ANALYSIS OF IPL TEAMS(2008-2020) AND TOSS RECORDS

This is a workbook where:

- **all the tosses in IPL**
- **teams winning the toss**
- **results of the matches**
- **toss decisions at different venues**
- **prediction of a new toss**

have been analysed and studied.

1.) THE DATA VISUALISATION

THE SAMPLE OF DATA USED FOR ANALYSIS

In [260…

```python
#importing the necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.ticker import MultipleLocator
#importing the excel file
md=pd.read_excel(r"C:\Users\lenovo\OneDrive\Desktop\IPL.xlsx")
md=md.dropna()
md.replace("Delhi Daredevils","Delhi Capitals", inplace=True)
print("TOTAL NUMBER OF MATCHES CONSIDERED FOR THE ANALYSIS",md.shape[0])
md.head()
```

```
TOTAL NUMBER OF MATCHES CONSIDERED FOR THE ANALYSIS 815
```

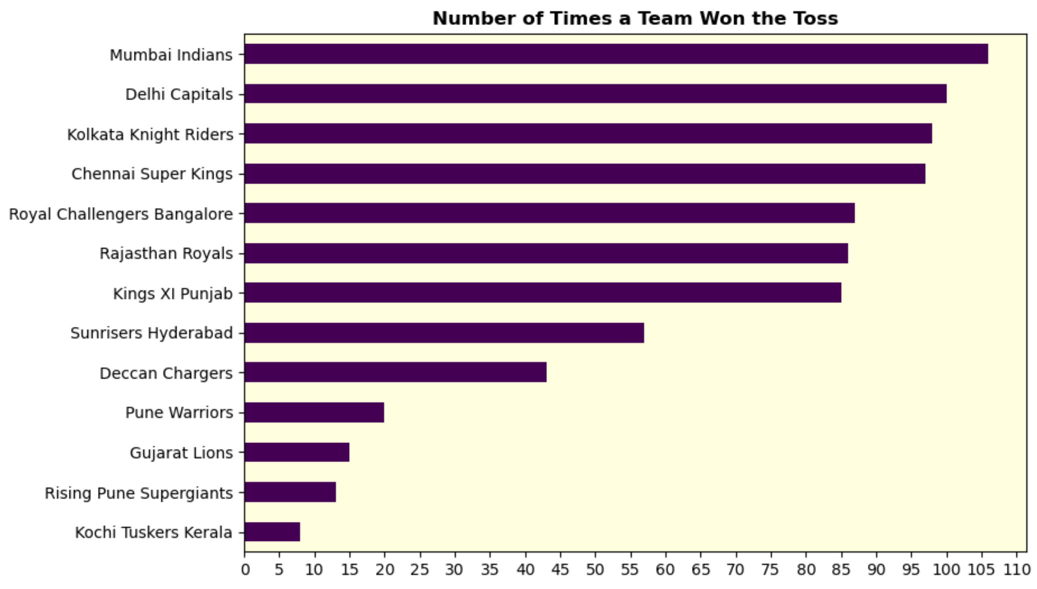
Out[260]:

	date	venue	neutral_venue	team1	team2	toss_winner	toss_decision	winner
0	2008-04-18	M Chinnaswamy Stadium	0	Royal Challengers Bangalore	Kolkata Knight Riders	Royal Challengers Bangalore	field	Kolkata Knight Riders
1	2008-04-19	Punjab Cricket Association Stadium, Mohali	0	Kings XI Punjab	Chennai Super Kings	Chennai Super Kings	bat	Chennai Super Kings
2	2008-04-19	Feroz Shah Kotla	0	Delhi Capitals	Rajasthan Royals	Rajasthan Royals	bat	Delhi Capitals
3	2008-04-20	Wankhede Stadium	0	Mumbai Indians	Royal Challengers Bangalore	Mumbai Indians	bat	Royal Challengers Bangalore
4	2008-04-20	Eden Gardens	0	Kolkata Knight Riders	Deccan Chargers	Deccan Chargers	bat	Kolkata Knight Riders

