers Hyderabad	BCJ Cutting	16	6.0	0.0	2.0	266.67	NaN	NaN
1	2017	1	1	Sunrisers Hyderabad	DA Warner	14	8.0	2.0	1.0	175.00	caught	Mandeep Singh

In [10]:
```
bowler_grp = deliveries.groupby(["match_id", "inning", "bowling_team", "bowler", "over"])
bowlers = bowler_grp["total_runs", "wide_runs", "bye_runs", "legbye_runs", "noball_runs"].sum().reset_index()

bowlers["runs"] = bowlers["total_runs"] - (bowlers["bye_runs"] + bowlers["legbye_runs"])
bowlers["extras"] = bowlers["wide_runs"] + bowlers["noball_runs"]

del( bowlers["bye_runs"])
del( bowlers["legbye_runs"])
del( bowlers["total_runs"])

dismissal_kinds_for_bowler = ["bowled", "caught", "lbw", "stumped", "caught and bowled", "hit wicket"]
dismissals = deliveries[deliveries["dismissal_kind"].isin(dismissal_kinds_for_bowler)]
dismissals = dismissals.groupby(["match_id", "inning", "bowling_team", "bowler", "over"])["dismissal_kind"].count().res
dismissals.rename(columns={"dismissal_kind": "wickets"}, inplace=True)

bowlers = bowlers.merge(dismissals, left_on=["match_id", "inning", "bowling_team", "bowler", "over"],
                        right_on=["match_id", "inning", "bowling_team", "bowler", "over"], how="left")
bowlers["wickets"] = bowlers["wickets"].fillna(0)

bowlers_over = bowlers.groupby(['match_id', 'inning', 'bowling_team', 'bowler'])['over'].count().reset_index()
bowlers = bowlers.groupby(['match_id', 'inning', 'bowling_team', 'bowler']).sum().reset_index().drop('over', 1)
bowlers = bowlers_over.merge(bowlers, on=["match_id", "inning", "bowling_team", "bowler"], how = 'left')
bowlers['Econ'] = np.round(bowlers['runs'] / bowlers['over'] , 2)
bowlers = ipl_matches[['id','season']].merge(bowlers, left_on = 'id', right_on = 'match_id', how = 'left').drop('id', a

bowlers.head(2)
```

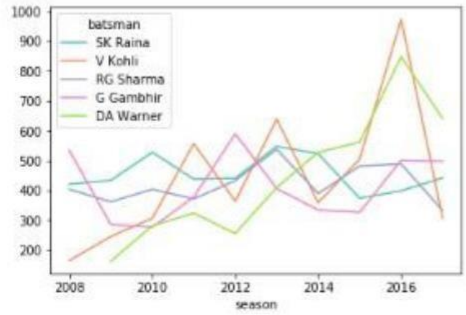
Out[10]:

	season	match_id	inning	bowling_team	bowler	over	wide_runs	noball_runs	runs	extras	wickets	Econ
0	2017	1	1	Royal Challengers Bangalore	A Choudhary	4	3	1	55	4	1.0	13.75
1	2017	1	1	Royal Challengers Bangalore	S Aravind	3	0	0	36	0	0.0	12.00

In [12]:
```
x, y = 2008, 2017
while x < y:
    wins_percity = matches_agg[matches_agg['season'] == x].groupby(['winner', 'city'])['match_id'].count().unstack()
    plot = wins_percity.plot(kind='bar', stacked=True, title="Team wins in different cities\nSeason "+str(x), figsize=(
    sns.set_palette("Set2", len(matches_agg['city'].unique()))
    plot.set_xlabel("Teams")
    plot.set_ylabel("No of wins")
    plot.legend(loc='best', prop={'size':8})
    x+=1
```

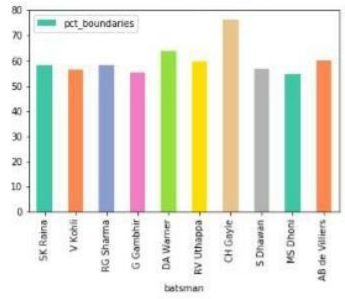


In [14]:
```
batsman_runsperseason = batsmen.groupby(['season', 'batting_team', 'batsman'])['batsman_runs'].sum().reset_index()
batsman_runsperseason = batsman_runsperseason.groupby(['season', 'batsman'])['batsman_runs'].sum().unstack().T
batsman_runsperseason['Total'] = batsman_runsperseason.sum(axis=1)
batsman_runsperseason = batsman_runsperseason.sort_values(by = 'Total', ascending = False).drop('Total', 1)
ax = batsman_runsperseason[:5].T.plot()
```



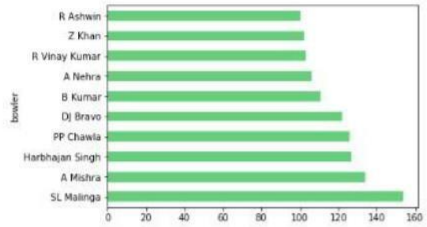
In [15]:
```
batsman_runs = batsmen.groupby(['batsman'])['batsman_runs', '4s', '6s'].sum().reset_index()
batsman_runs['4s_6s'] = batsman_runs['4s'] * 4 + batsman_runs['6s'] * 6
batsman_runs['pct_boundaries'] = np.round(batsman_runs['4s_6s'] / batsman_runs['batsman_runs'] * 100, 2)
batsman_runs = batsman_runs.sort_values(by = 'batsman_runs', ascending = False)
batsman_runs[:10].plot(x= 'batsman', y = 'pct_boundaries', kind = 'bar')
```

Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x119d3b4e0>
```



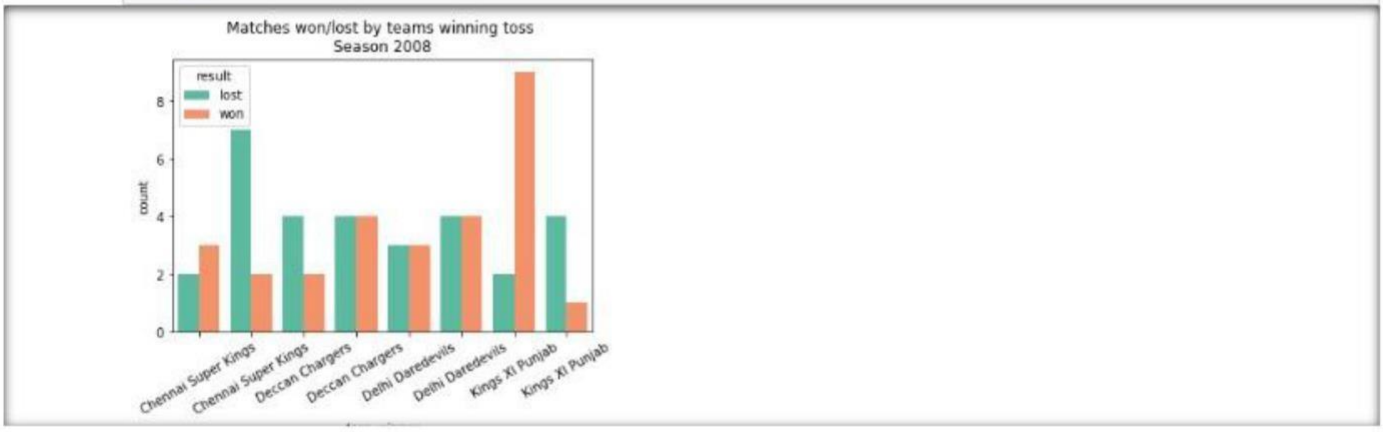
In [16]:
```
bowlers_wickets = bowlers.groupby(['bowler'])['wickets'].sum()
bowlers_wickets.sort_values(ascending = False, inplace = True)
bowlers_wickets[:10].plot(x= 'bowler', y = 'runs', kind = 'barh', colormap = 'Accent')
```

Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11b4ca2b0>
```



In [18]:
```
ipl_matches['player_of_match'].value_counts()[:10].plot(kind = 'bar')
```

In [19]:
```
toss = matches_agg.groupby(['season', 'toss_winner']).winner.value_counts().reset_index(name = 'count')
toss['result'] = np.where(toss.toss_winner == toss.winner, 'won', 'lost')
toss_result = toss.groupby(['season', 'toss_winner','result'])['count'].sum().reset_index()

for x in range(2008, 2017, 1):
    toss_result_x = toss_result[toss_result['season'] == x]
    plot = sns.barplot(x="toss_winner", y="count", hue="result", data=toss_result_x)
    plot.set_title('Matches won/lost by teams winning toss \nSeason ' +str(x))
    plot.set_xticklabels(toss_result_x['toss_winner'],rotation=30)
    plt.show()
    x+=1
```



