

# IPL winning prediction

## Introduction

Machine Learning , Deep Learning and Data Science are the fastest-growing technologies now. This field results in amazing changes in the medical field, production, robotics etc. The main reason for the advancement in this field is the increase in the computational power and availability of large amounts of data.

## Dataset

The data set we are using here is the IPL dataset, which contains the details regarding the winner and match stats. It contains the details like teams played, winner, venue of the match, won by how many wickets and runs, toss decision, whether DLS applied or not, names of the umpires etc. Performing exploratory data analysis and data engineering on this data is very important.

### \*Importing Dependencies Libraries\*

```
In [1]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, accuracy_score
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
```

### \*Loading Dataset\*

```
In [2]: df = pd.read_csv('/content/drive/MyDrive/ML Project/My project/IPL/Data/matches.csv')
```

### \*EDA\*

```
In [3]: df.shape
```

```
Out[3]: (756, 18)
```

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

```
Out[4]:
```

	id	Season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	
0	1	IPL-2017	Hyderabad	05-04-2017	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	St Hyd
1	2	IPL-2017	Pune	06-04-2017	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0	Risin Sup
2	3	IPL-2017	Rajkot	07-04-2017	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	0	
3	4	IPL-2017	Indore	08-04-2017	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	0	k
				08-	Royal		Royal				

4	5	IPL-2017	Bangalore	04-2017	Challengers Bangalore	Delhi Daredevils	Challengers Bangalore	bat	normal	0	Chal Bar
---	---	----------	-----------	---------	-----------------------	------------------	-----------------------	-----	--------	---	----------



In [5]:

```
df.info()
```

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

In [6]:

```
df.describe()
```

Out[6]:

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

In [7]:

```
df.isna().sum()
```

Out[7]:

id	0
Season	0
city	7
date	0
team1	0
team2	0
toss_winner	0
toss_decision	0
result	0
dl_applied	0
winner	4
win_by_runs	0
win_by_wickets	0
player_of_match	4
venue	0
umpire1	2
umpire2	2
umpire3	627

umpire3  
dtype: int64

```
In [8]: df = df.drop(['umpire3'], axis=1)
```

```
In [9]: df.head()
```

```
Out[9]:
```

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

```
In [10]: df.dropna()
```

```
Out[10]:
```

son	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner
IPL-017	Hyderabad	05-04-2017	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad
IPL-017	Pune	06-04-2017	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0	Rising Pune Supergiant
IPL-017	Rajkot	07-04-2017	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	0	Kolkata Knight Riders
IPL-017	Indore	08-04-2017	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	0	Kings XI Punjab
IPL-017	Hyderabad	09-04-2017	Gujarat Lions	Sunrisers Hyderabad	Sunrisers Hyderabad	field	normal	0	Sunrisers Hyderabad
...	...	...	...	...	...	...	...	...	...
IPL-019	Mohali	05-05-2019	Chennai Super Kings	Kings XI Punjab	Kings XI Punjab	field	normal	0	Kings XI Punjab
IPL-019	Mumbai	05-05-2019	Kolkata Knight Riders	Mumbai Indians	Mumbai Indians	field	normal	0	Mumbai Indians
IPL-019	Chennai	07-05-2019	Chennai Super Kings	Mumbai Indians	Chennai Super Kings	bat	normal	0	Mumbai Indians

IPL-019	Visakhapatnam	10-05-2019	Delhi Capitals	Chennai Super Kings	Chennai Super Kings	field	normal	0	Chennai Super Kings
IPL-019	Hyderabad	12-05-2019	Mumbai Indians	Chennai Super Kings	Mumbai Indians	bat	normal	0	Mumbai Indians

columns



In [11]: `df.shape`

Out[11]: (756, 17)

In [12]: `df['team1'].unique()`

Out[12]: 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)

