



**SVKM's NMIMS**  
**Mukesh Patel School of Technology Management & Engineering**  
**A PROJECT REPORT ON**  
**Analysis of IPL dataset and prediction of outcomes**  
**based on observed trends/patterns**

**Faculty mentor:**

**MR. SURAJ PATIL**  
(Assistant Professor)  
MPSTME, NMIMS  
Shirpur Campus

**Submitted by:**

**ROHAN MATHUR (B232)**  
**SAMYAK SHAH (B245)**  
Course: B TECH CS  
Batch: 2019-2023

## **Introduction:**

The Indian Premier League (IPL) is a professional Twenty20 cricket league, contested by eight teams based out of eight different Indian cities. The IPL is the most-attended cricket league in the world and in 2014 was ranked sixth by average attendance among all sports league. The brand value of the IPL in 2019 was ₹47,500 crore (US\$6.7 billion). There have been thirteen seasons of the IPL tournament. The current IPL title holders are the Mumbai Indians, who won the 2020 season. The venue for the 2020 season was moved due to the COVID-19 pandemic and games were played in the United Arab Emirates. There is, therefore, a big demand to identify and interpret patterns and trends emerging throughout the matches by creating models and predicting outcomes of future games, by visualizing and mining the raw data of previous games.

## **Problem Statement:**

We undertake 2 tasks here:

- First, to find out if the outcome of toss has any effect on the winner of the match. We will use Naïve Bayes Classifier to create a model to predict the outcome of match winner, based on the toss winner.
- And second, to find out if the team who bats first has any advantage over the other team, by creating a decision tree classification model.

## **About the Data:**

Data contains 16 columns.

1. id – Each match has a unique id.
2. city – City where the match is played
3. date- Dates on which matches were held
4. Player\_of\_match- Player who received ‘man of the match’ award
5. venue- Name of the stadium (Generally it is a home ground of either team)
6. neutral\_venue-If the match is not played at the home ground of either of the team, then a neutral venue is chosen. ‘0’ represents no and ‘1’ represents yes.
7. team1- The host team
8. team2- Second team
9. toss\_winner- Team that wins the toss
10. toss\_decision- Decision taken after winning the toss. Either ‘bat’ or ‘field’.
11. winner- Team which wins the match.
12. result- Won by ‘runs’ or by ‘wickets’.
13. result\_margin- Margin of win by ‘runs’ or by ‘wickets’.
14. eliminator- If a match gets tied, then one-over eliminator (super over) is played.
15. method- Method of completion of the game. Contains two values- ‘NA’ if the match is completed normally with complete innings from both teams, on the other hand if somehow match could not be completed then ‘Duckworth Lewis method’(D/L) is used .
16. umpire1- Umpire behind the stumps.

## 17. umpire2- Square leg umpire.

**\*\*Note: Few columns are removed and some new columns are added during data preprocessing.**

### Implementation:


#### 1) Importing dataset

Firstly, we will import the dataset.

```
> ipl_data<-read.csv('C:\\Users\\admin\\Downloads\\IPL Matches 2008-2020 (1).csv')
> View(ipl_data)
```

	id	city	date	player_of_match	venue	neutral_venue	team1	team2
1	335982	Bangalore	2008-04-18	BB McCullum	M Chinnaswamy Stadium		0 Royal Challengers Bangalore	Kolkata Knight Riders
2	335983	Chandigarh	2008-04-19	MEK Hussey	Punjab Cricket Association Stadium, Mohali		0 Kings XI Punjab	Chennai Super Kings
3	335984	Delhi	2008-04-19	MF Maharoo	Feroz Shah Kotla		0 Delhi Capitals	Rajasthan Royals
4	335985	Mumbai	2008-04-20	MV Boucher	Wankhede Stadium		0 Mumbai Indians	Royal Challengers Bangalore
5	335986	Kolkata	2008-04-20	DJ Hussey	Eden Gardens		0 Kolkata Knight Riders	Deccan Chargers
6	335987	Jaipur	2008-04-21	SR Watson	Sawai Mansingh Stadium		0 Rajasthan Royals	Kings XI Punjab
7	335988	Hyderabad	2008-04-22	V Sehwag	Rajiv Gandhi International Stadium, Uppal		0 Deccan Chargers	Delhi Capitals
8	335989	Chennai	2008-04-23	ML Hayden	MA Chidambaram Stadium, Chepauk		0 Chennai Super Kings	Mumbai Indians
9	335990	Hyderabad	2008-04-24	YK Pathan	Rajiv Gandhi International Stadium, Uppal		0 Deccan Chargers	Rajasthan Royals
10	335991	Chandigarh	2008-04-25	KC Sangakkara	Punjab Cricket Association Stadium, Mohali		0 Kings XI Punjab	Mumbai Indians
11	335992	Bangalore	2008-04-26	SR Watson	M Chinnaswamy Stadium		0 Royal Challengers Bangalore	Rajasthan Royals
12	335993	Chennai	2008-04-26	JDP Oram	MA Chidambaram Stadium, Chepauk		0 Chennai Super Kings	Kolkata Knight Riders
13	335994	Mumbai	2008-04-27	AC Gilchrist	Dr DY Patil Sports Academy		0 Mumbai Indians	Deccan Chargers
14	335995	Chandiqrh	2008-04-27	SM Katich	Punjab Cricket Association Stadium, Mohali		0 Kinqs XI Punjab	Delhi Capitals