FIGURES

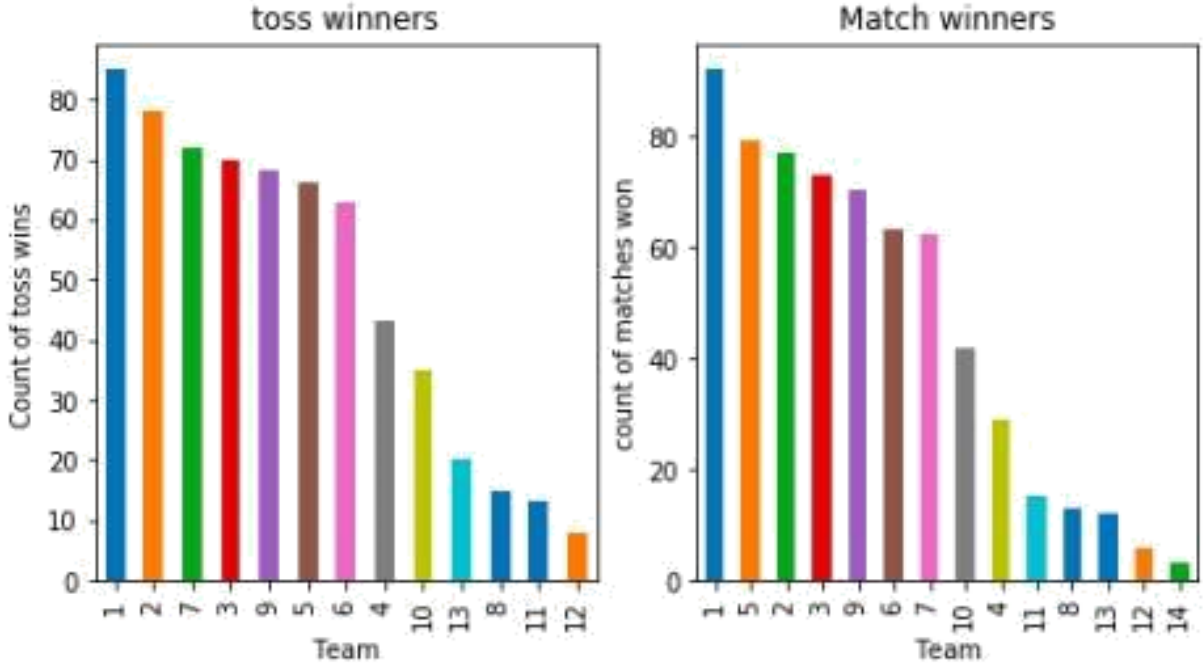


Figure 1

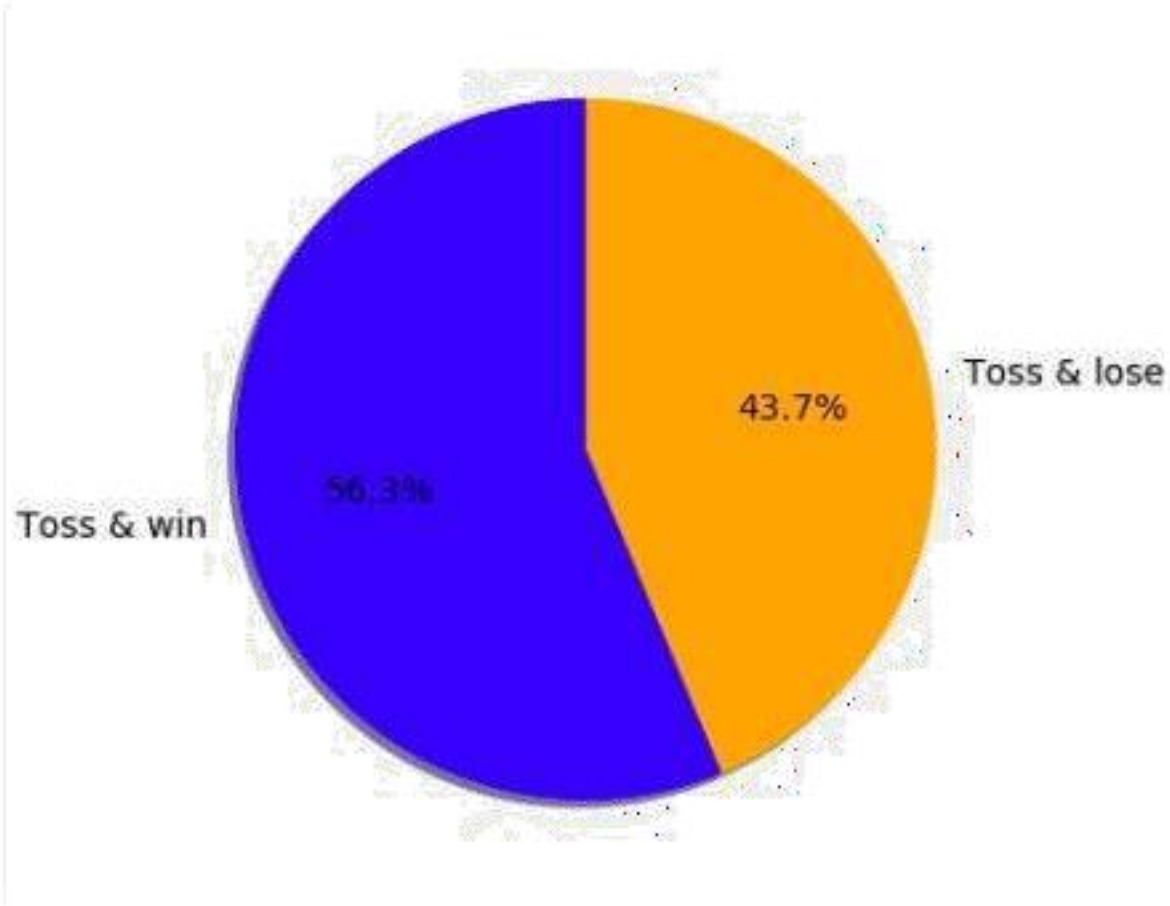


Figure 2

```
import seaborn as sns
team1=dicVal['DD']
team2=dicVal['RR']
mtemp=matches[((matches['team1']==team1)|(matches['team2']==team1))&((matches['team1']==team2)|(matches['team2']==team2
sns.countplot(x='venue', hue='winner',data=mtemp,palette='Paired')
mlt.xticks(rotation='vertical')
leg = mlt.legend( loc = 'upper right')
fig=mlt.gcf()
fig.set_size_inches(10,6)
mlt.show()
```