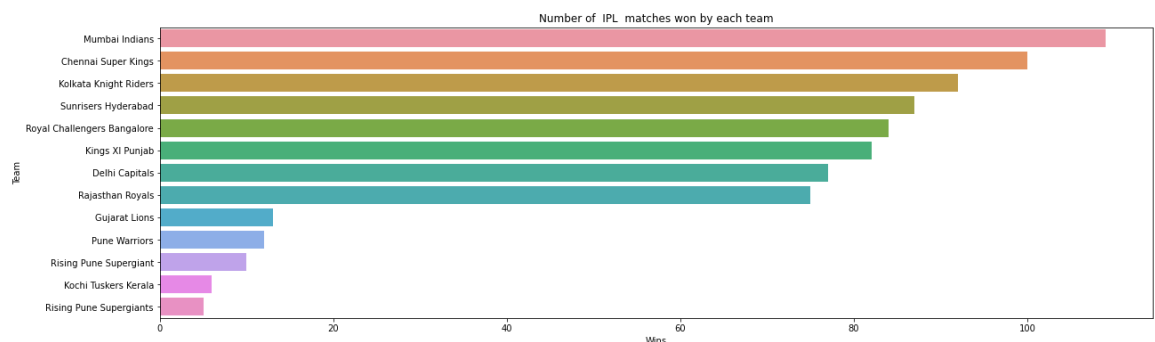
In [13]: `df['team1']=df['team1'].str.replace('Delhi Daredevils','Delhi Capitals')  
df['team2']=df['team2'].str.replace('Delhi Daredevils','Delhi Capitals')  
df['winner']=df['winner'].str.replace('Delhi Daredevils','Delhi Capitals')`

In [14]: `df['team1']=df['team1'].str.replace('Deccan Chargers','Sunrisers Hyderabad')  
df['team2']=df['team2'].str.replace('Deccan Chargers','Sunrisers Hyderabad')  
df['winner']=df['winner'].str.replace('Deccan Chargers','Sunrisers Hyderabad')`

### \*Data Visualization\*

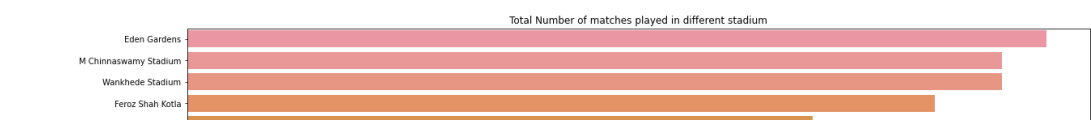
In [15]: `plt.figure(figsize = (20,6))  
sns.countplot(y='winner', data=df, order=df['winner'].value_counts().index)  
plt.xlabel('Wins')  
plt.ylabel('Team')  
plt.title('Number of IPL matches won by each team')`

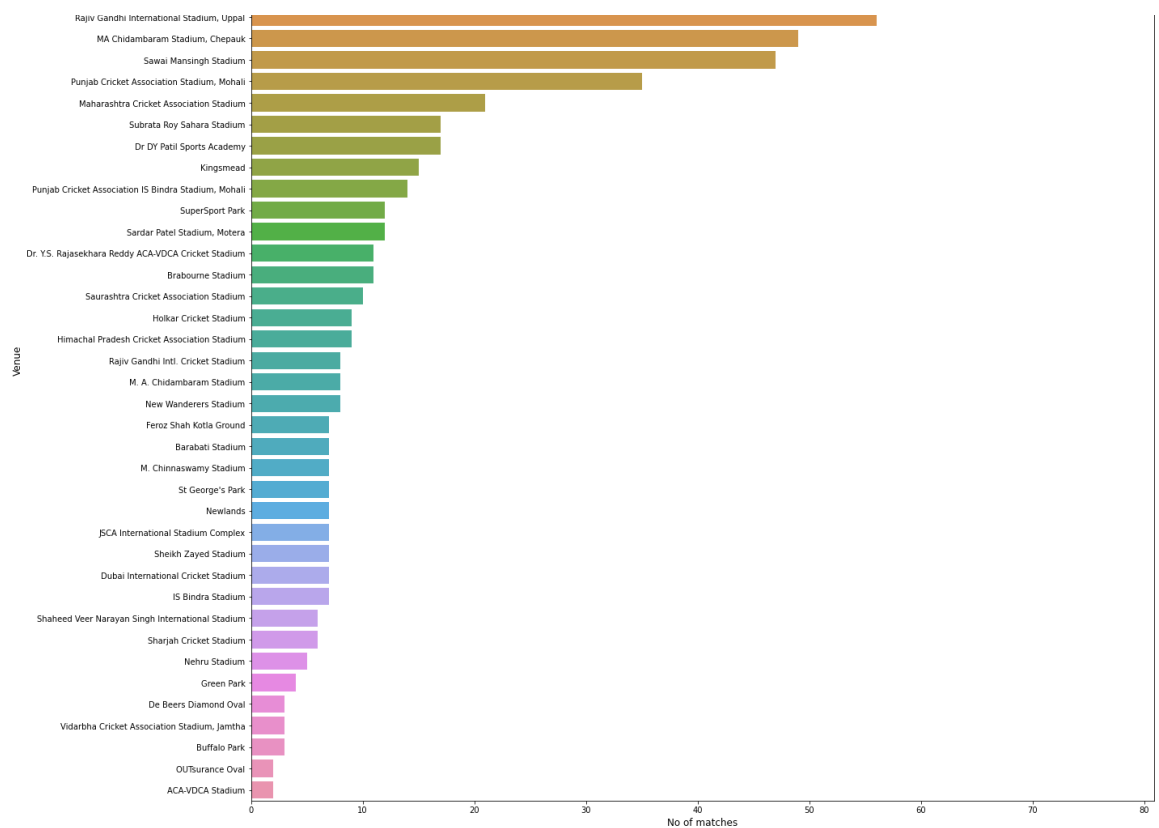
Out[15]: Text(0.5, 1.0, 'Number of IPL matches won by each team')