toss_winner	toss_decision	winner	result	result_margin	eliminator	method	umpire1	umpire2
Royal Challengers Bangalore	field	Royal Challengers Bangalore	wickets	9	N	NA	BF Bowden	VA Kulkarni
Chennai Super Kings	field	Chennai Super Kings	wickets	4	N	NA	BNJ Oxenford	C Shamshuddin
Pune Warriors	field	Royal Challengers Bangalore	runs	35	N	NA	BF Bowden	SK Tarapore
Mumbai Indians	bat	Mumbai Indians	runs	27	N	NA	S Ravi	SJA Taufel
Chennai Super Kings	field	Chennai Super Kings	wickets	9	N	NA	S Das	BR Doctrove
Rajasthan Royals	bat	Rajasthan Royals	runs	45	N	NA	BF Bowden	SK Tarapore
Deccan Chargers	bat	Kings XI Punjab	wickets	4	N	NA	HDPK Dharmasena	BNJ Oxenford
Mumbai Indians	field	Mumbai Indians	wickets	5	N	NA	S Das	BR Doctrove
Chennai Super Kings	field	Chennai Super Kings	wickets	5	N	NA	JD Cloete	SJA Taufel
Kings XI Punjab	bat	Delhi Daredevils	wickets	5	N	NA	HDPK Dharmasena	BNJ Oxenford
Mumbai Indians	field	Kolkata Knight Riders	runs	32	N	NA	S Das	BR Doctrove
Kings XI Punjab	field	Kings XI Punjab	wickets	6	N	NA	VA Kulkarni	SK Tarapore
Delhi Daredevils	field	Royal Challengers Bangalore	runs	21	N	NA	HDPK Dharmasena	C Shamshuddin
Rajasthan Royals	bat	Deccan Chargers	wickets	5	N	NA	S Ravi	SJA Taufel

 ipl\_data 812 obs. of 23 variab...

#### 2) Preprocessing the dataset:

We preprocess the data to remove any inconsistencies such as NA values, missing data. Preprocessing also includes splitting of the dataset into training and testing datasets.

Removing unnecessary columns from the dataset:

```
> ipl_data<-ipl_data[-17]
> ipl_data<-ipl_data[-16]
```

Showing 1 to 11 of 816 entries, 15 total columns

```
> DL<-ifelse(is.na(ipl_data$method), "N", "Y")
> DL
[1] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[30] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[59] "N" "N" "Y" "N" "N" "Y" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[88] "N" "N" "Y" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[117] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[146] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[175] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "Y" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[204] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "Y" "N" "N" "N" "N" "N" "N" "N" "N" "Y"
[233] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[262] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[291] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[320] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[349] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[378] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[407] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "Y" "N" "N" "N" "N" "N" "N" "N"
[436] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[465] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "Y" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[494] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "Y" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[523] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "Y" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[552] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "Y" "N" "N" "N" "Y" "Y" "N" "N" "N" "N" "N" "N" "N" "N"
[581] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[610] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "Y" "N" "N" "N" "N" "N"
[639] "N" "N" "N" "Y" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "Y" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[668] "Y" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[697] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[726] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[755] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[784] "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N" "N"
[813] "N" "N" "N" "N"
```

```
> ipl_data<-data.frame(ipl_data,DL)
```

```
> ipl_data<-ipl_data[-15]
> ipl_data$city<-ifelse(is.na(ipl_data$city),"Dubai",ipl_data$city)
ipl_data$result_margin<-ifelse(ipl_data$result=="tie",0,ipl_data$result_margin)
```

```
> unique(ip1_data$winner)
[1] "Kolkata Knight Riders"      "Chennai Super Kings"      "Delhi Daredevils"      "Royal Challengers Bangalore"
[5] "Rajasthan Royals"         "Kings XI Punjab"         "Deccan Chargers"      "Mumbai Indians"
[9] "Pune Warriors"            "Kochi Tuskers Kerala"    NA                       "Sunrisers Hyderabad"
[13] "Rising Pune Supergiants"   "Gujarat Lions"           "Rising Pune Supergiant"  "Delhi Capitals"
```

## Correcting the name “Rising Pune Supergiant” to “Rising Pune Supergiants”

```
> ip1_data$team1[ip1_data$team1=="Rising Pune Supergiant"]<-"Rising Pune Supergiants"
> ip1_data$team2[ip1_data$team2=="Rising Pune Supergiant"]<-"Rising Pune Supergiants"
> ip1_data$winner[ip1_data$winner=="Rising Pune Supergiant"]<-"Rising Pune Supergiants"
> ip1_data$toss_winner[ip1_data$toss_winner=="Rising Pune Supergiant"]<-"Rising Pune Supergiants"
```