NUMBER OF TIMES EACH TEAM HAVE WON THE TOSS

In [261…

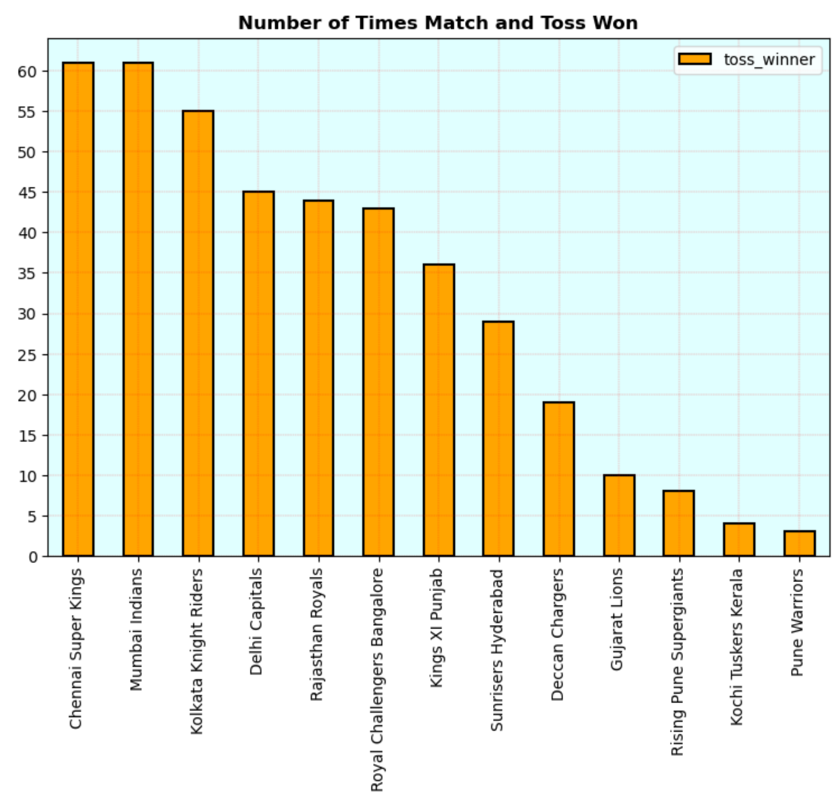
```python
import matplotlib.pyplot as plt
plt.rcParams['figure.figsize']=(9,6)
plt.rcParams['figure.facecolor']="white"
ax=plt.axes()
ax.set_facecolor("lightyellow")
#counting the values of toss winners
new=md["toss_winner"].value_counts().sort_values()
#plotting a bar graph(horizontal)
new.plot( kind="barh",colormap= 'viridis')
plt.gca().xaxis.set_major_locator(MultipleLocator(5))
plt.title("Number of Times a Team Won the Toss", fontweight="bold")
plt.show()
```



NUMBER OF TIMES TEAMS WINNING TOSS AND WINNING MATCH

In [262…

```python
tosswin=md[md['toss_winner'] == md['winner']]
#specifing the size of figure
plt.rcParams['figure.figsize']=(9,6)
#adding a bgcolour to graph
ax=plt.axes()
ax.set_facecolor("lightcyan")
tosswin['toss_winner'].value_counts().plot(kind='bar',edgecolor='black', linewidth=1.5, color='orange')
plt.grid(True, linestyle="--", alpha= 0.5, color="red", linewidth=0.3)
#decreasing the ticks size from default(10) to 5 (ticks means the units division)
plt.gca().yaxis.set_major_locator(MultipleLocator(5))
plt.title("Number of Times Match and Toss Won", fontweight="bold")
plt.legend(loc='upper right')
plt.axis()
plt.show()
#total times a team won the toss and won the match
mu=tosswin['winner'].value_counts()
cu=(mu/new*100).to_frame()
cu.columns=["Win Percentage"]
#toss win to match win percentage ratio
print("Toss win to match win percentage for each team\n")
cu
```



```
Toss win to match win percentage for each team
```

Out[262]:

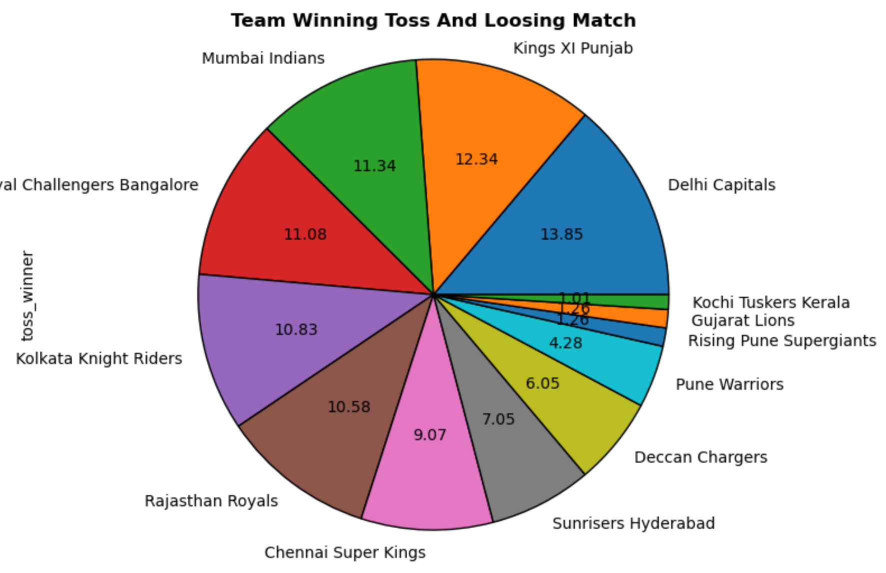
	Win Percentage
Chennai Super Kings	62.886598
Deccan Chargers	44.186047
Delhi Capitals	45.000000
Gujarat Lions	66.666667
Kings XI Punjab	42.352941
Kochi Tuskers Kerala	50.000000
Kolkata Knight Riders	56.122449
Mumbai Indians	57.547170
Pune Warriors	15.000000
Rajasthan Royals	51.162791
Rising Pune Supergiants	61.538462
Royal Challengers Bangalore	49.425287
Sunrisers Hyderabad	50.877193

NUMBER OF TIMES TEAMS WINNING TOSS AND NOT WINNING MATCH

In [263…

```
tossloss=md[md['toss_winner'] != md['winner']]
tossloss['toss_winner'].value_counts().plot(kind='pie',autopct='%.2f', wedgeprops={'edgecolor':"black"}, startangle=0)
plt.title("Team Winning Toss And Loosing Match", fontweight="bold")
plt.axis('equal')
plt.show()
#total times a team won the toss and lost the match
los= tossloss['toss_winner'].value_counts().to_frame()
los.columns=["Losing Match"]

print("\n")
los
```