In [16]: `plt.figure(figsize = (20,20))  
sns.countplot(y='venue', data=df, order=df['venue'].value_counts().index)  
plt.xlabel('No of matches',fontsize=12)  
plt.ylabel('Venue',fontsize=12)  
plt.title('Total Number of matches played in different stadium')`

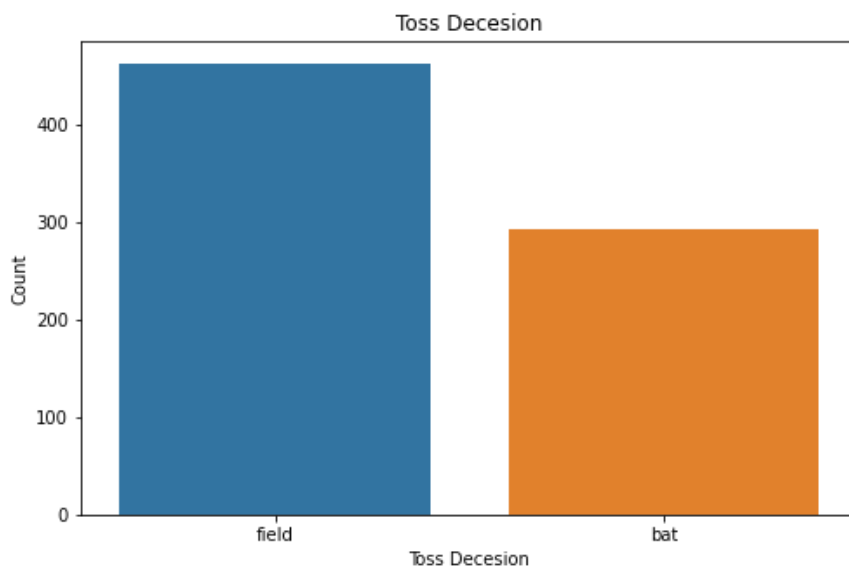
Out[16]: Text(0.5, 1.0, 'Total Number of matches played in different stadium')





```
In [17]: plt.figure(figsize=(8,5))
sns.countplot(x='toss_decision', data=df)
plt.xlabel('Toss Decesion', fontsize=10)
plt.ylabel('Count',fontsize=10)
plt.title('Toss Decesion')
```

```
Out[17]: Text(0.5, 1.0, 'Toss Decesion')
```



### \*Data Modification\*

```
In [18]: x = ["city", "toss_decision", "result", "dl_applied"]
for i in x:
    print("-----")
    print(df[i].unique())
    print(df[i].value_counts())

-----
['Hyderabad' 'Pune' 'Rajkot' 'Indore' 'Bangalore' 'Mumbai' 'Kolkata'
'Delhi' 'Chandigarh' 'Kanpur' 'Jaipur' 'Chennai' 'Cape Town'
'Port Elizabeth' 'Durban' 'Centurion' 'East London' 'Johannesburg'
'Kimberley' 'Bloemfontein' 'Ahmedabad' 'Cuttack' 'Nagpur' 'Dharamsala']
```

```

'Kochi' 'Visakhapatnam' 'Raipur' 'Ranchi' 'Abu Dhabi' 'Sharjah' nan
'Mohali' 'Bengaluru']
Mumbai          101
Kolkata          77
Delhi            74
Bangalore        66
Hyderabad        64
Chennai          57
Jaipur           47
Chandigarh       46
Pune             38
Durban           15
Bengaluru        14
Visakhapatnam    13
Centurion        12
Ahmedabad        12
Rajkot           10
Mohali           10
Indore           9
Dharamsala       9
Johannesburg     8
Cuttack          7
Ranchi           7
Port Elizabeth   7
Cape Town        7
Abu Dhabi        7
Sharjah          6
Raipur           6
Kochi            5
Kanpur           4
Nagpur           3
Kimberley        3
East London      3
Bloemfontein     2
Name: city, dtype: int64
-----
['field' 'bat']
field    463
bat      293
Name: toss_decision, dtype: int64
-----
['normal' 'tie' 'no result']
normal    743
tie        9
no result   4
Name: result, dtype: int64
-----
[0 1]
0    737
1     19
Name: dl_applied, dtype: int64