## Correcting the name “Delhi Daredevils” to “Delhi Capitals”

```
> ip1_data$team1[ip1_data$team1=="Delhi Daredevils"]<-"Delhi Capitals"
> ip1_data$team2[ip1_data$team2=="Delhi Daredevils"]<-"Delhi Capitals"
> ip1_data$winner[ip1_data$winner=="Delhi Daredevils"]<-"Delhi Capitals"
> ip1_data$toss_winner[ip1_data$toss_winner=="Delhi Daredevils"]<-"Delhi Capitals"
```

## Finding NA values in each column

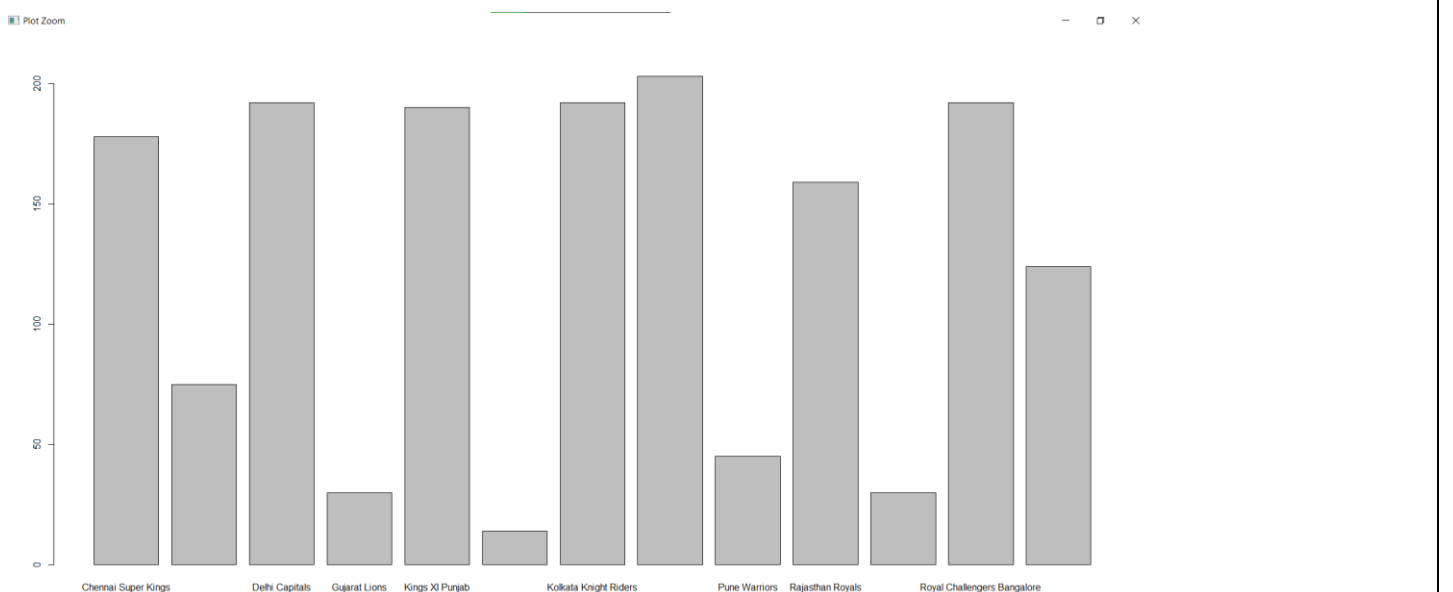
```
> colSums(is.na(ip1_data))
      id      city      date player_of_match      venue      neutral_venue      team1
      0         0         0         4         0         0         0
team2  toss_winner  toss_decision      winner      result  result_margin  eliminator
      0         0         0         4         4         4         4
      DL
      0
```

Here, it shows there are 4 rows that contain NA values..

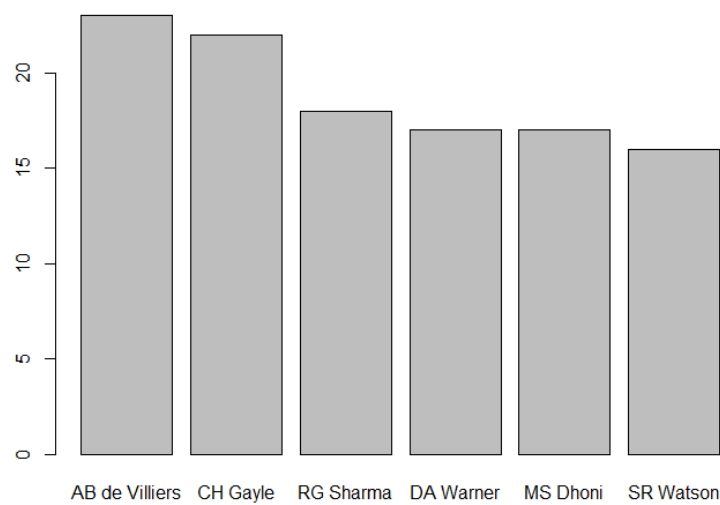
## Removing NA values

```
> ip1_data<-na.omit(ip1_data)
```