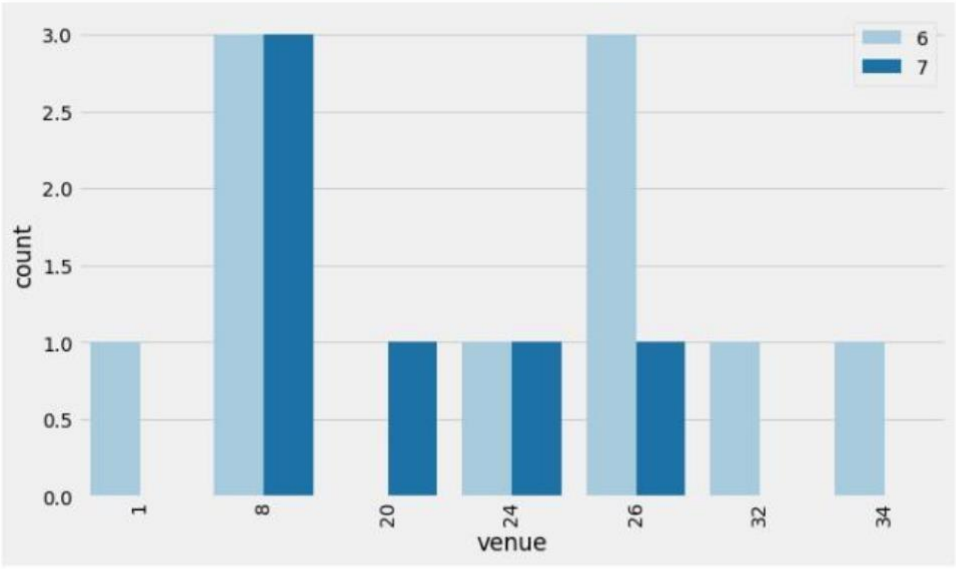


Figure 3

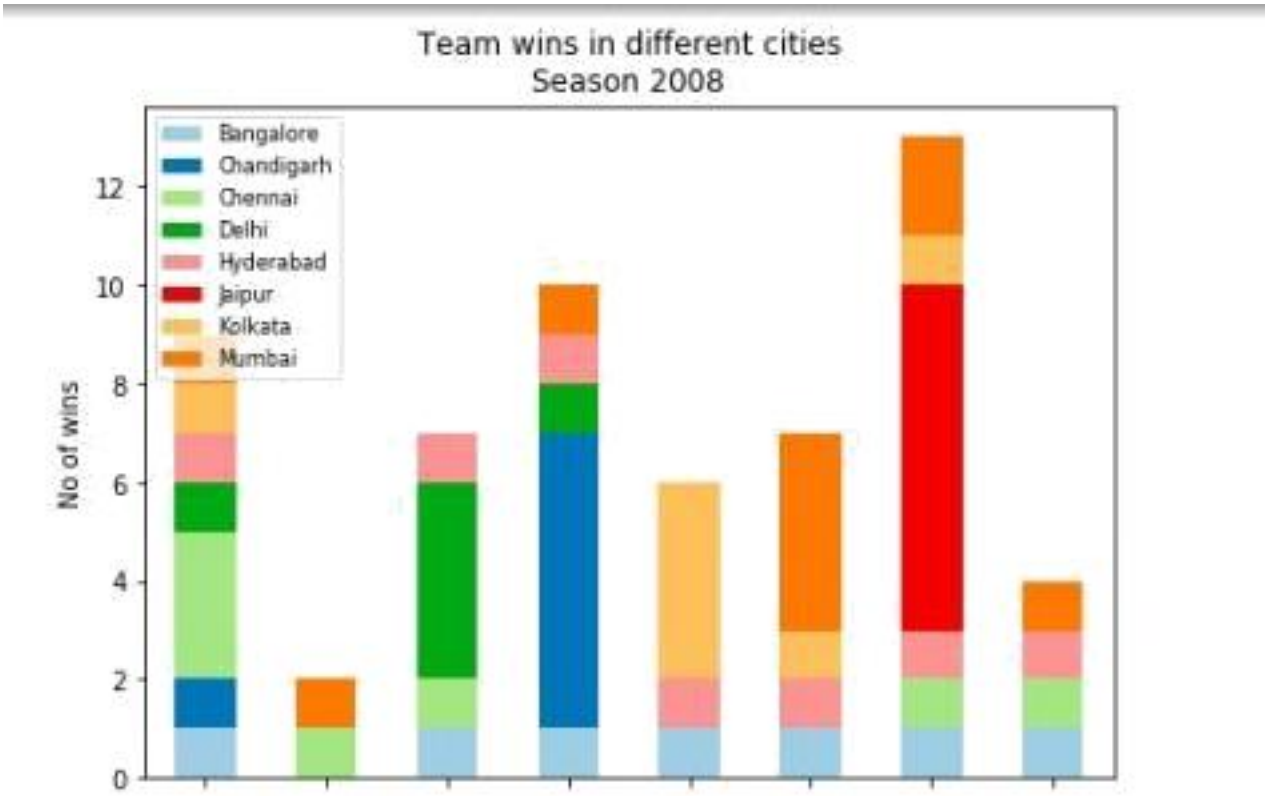


Figure 4

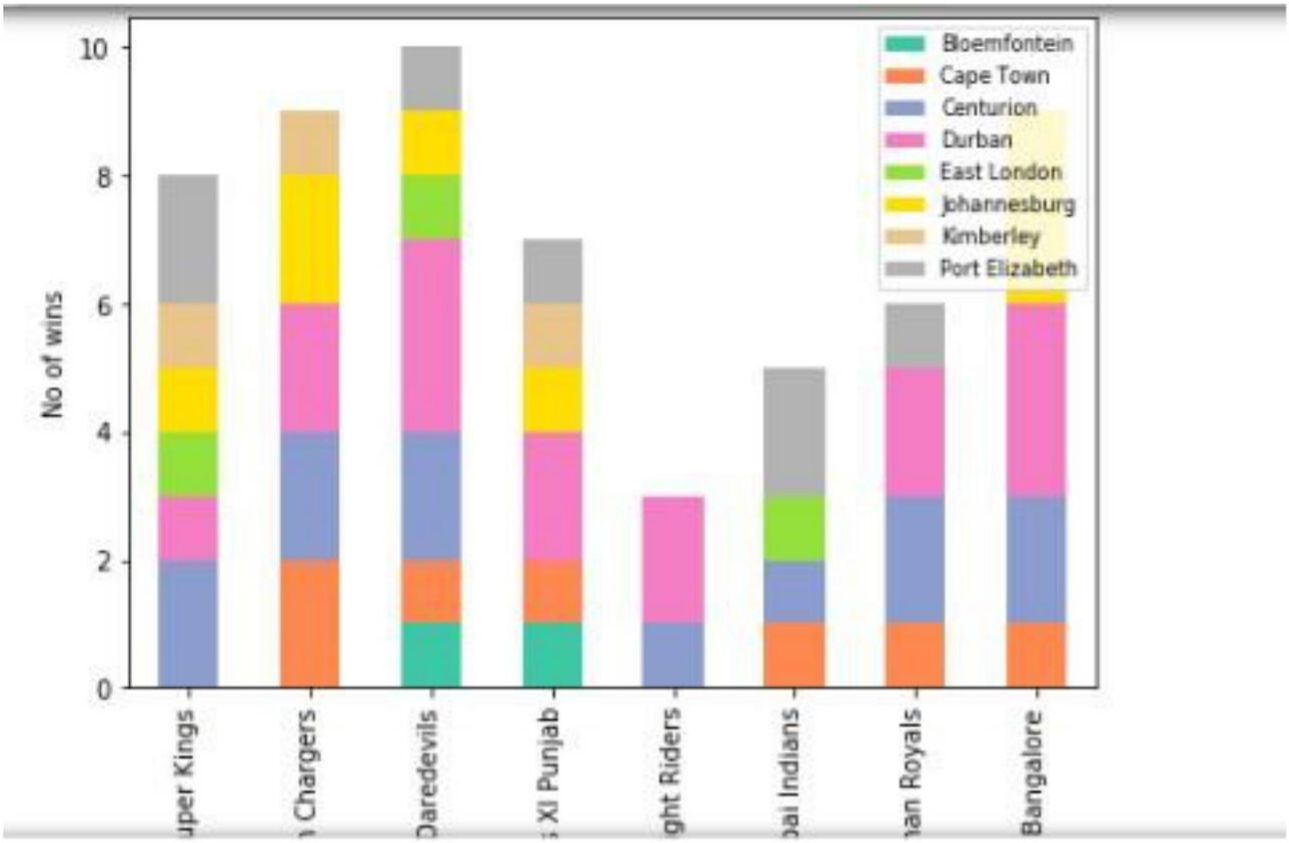


Figure 5

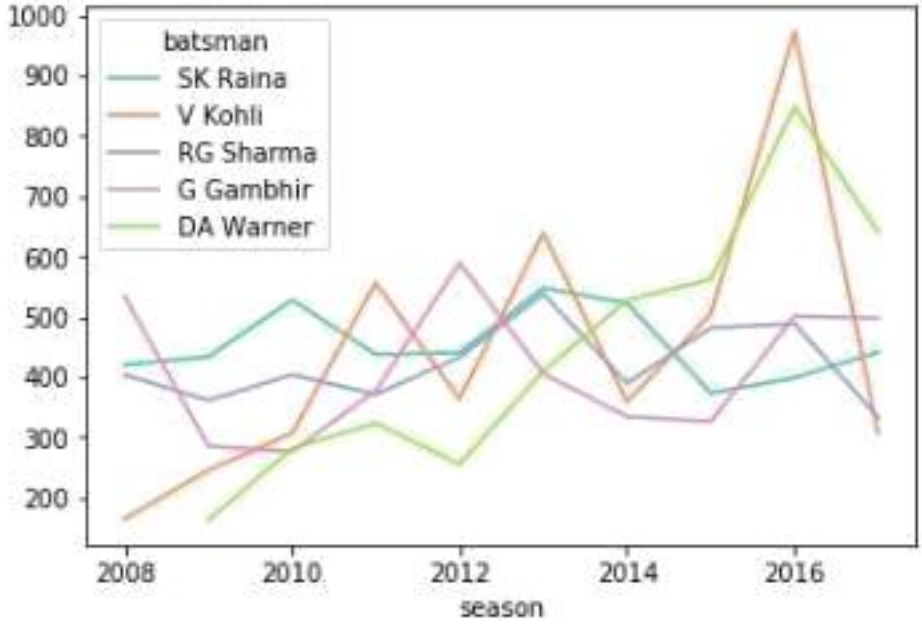


Figure 6

```
In [15]: batsman_runs = batsmen.groupby(['batsman'])['batsman_runs', '4s', '6s'].sum().reset_index()
         batsman_runs['4s_6s'] = batsman_runs['4s'] * 4 + batsman_runs['6s'] * 6
         batsman_runs['pct_boundaries'] = np.round(batsman_runs['4s_6s'] / batsman_runs['batsman_runs'] * 100, 2)
         batsman_runs = batsman_runs.sort_values(by = 'batsman_runs', ascending = False)
         batsman_runs[:10].plot(x= 'batsman', y = 'pct_boundaries', kind = 'bar')