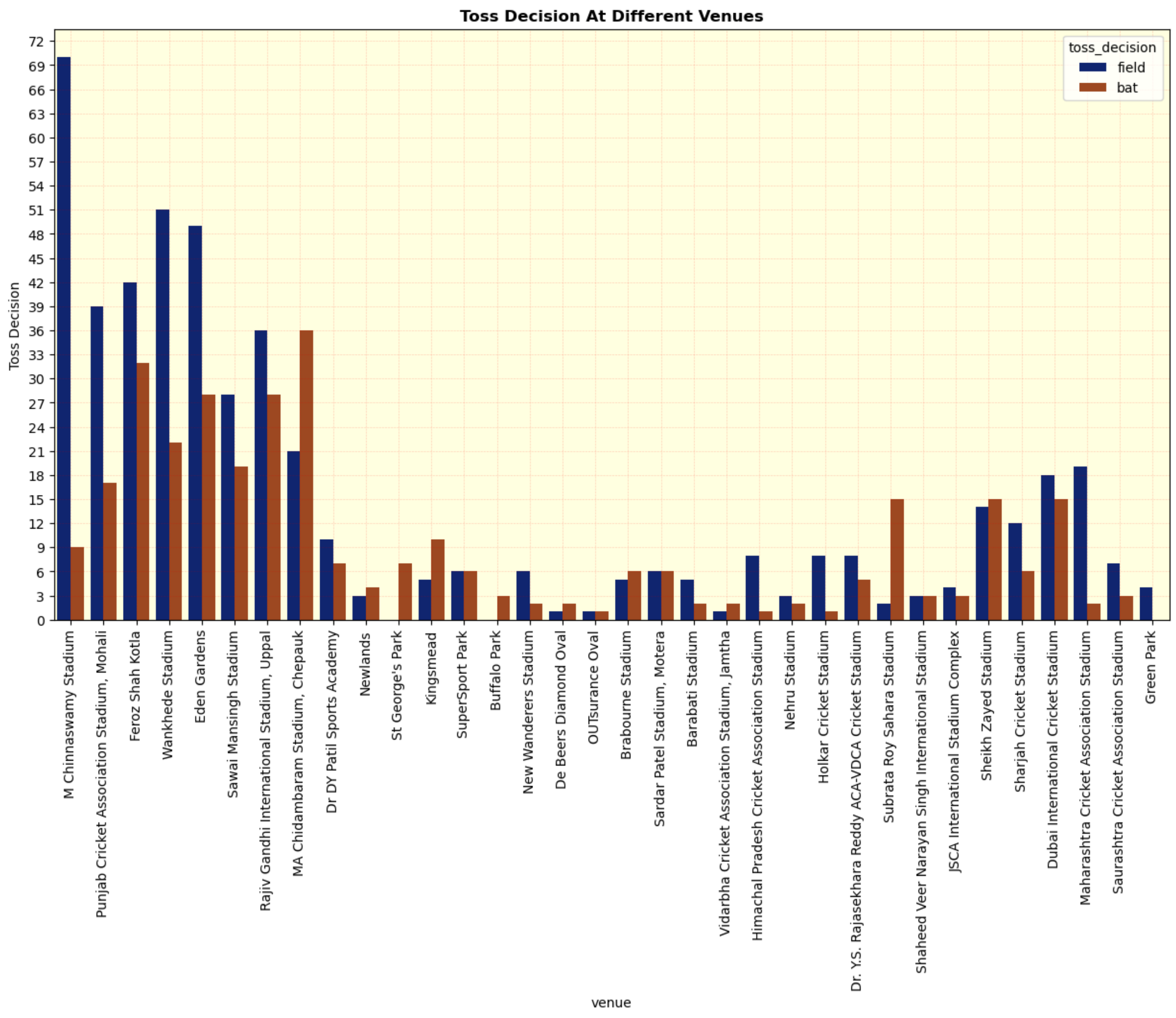


Out[263]:

	Losing Match
Delhi Capitals	55
Kings XI Punjab	49
Mumbai Indians	45
Royal Challengers Bangalore	44
Kolkata Knight Riders	43
Rajasthan Royals	42
Chennai Super Kings	36
Sunrisers Hyderabad	28
Deccan Chargers	24
Pune Warriors	17
Rising Pune Supergiants	5
Gujarat Lions	5
Kochi Tuskers Kerala	4

DIFFERENT TOSS DECISIONS AT DIFFERENT VENUES

In [264…
```
from matplotlib.figure import Figure
#another way of adding background figure
fig, ax = plt.subplots(figsize=(15, 8))
ax.set_facecolor("lightyellow")
sns.countplot(x=md["venue"],hue=md["toss_decision"],data=md,palette='dark')
plt.gca().yaxis.set_major_locator(MultipleLocator(3))
plt.xticks(rotation=90)
plt.ylabel("Toss Decision")
plt.grid(True, linestyle=":",color="red", alpha=0.4, linewidth=0.4)
plt.title("Toss Decision At Different Venues", fontweight="bold")
plt.show()
```