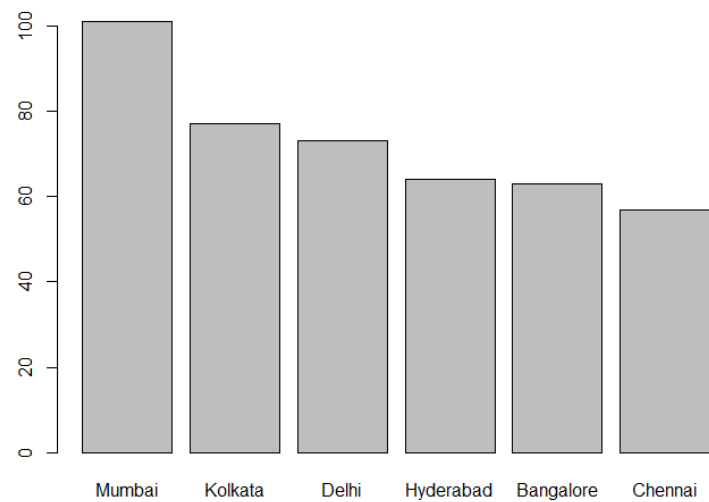
Total matches played by each team:



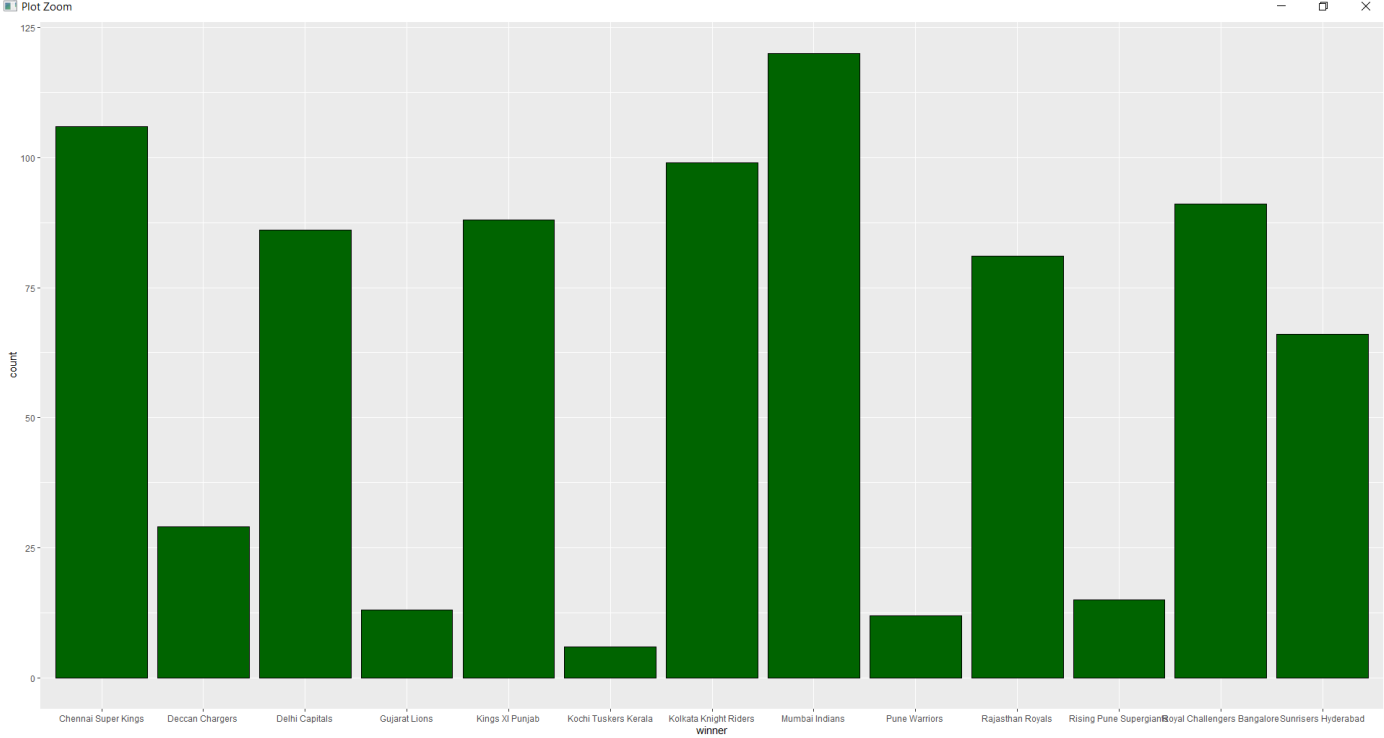
Top 5 Man of the Match Award Winners



Top 5 City Venues



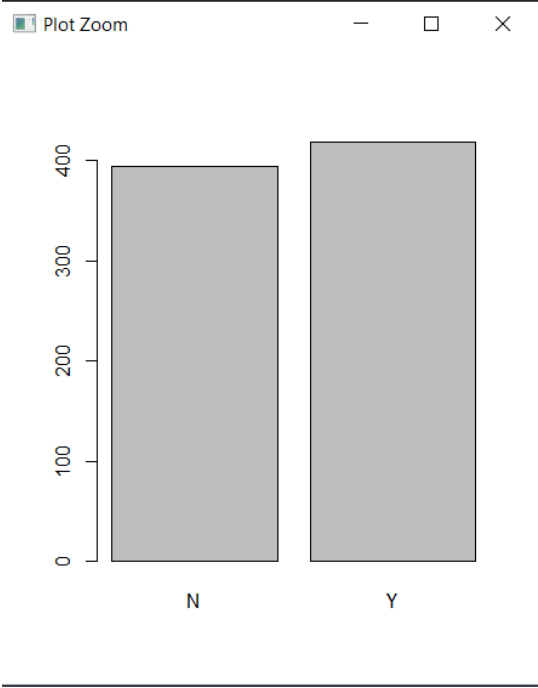
No. of wins per Team



#TASK 1

To predict chances of a team winning match if it also wins the toss using Naïve Bayes classifier

Matches won after winning toss:



## Toss winner vs Match Winner Plot

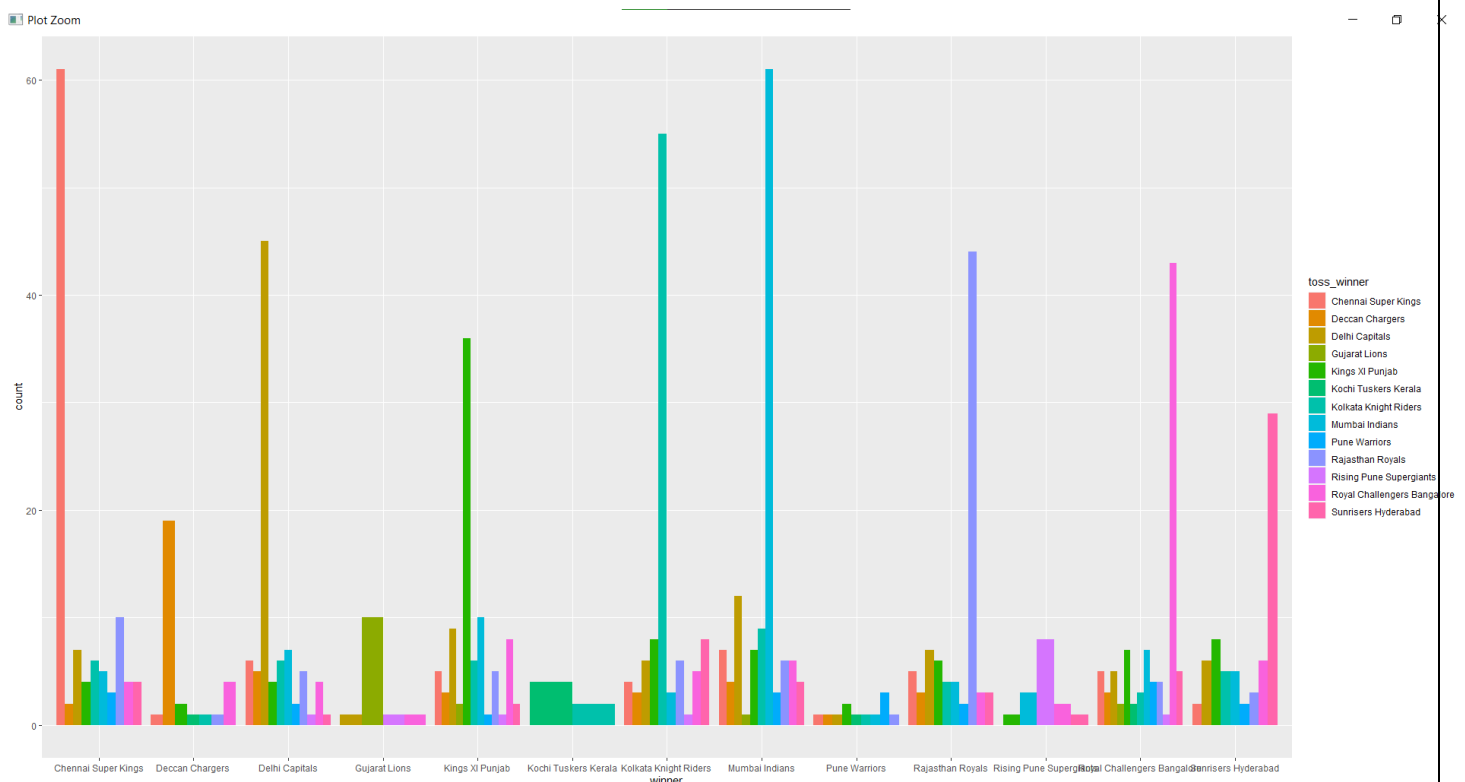
### 1) Creating a new class

Creating a class “**match\_toss\_win**”, which tells us whether a team which has won the toss has won the match as well. If true then the value is ‘Y’ otherwise ‘N’.

```
> match_toss_win<-ifelse(ipl_data$toss_winner==ipl_data$winner,"Y","N")
> unique(match_toss_win)
[1] "N" "Y"
> ipl_data<-data.frame(ipl_data,match_toss_win)
```