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x119d3b4e0>
```

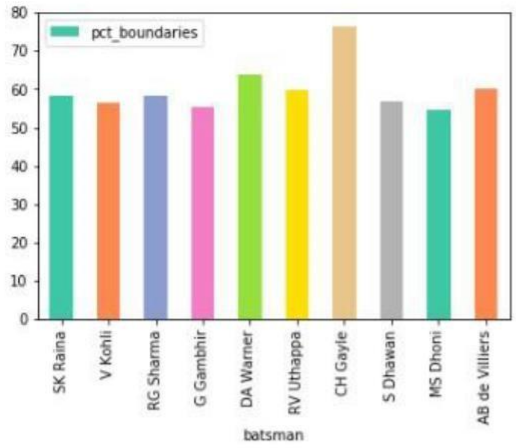
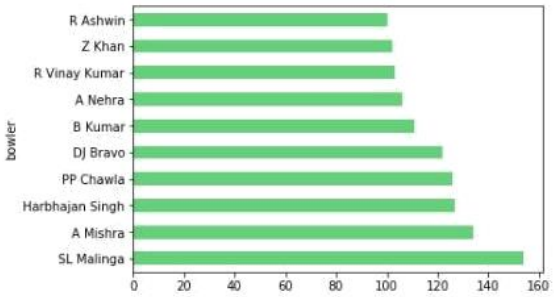


Figure 7

```
In [16]: bowlers_wickets = bowlers.groupby(['bowler'])['wickets'].sum()
         bowlers_wickets.sort_values(ascending = False, inplace = True)
         bowlers_wickets[:10].plot(x= 'bowler', y = 'runs', kind = 'barh', colormap = 'Accent')

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x11b4ca2b0>
```




```
In [18]: ipl_matches['player_of_match'].value_counts()[:10].plot(kind = 'bar')

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x11b4dbe10>
```