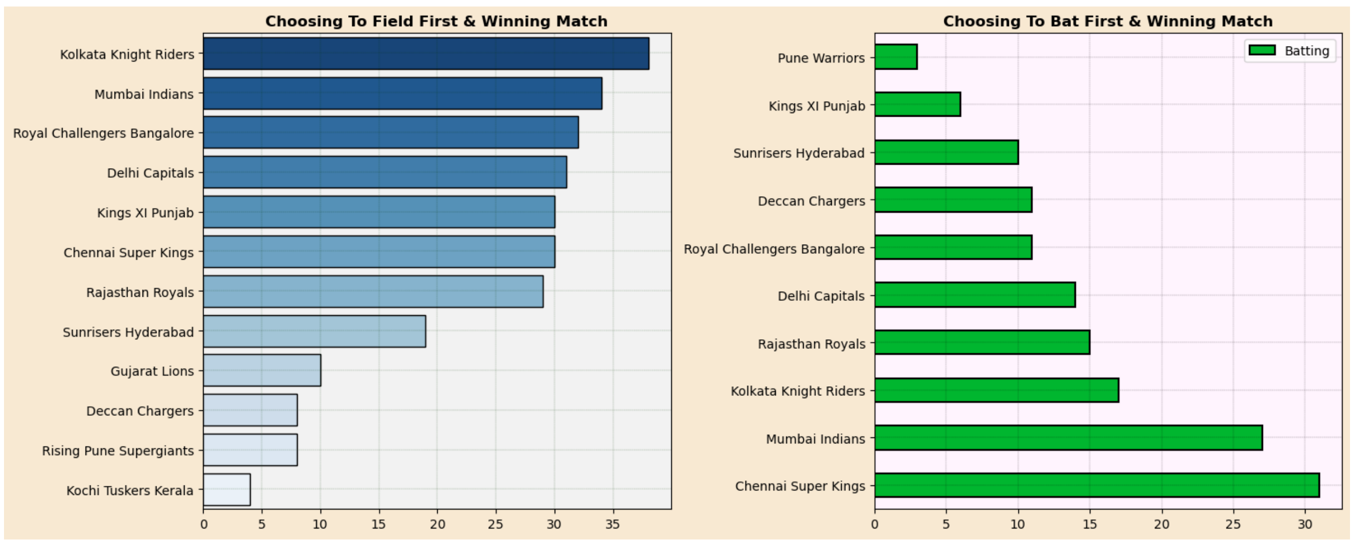
We see that amomg all the stadiums(hosted 5 or more matches):

- ***MA CHIDAMABARAM STADIUM, CHEPAUK***
- ***KINGSMEAD STADIUM, SOUTH AFRICA***
- ***SHEIKH ZAYED STADIUM, ABU DHABI***
- ***SUBRATA ROY SAHARA STADIUM***
- ***BRABOURNE STADIUM***

are the only stadiums where teams preferred to Bat first after winning toss.