### 2) Splitting dataset into training and testing:

```
> ipl_data<-data.frame(ipl_data,match_toss_win)
> set.seed(2)
> id<-sample(2,nrow(ipl_data),prob=c(.7,.3),replace = TRUE)
> train_data<-ipl_data[id==1,]
> test_data<-ipl_data[id==2,]
```



▶ test_data	250 obs. of 16 variables
▶ train_data	562 obs. of 16 variables



### 3) Building the model

After the splitting of the dataset into training and testing dataset, we will build the model and train it using the training dataset

```
> library(e1071)
> library(rpart)
> library(caret)

> model<-naiveBayes(match_toss_win~.,train_data)
```

### 4) Predicting the model

After building the model, we test it using the testing dataset. We can use the model to predict the outcome of a scenario as well.

```
> p<-predict(model,test_data)
> confusionMatrix(table(p,test_data$match_toss_win))
```

Outcome:

```
p      N  Y
N  65  47
Y  58  80

      Accuracy : 0.58
      95% CI   : (0.5162, 0.6419)
No Information Rate : 0.508
P-Value [Acc > NIR] : 0.01327

      Kappa   : 0.1586

McNemar's Test P-Value : 0.32911

      Sensitivity : 0.5285
      Specificity : 0.6299
      Pos Pred value : 0.5804
      Neg Pred value : 0.5797
      Prevalence   : 0.4920
      Detection Rate : 0.2600
      Detection Prevalence : 0.4480
      Balanced Accuracy : 0.5792

      'Positive' Class : N
```

*Accuracy of the model comes out to be 58% (60% approx.)*

## #TASK 2

**To predict whether a team which “bats” first also wins the game**

### 1) Creating a new class

Creating a class “first\_bat\_win”, which tells us whether a team which batted first has won the match as well. If true then the value is ‘Y’ otherwise ‘N’.

```
> first_bat_win<-ifelse(ipl_data$toss_decision=="bat",ifelse(ipl_data$toss_winner==ipl_data$winner,"Y","N"),ifelse(ipl_data$toss_winner!=ipl_data$winner,"Y","N"))
> ipl_data<-data.frame(ipl_data,first_bat_win)
```

### 1) Splitting dataset into training and testing:

```
> train<-sample(1:nrow(ipl_data),size = ceiling(0.70*nrow(ipl_data)),replace = FALSE);
> c_train<-ipl_data[train,]
> c_test<-ipl_data[-train,]
> c1_test<-c_test[,-16]
`
```

▶ c_test	243 obs. of 23 variables
▶ c_train	569 obs. of 23 variables
▶ c1_test	243 obs. of 22 variables

### 2) Plotting the tree

```
> tree<-rpart(first_bat_win~team1+team2+toss_winner+winner+toss_decision,data=c_train)
```

```
> summary(tree)
```

Call:

```
rpart(formula = first_bat_win ~ team1 + team2 + toss_winner + winner + toss_decision, data = c_train)
n= 569
```

	CP	nsplit	rel error	xerror	xstd
1	0.07874016	0	1.0000000	1.0000000	0.04668553
2	0.05905512	2	0.8425197	1.0590551	0.04688640
3	0.04330709	3	0.7834646	0.9921260	0.04664875
4	0.03149606	6	0.6535433	0.8582677	0.04565538
5	0.02362205	7	0.6220472	0.8307087	0.04536207
6	0.01968504	10	0.5472441	0.8110236	0.04513330
7	0.01673228	11	0.5275591	0.8110236	0.04513330
8	0.01574803	15	0.4606299	0.7834646	0.04478551
9	0.01181102	16	0.4448819	0.7834646	0.04478551
10	0.01000000	17	0.4330709	0.7598425	0.04446136