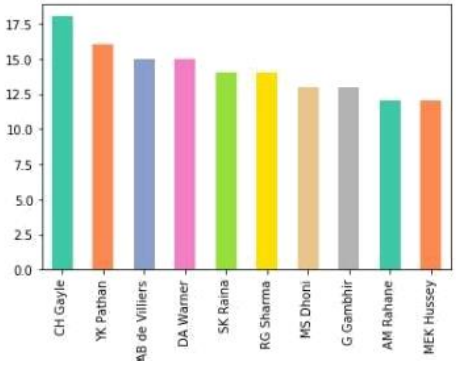


Figure 8

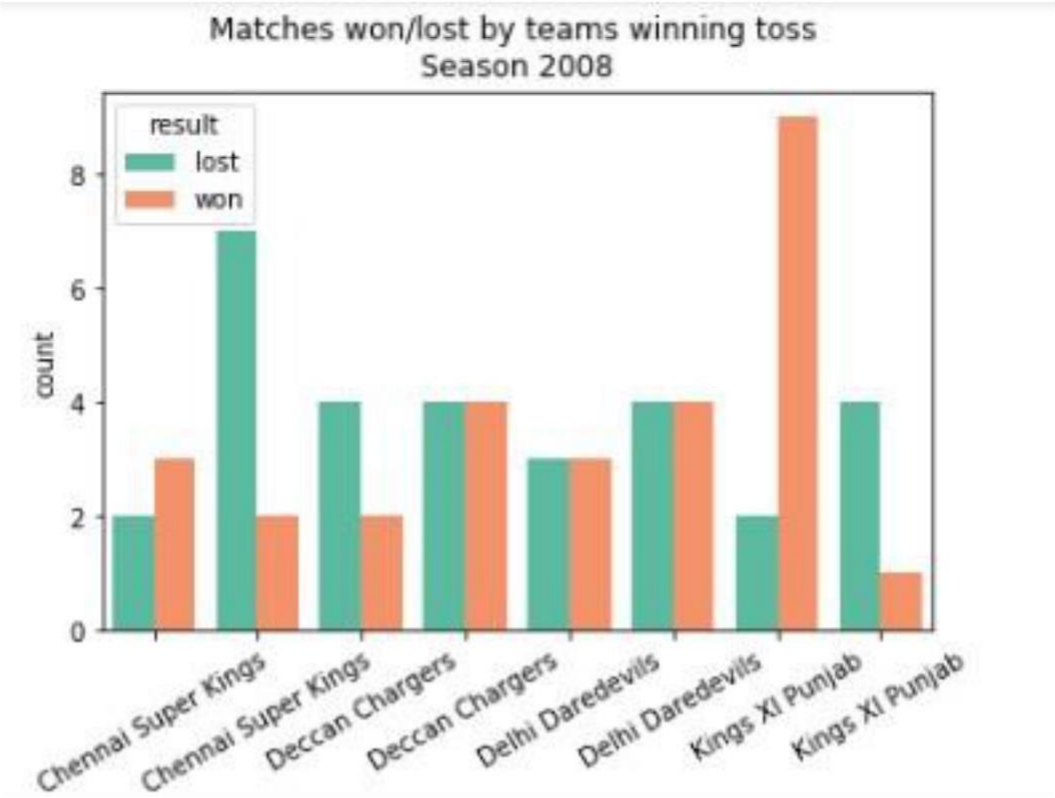


Figure 9

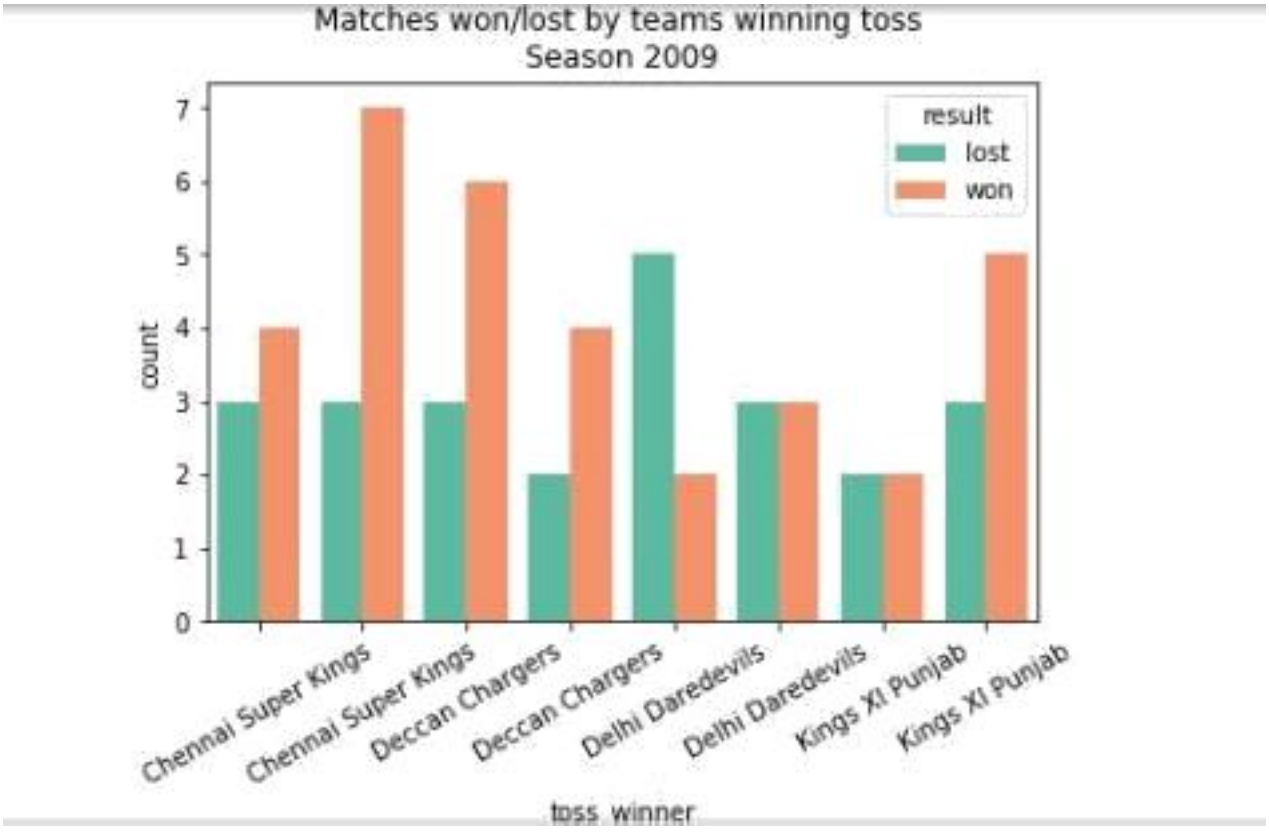


Figure 10

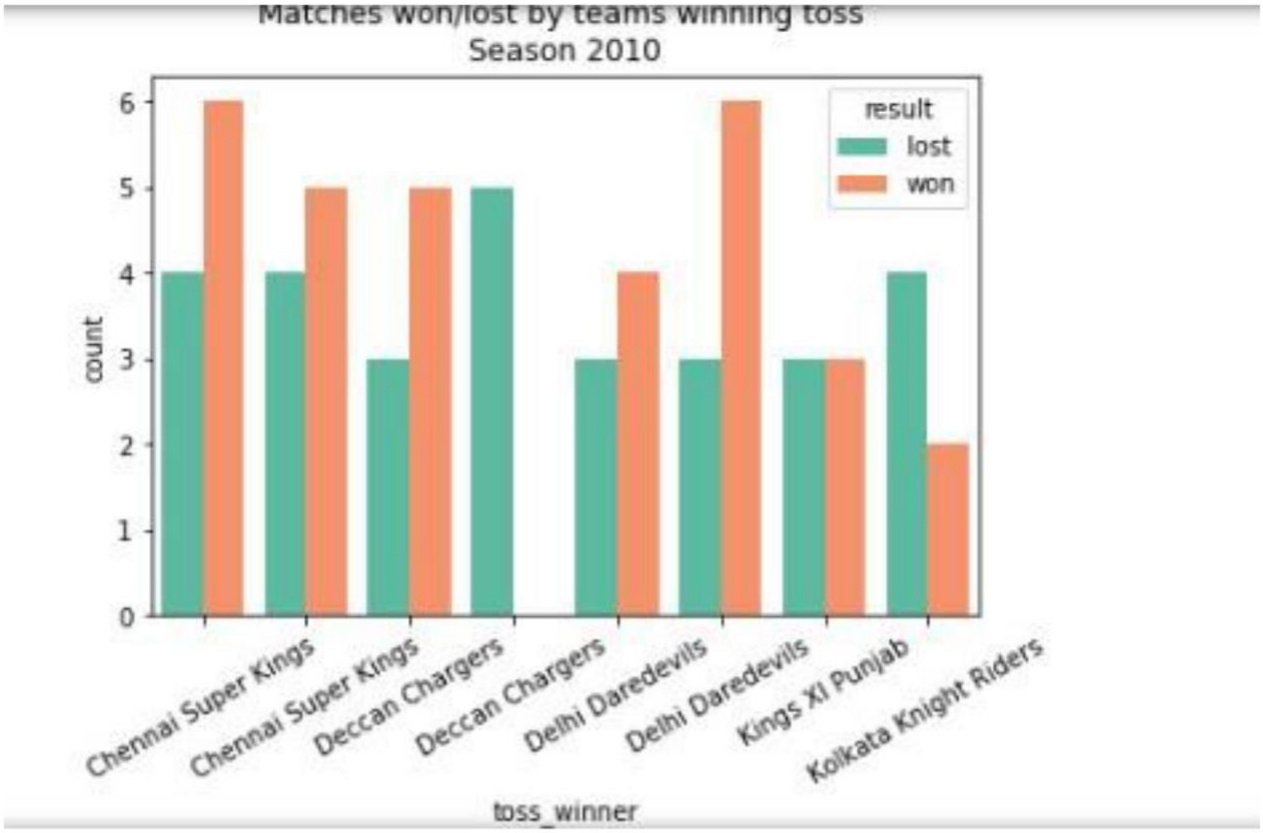


Figure 11

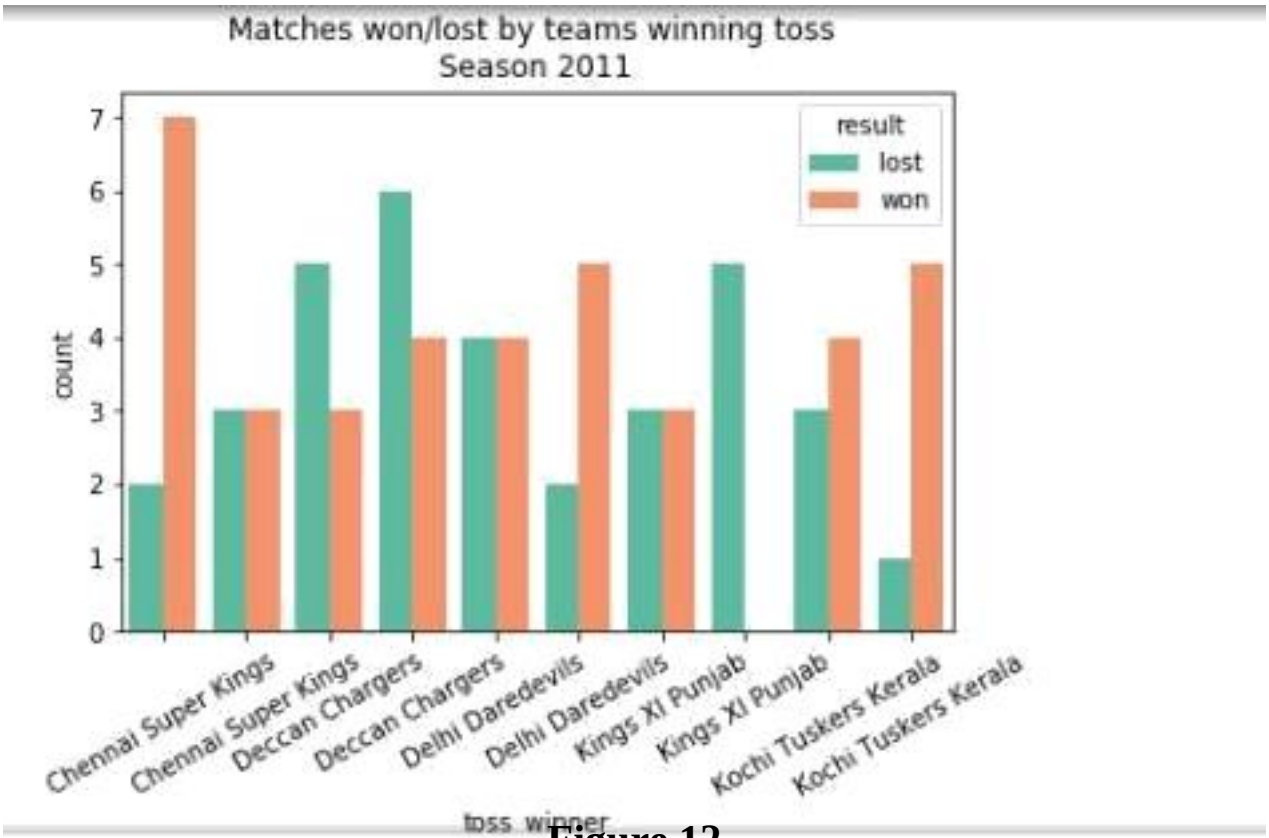


Figure 12

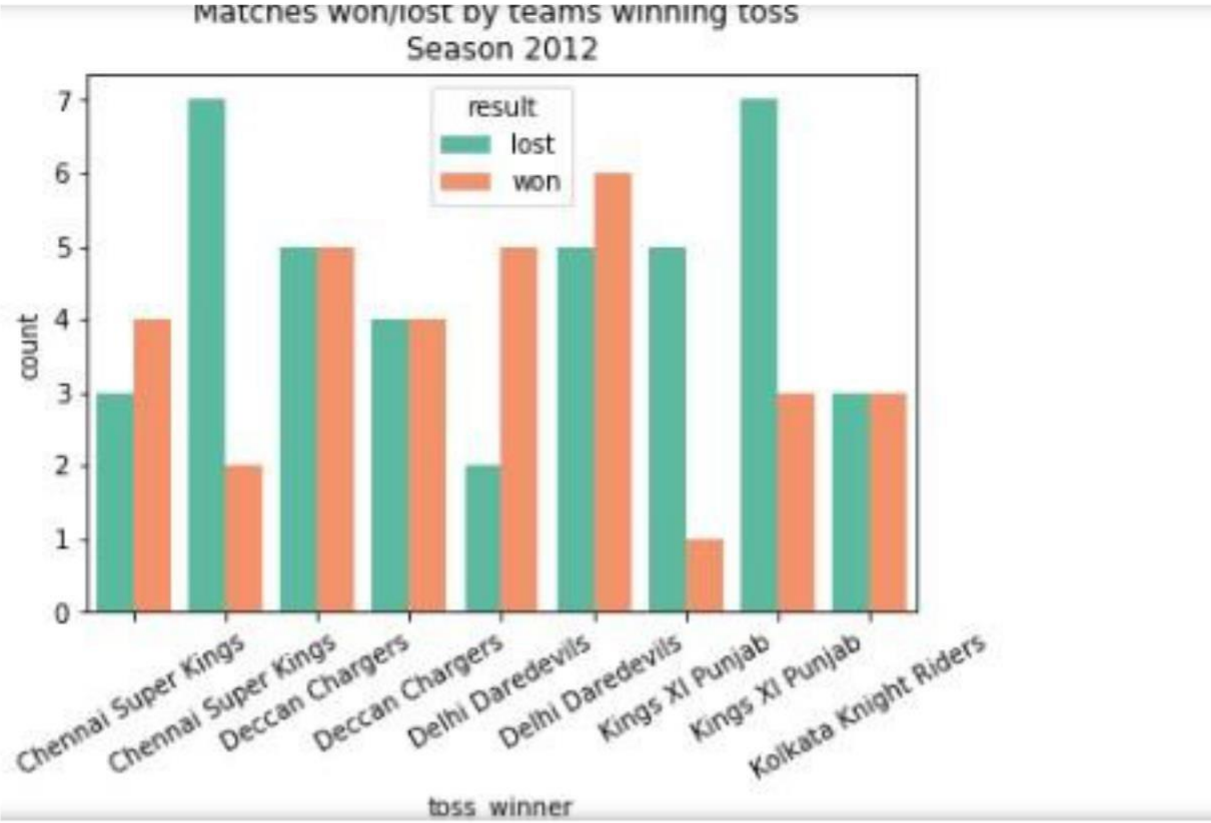


Figure 13

Result Analysis:

Conclusion

In Conclusion, the IPL wasp predicts the results fairly well and has a capability to predict score with a precision of approximately **70%**. It can be used well by Tv channels for prediction purposes. Greater accuracy could be achieved from the predictor if we take into account the players playing in the XI and pitch conditions. That shall also require good data-mining and data visualization skills and technical capabilities.

Moreover random forest classifier has been used in the work along which has helped to achieve a decent amount of accuracy.

At the end we may conclude that the project has improvement scope and even the present work is good enough to predict the score and win chances of a team correctly.

REFERENCES:

[1] WINNING AND SCORE PREDICTOR (WASP) TOOL - Anik Shah , Dhaval Jha, Jaladhi Vyas. International conferences on recent innovation in Science, Technology, Management and Environment

[2] "What's WASP all about? Corporate News. New Zealand Cricket. 23 January 2014. Retrieved, 21 May, 2014

[3] Banare, Abhijit (January 20, 2014). "WASP: Winning and Score Predictor makes for an interesting watch on television". CricketCountr Retrieved 2014-02-03.

[4] Auto-play: A Data visualization Approach to ODI Cricket Simulation and Prediction Vignesh Veppur Sankaranarayanan, Junaed Sattar and Laks V. S. Lakshmanan. Department of Computer Science University of British Columbia