DIFFERENT TOSS DECISIONS AT WINNING CAUSES

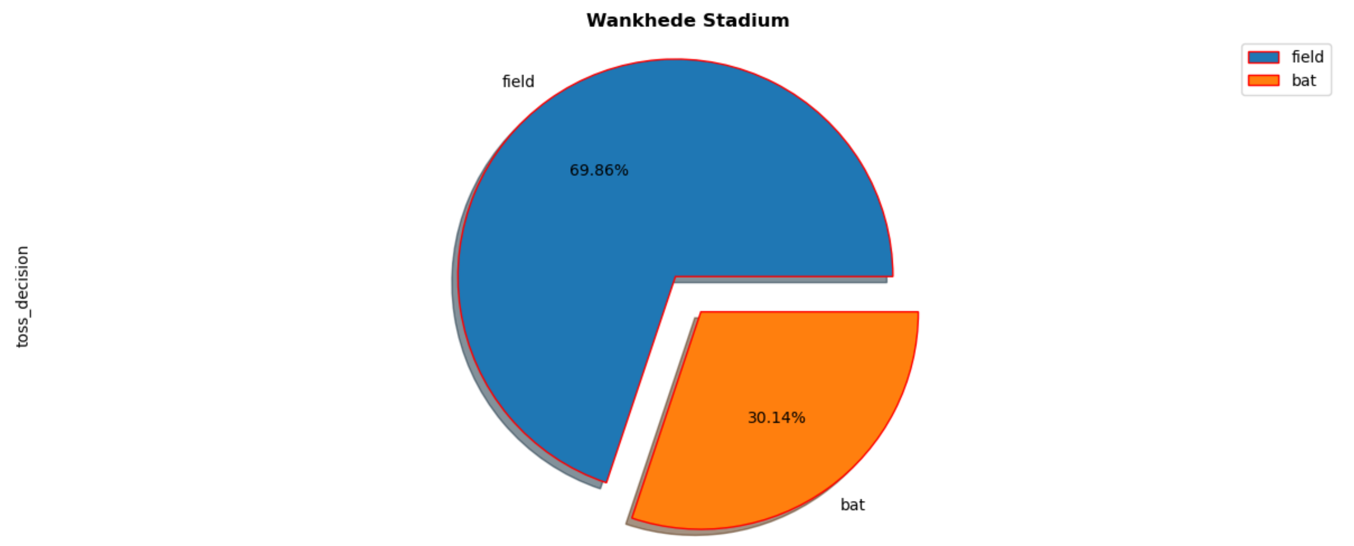
In [265…
```
towin=md[md['toss_winner'] == md['winner']]
feilding=towin[towin["toss_decision"]=="field"]
sep=feilding["toss_winner"].value_counts()
batting=towin[towin["toss_decision"]=="bat"]
plt.rcParams['figure.figsize']=(15,6)
plt.rcParams['figure.facecolor']="#f8e9d2"
plt.subplot(1, 2, 1)
plt.gca().set_facecolor('#F2F2F2')
#using the heat map colour palette
palette = sns.color_palette("Blues_r", n_colors=sep.shape[0])
plt.title("Choosing To Field First & Winning Match", fontweight="bold")
plt.grid(True, linestyle="--", alpha= 0.5, color="red", linewidth=0.3)
palette = dict(zip(sep.index, palette))
sns.barplot(y=sep.index, x=sep.values,palette=palette,edgecolor="black")
plt.grid(True, linestyle="--", alpha= 0.5, color="#073b00", linewidth=0.3)
plt.subplot(1, 2, 2)
plt.gca().set_facecolor("#fff4fe")
batting=towin[towin["toss_decision"]=="bat"]
batting["toss_winner"].value_counts().plot(kind="barh",edgecolor='black', linewidth=1.5, color= "#00b62f")
plt.grid(True, linestyle="--", alpha= 0.5, color="black", linewidth=0.3)
plt.legend(("Batting",))
plt.title("Choosing To Bat First & Winning Match", fontweight="bold")
plt.tight_layout()
plt.show()
```



TOSS DECISION AT ANY PARTICULAR GROUND(USER INPUT)

In [266…

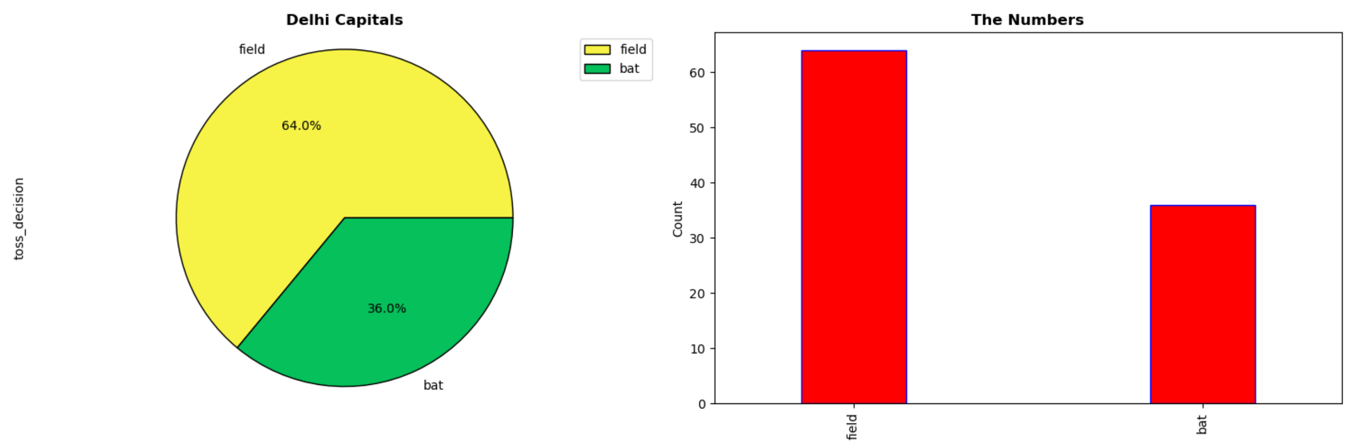
```
plt.rcParams['figure.facecolor']="#ffffff"
ad=eval(input("Enter the stadium name"))
wankhede_data = md[md['venue'] == ad]
wankhede_data['toss_decision'].value_counts().plot(kind='pie', autopct='%1.2f%%', explode=[0.1,0.1], shadow=True, wedgeprops={'edgecolor':"red"})
plt.axis('equal')
plt.title(ad, fontweight="bold")
plt.legend()
plt.show()
```