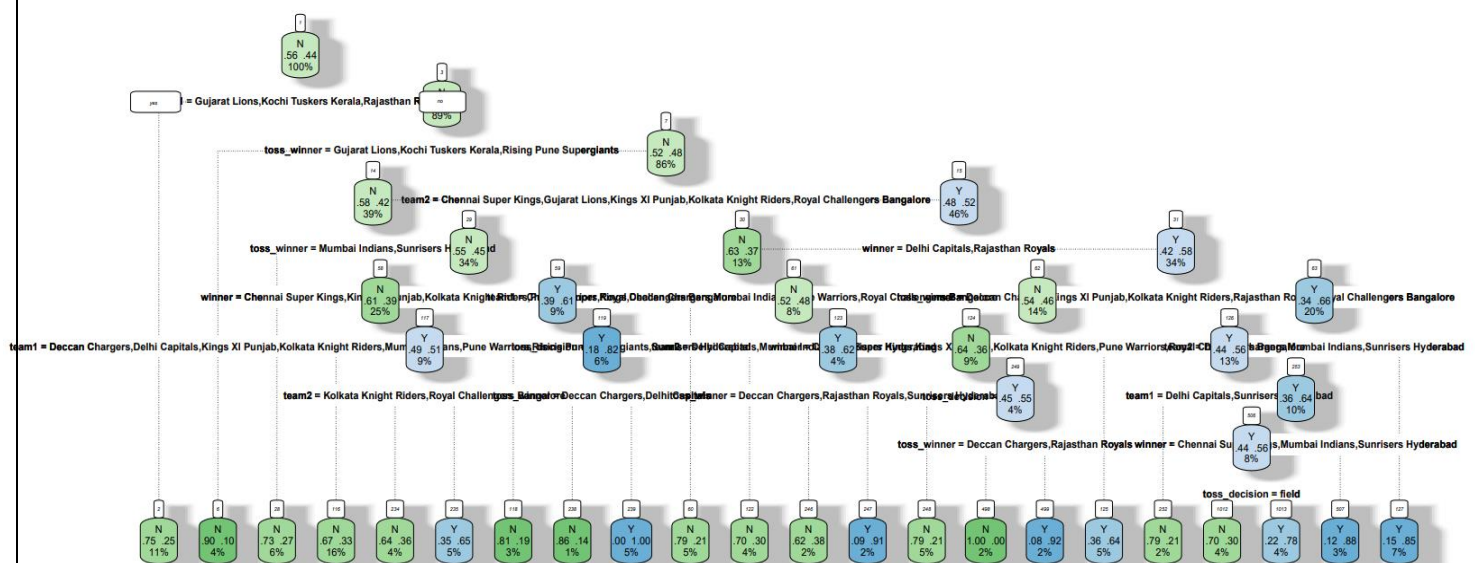
Variable importance

toss_winner	winner	toss_decision	team1	team2
27	25	17	16	14

```
Node number 1: 569 observations,      complexity param=0.07874016
predicted class=N expected loss=0.4463972 P(node) =1
  class counts:   315   254
  probabilities: 0.554 0.446
left son=2 (283 obs) right son=3 (286 obs)
Primary splits:
  winner      splits as  RRLRLRLRLLLR, improve=7.653046e+00, (0 missing)
  toss_winner  splits as  RRLRLRLRRLRR, improve=5.426690e+00, (0 missing)
  team1        splits as  RRLRLRLRLLLR, improve=5.312870e+00, (0 missing)
  team2        splits as  LRLRLRLRRLRR, improve=3.066605e+00, (0 missing)
  toss_decision splits as  RL, improve=7.277305e-07, (0 missing)
Surrogate splits:
  team1        splits as  RRLRLRLRLLLR, agree=0.749, adj=0.495, (0 split)
  toss_winner  splits as  RRLRLRLRLLLR, agree=0.701, adj=0.399, (0 split)
  team2        splits as  RRLRLRLRLLLR, agree=0.687, adj=0.371, (0 split)
  toss_decision splits as  RL, agree=0.518, adj=0.032, (0 split)
```

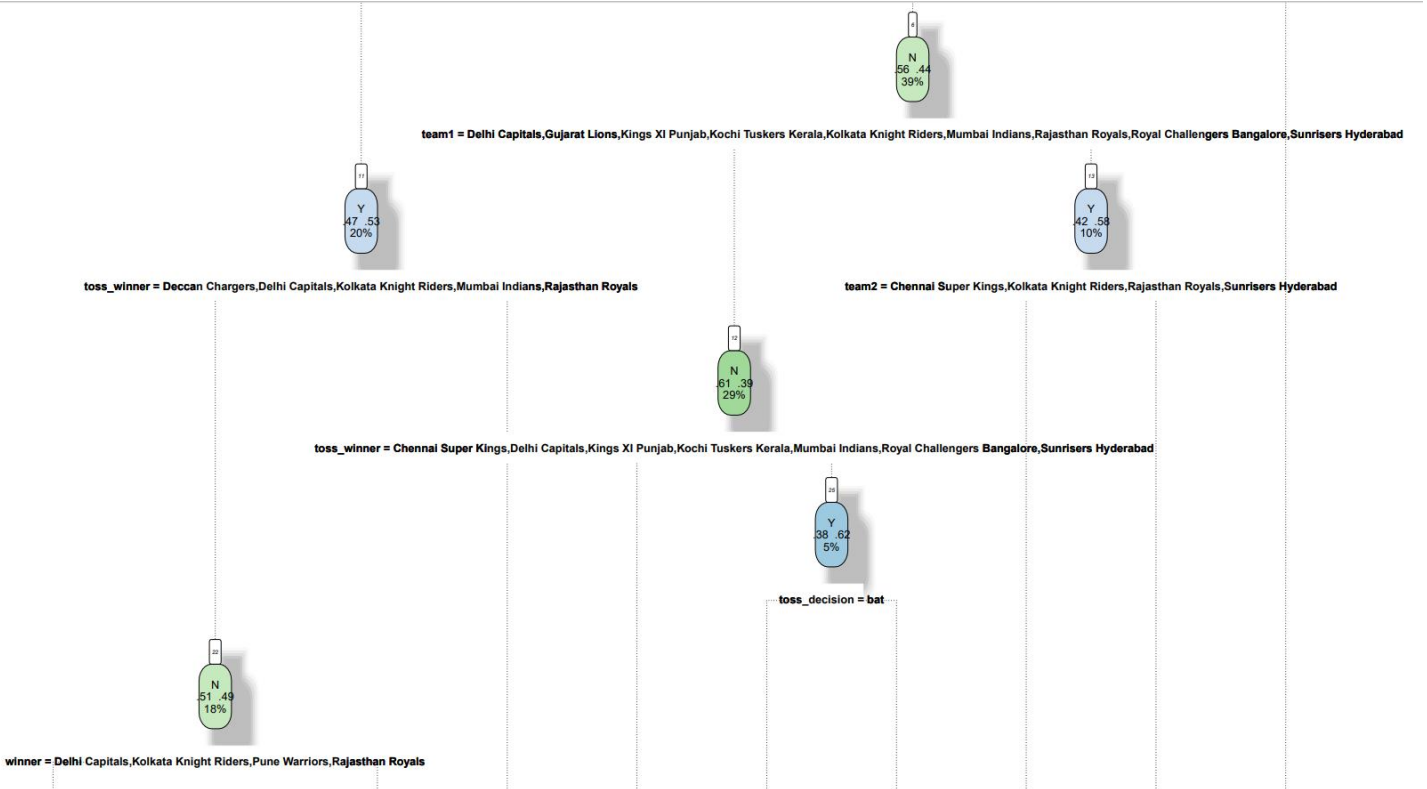
### Tree plotting function:

```
> fancyRpartPlot(tree, caption=NULL)
```