```

```
In [19]: df.drop(["id", "Season","city","date", "player_of_match", 'umpire1', "venue", "umpire2"],
```

```
In [20]: df.head()
```

```
Out[20]:
```

	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by_w
0	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad	35	
1	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0	Rising Pune Supergiant	0	
2	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	0	Kolkata Knight Riders	0	
3	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	0	Kings XI Punjab	0	
4	Royal Challengers	Delhi	Royal Challengers	bat	normal	0	Royal Challengers	15	



**\*Splitting Dataset\***

In [21]:

```
x = df.drop(['winner'], axis=1)
y = df['winner']
```

In [22]:

```
x = pd.get_dummies(x, ["team1","team2", "toss_winner", "toss_decision", "result"], drop_f:
```

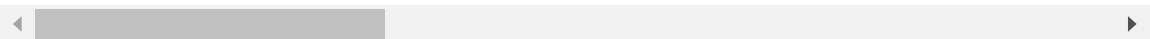
In [23]:

```
x.head()
```

Out[23]:

	dl_applied	win_by_runs	win_by_wickets	team1_Delhi Capitals	team1_Gujarat Lions	team1_Kings XI Punjab	team1_Kochi Tuskers Kerala	team1_K Knight
0	0	35	0	0	0	0	0	
1	0	0	7	0	0	0	0	
2	0	0	10	0	1	0	0	
3	0	0	6	0	0	0	0	
4	0	15	0	0	0	0	0	

5 rows × 44 columns



**\*Label encoding\***

In [24]:

```
le = LabelEncoder()
y = le.fit_transform(y)
```

In [25]:

```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.2, random_state=5)
```

**\*Model training\***

In [26]:

```
model = RandomForestClassifier(n_estimators=100,min_samples_split=5,
                             max_features = "auto")
```

In [27]:

```
model.fit(x_train, y_train)
```

/usr/local/lib/python3.8/dist-packages/sklearn/ensemble/\_forest.py:424: FutureWarning: `max\_features='auto'` has been deprecated in 1.1 and will be removed in 1.3. To keep the past behaviour, explicitly set `max\_features='sqrt'` or remove this parameter as it is also the default value for RandomForestClassifiers and ExtraTreesClassifiers.

Out[27]:

RandomForestClassifier(max\_features='auto', min\_samples\_split=5)

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.**

**On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

In [28]:

```
y_pred = model.predict(x_test)
```

In [29]:

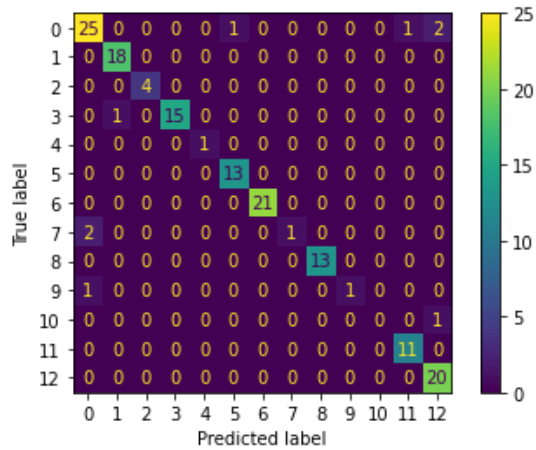
```
accuracy = accuracy_score(y_pred, y_test)
accuracy
```

Out[29]: 0.9707837730042105

### Confusion Matrix

In [30]:

```
confusion_matrix = confusion_matrix(y_test, y_pred)
cm = ConfusionMatrixDisplay(confusion_matrix=confusion_matrix)
cm.plot()
plt.show()
```



### \*Conclusion\*

This project shows the implementation of the IPL Win prediction model. You have got an insight into how to analyze a given raw data and convert that into useful features by removing unwanted features, that is, performing exploratory data analysis.