TOSS DECISION BY ANY TEAM (USER INPUT)

In [267…

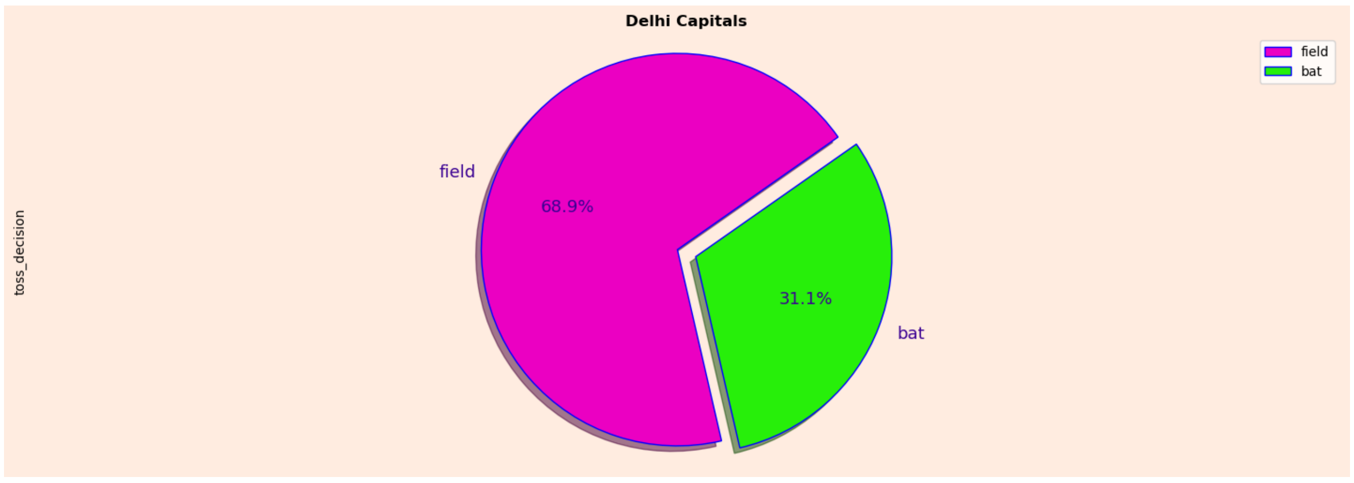
```
ms=eval(input("Enter the Team name"))
plt.rcParams['figure.figsize']=(15,5)
data=md[md['toss_winner'] == ms]
plt.subplot(1, 2, 1)
decision_counts=data['toss_decision'].value_counts()
decision_counts.plot(kind='pie', autopct='%1.1f%%', wedgeprops={"edgecolor":"black"}, colors=["#f6f346","#06c05b"] )
plt.axis("equal")
plt.legend(loc='upper right')
plt.title(ms, fontweight="bold")
plt.subplot(1, 2, 2)
plt.rcParams['figure.figsize']=(5,5)
decision_counts.plot(kind='bar', color="red",width=0.3, edgecolor="blue")
plt.title("The Numbers", fontweight="bold")
plt.ylabel("Count")
plt.tight_layout()
plt.show()
```



TOSS DECISION TAKEN BY THE TEAM IN WINNIG CAUSE(USER INPUT)

In [269…

```
plt.rcParams['figure.figsize']=(18,6)
plt.rcParams["figure.facecolor"]="#ffece0"
data=md[md['toss_winner'] == ms]
decision_win=data[data['winner'] == ms]
decision_win['toss_decision'].value_counts().plot(kind='pie', autopct='%1.1f%%',wedgeprops={"edgecolor":"blue"},shadow=True,colors=["#eb00c2","#27ef0a"],explode=[0.05,
plt.axis('equal')
plt.title(ms,fontweight="bold")
plt.legend(loc='upper right')
plt.show()
```