## Decision tree:

## Zoomed version:



## Predicting the Model:

```
> p<-predict(object = tree,c1_test,type="class")
> print(p)
```

```
7 8 10 13 16 18 19 26 28 32 33 35 42 48 54 55 57
N N N N N N N N N N N N N N N N N
58 60 63 64 65 72 73 80 82 87 90 92 94 96 100 105 111
N N N N N Y N Y N Y N Y Y N N N
112 114 116 117 120 124 127 129 130 132 134 138 143 148 150 151 156
N N N N Y Y N N N Y N N N Y Y N Y
157 161 163 172 179 183 185 186 192 195 196 197 203 211 214 219 222
N Y N Y N N N Y Y N Y N Y N N N N
225 226 227 246 250 251 253 258 259 261 262 267 277 278 280 285 286
N N Y N N N N N N N N Y N N Y Y
291 294 295 299 302 303 304 310 311 312 314 318 320 324 325 328 333
N N N N N Y Y N N N N N N Y Y N
334 343 345 347 362 364 372 374 375 376 382 385 387 390 391 394 397
N Y N N Y Y N N Y N N Y Y Y N N N
398 399 402 403 406 407 408 412 413 416 418 420 429 430 432 435 438
Y N N N Y N N N Y N Y N N N N
443 446 450 451 458 459 462 464 468 473 474 477 479 484 485 486 490
N N Y N N N N N N N N N N N N Y
491 493 494 500 501 509 513 515 516 519 522 524 530 533 534 535 538
N Y N N N Y N N N N N N N Y N N
541 542 545 552 560 564 565 567 571 574 580 584 586 588 590 593 599
N N N N Y Y N Y N N N N N N N N
601 610 614 615 616 622 624 628 629 632 638 640 644 649 655 661 662
Y N N N N N Y N N N N N N N N N
675 680 696 697 698 700 701 702 705 710 712 714 719 720 735 740 743
Y N N N N Y N Y N N Y N N Y N Y
747 749 752 755 759 761 764 766 768 773 780 783 786 790 792 799 805
Y N N N Y N N N Y N N N N N Y N
808 810 811 812 813
Y N N N Y
```

## Finding Accuracy through Confusion matrix :

```
> confusionMatrix(table(p,c_test$first_bat_win))  
Confusion Matrix and Statistics
```

```
p      N      Y  
N 118    62  
Y  14    49
```

```
Accuracy : 0.6872  
95% CI : (0.6249, 0.745)
```

```
No Information Rate : 0.5432  
P-Value [Acc > NIR] : 3.248e-06
```

```
Kappa : 0.3473
```

```
McNemar's Test P-Value : 6.996e-08
```

```
Sensitivity : 0.8939  
Specificity : 0.4414  
Pos Pred Value : 0.6556  
Neg Pred Value : 0.7778  
Prevalence : 0.5432  
Detection Rate : 0.4856  
Detection Prevalence : 0.7407  
Balanced Accuracy : 0.6677
```

```
'Positive' Class : N
```

***Accuracy comes out to be 68.72%***

## Conclusion:

- We established the effect of toss outcomes on the match winner and found the accuracy to be more than 50%, indicating that team that wins the toss indeed has an advantage over the other team.
- We also found that the team bats first has a significant advantage over the other team as the accuracy of the decision tree model was about 68%, indicating the same.