2.) THE DATA PREPROCESSING

In [252…

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, confusion_matrix
label=LabelEncoder()
lb=LabelEncoder()
md['venue']=lb.fit_transform(md['venue'])
md['team1']=label.fit_transform(md['team1'])
md['team2']=label.fit_transform(md['team2'])
md['toss_winner']=label.fit_transform(md['toss_winner'])
md.tail()
```

Out[252]:

	date	venue	neutral_venue	team1	team2	toss_winner	toss_decision	winner
811	2020-09-28	6	0	11	7	7	field	Royal Challengers Bangalore
812	2020-11-05	6	0	7	2	2	field	Mumbai Indians
813	2020-11-06	28	0	11	12	12	field	Sunrisers Hyderabad
814	2020-11-08	28	0	2	12	2	bat	Delhi Capitals
815	2020-11-10	6	0	2	7	2	bat	Mumbai Indians

Since the date of match does not lead to any major factor for toss decision and the winner of the match is not related to the decision making of toss so these fields are dropped for computational ease and effectiveness.

In [253…

```
new=md.drop(columns=['winner','toss_decision','date'])
med=md['winner']
dec=md['toss_decision']
x_train, x_test, y_train, y_test = train_test_split(new, dec, test_size=0.20, random_state=42)
```

3.) THE MODEL DEPLOYEMENT

K NEAREST NEIGHBOUR CLASSIFIER

In [254…

```
model=KNeighborsClassifier(n_neighbors=7)
model.fit(x_train,y_train)
y_pred=(model.predict(x_test))
accuracy= accuracy_score(y_test,y_pred, normalize=True)
print("The perdiction for toss is with an accuracy of",accuracy*100,"% for K Nearest Classifier")
```

```
The perdiction for toss is with an accuracy of 68.09815950920245 % for K Nearest Classifier
```

DECISION TREE CLASSIFIER

In [255…

```
model1=DecisionTreeClassifier(max_depth=5)
model1.fit(x_train,y_train)
y_pred1=(model1.predict(x_test))
accuracy= accuracy_score(y_test,y_pred1, normalize=True)
print("The perdiction for toss is with an accuracy of",accuracy*100,"% for Decision Tree Classifier")
```

```
The perdiction for toss is with an accuracy of 72.39263803680981 % for Decision Tree Classifier
```

GAUSSIAN NAIVE BAYES CLASSIFIER

In [256…

```
model2=GaussianNB()
model2.fit(x_train,y_train)
y_pred2=(model2.predict(x_test))
accuracy= accuracy_score(y_test,y_pred2, normalize=True)
print("The perdiction for toss is with an accuracy of",accuracy*100,"% for Decision Tree Classifier")
```

```
The perdiction for toss is with an accuracy of 69.93865030674846 % for Decision Tree Classifier
```

The **Decision Tree Classifier** gives more accurate (72.4%) predictions in comparision to other models. So this model is employed for furthur predictions.

4.) THE MODEL PREDICTION

In the following step user needs to enter the required parameters and the predicted result will be shown.

NOTE - {'venue':['Eden Gardens'],'neutral_venue':[0],'team1':['Kolkata Knight Riders'],'team2':['Sunrisers Hyderabad'],'toss_winner':['Sunrisers Hyderabad']} this is the input format which the user can customize.

In [271…

```
imp=eval(input("Enter the match details in given order:({venue:[],neutral_venue:[],team1:[],team2:[],toss_winner:[]})"))
match=pd.DataFrame(imp)
match['venue']=label.fit_transform(match['venue'])
match['team1']=label.fit_transform(match['team1'])
match['team2']=label.fit_transform(match['team2'])
match['toss_winner']=label.fit_transform(match['toss_winner'])
prediction = model.predict(pd.DataFrame(match))
print("\n")
print("The Prediction for toss:",prediction[0])
```

```


The Prediction for toss: field
```

AUTHOR
SIDDHARTH KUMAR CHOUDHARY