```
1 import pandas as pd
2 import numpy as np
3 import seaborn as sns
4 sns.set_style("whitegrid")
5 import matplotlib.pyplot as plt
6 import sklearn
```

```
1 data = pd.read_csv('/content/matches.csv')
```

```
1 data.head()
```

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_
0	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad	
1	2	2017	Pune	2017-04-06	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0	Rising Pune Supergiant	
2	3	2017	Rajkot	2017-04-07	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	0	Kolkata Knight Riders	
3	4	2017	Indore	2017-04-08	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	0	Kings XI Punjab	
4	5	2017	Bangalore	2017-04-08	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	normal	0	Royal Challengers Bangalore	

```
1 data.describe()
```

	id	season	dl_applied	win_by_runs	win_by_wickets
count	756.000000	756.000000	756.000000	756.000000	756.000000
mean	1792.178571	2013.444444	0.025132	13.283069	3.350529
std	3464.478148	3.366895	0.156630	23.471144	3.387963
min	1.000000	2008.000000	0.000000	0.000000	0.000000
25%	189.750000	2011.000000	0.000000	0.000000	0.000000
50%	378.500000	2013.000000	0.000000	0.000000	4.000000
75%	567.250000	2016.000000	0.000000	19.000000	6.000000
max	11415.000000	2019.000000	1.000000	146.000000	10.000000

```
1 data.isnull().sum()
```

```
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          637
dtype: int64
```

```
1 data = data.iloc[:,:-1]
2 data.dropna(inplace=True)
```

```
1 data
```

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner
0	1	2017	Hyderabad	2017-04-05	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad
1	2	2017	Pune	2017-04-06	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0	Rising Pune Supergian
2	3	2017	Rajkot	2017-04-07	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	0	Kolkata Knigh Riders
3	4	2017	Indore	2017-04-08	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	0	Kings X Punjab
5	6	2017	Hyderabad	2017-04-09	Gujarat Lions	Sunrisers Hyderabad	Sunrisers Hyderabad	field	normal	0	Sunrisers Hyderabad
...	...	...	...	...	...	...	...	...	...	...	..
750	11346	2019	Mohali	05/05/19	Chennai Super Kings	Kings XI Punjab	Kings XI Punjab	field	normal	0	Kings X Punjab
751	11347	2019	Mumbai	05/05/19	Kolkata Knight Riders	Mumbai Indians	Mumbai Indians	field	normal	0	Mumba Indians
752	11412	2019	Chennai	07/05/19	Chennai Super Kings	Mumbai Indians	Chennai Super Kings	bat	normal	0	Mumba Indians
754	11414	2019	Visakhapatnam	10/05/19	Delhi Capitals	Chennai Super Kings	Chennai Super Kings	field	normal	0	Chenna Supe Kings
755	11415	2019	Hyderabad	12/05/19	Mumbai Indians	Chennai Super Kings	Mumbai Indians	bat	normal	0	Mumba Indians

743 rows × 17 columns

```
1 data["team1"].unique()
```

```
array(['Sunrisers Hyderabad', 'Mumbai Indians', 'Gujarat Lions',
       'Rising Pune Supergiant', 'Kolkata Knight Riders',
       'Royal Challengers Bangalore', 'Delhi Daredevils',
       'Kings XI Punjab', 'Chennai Super Kings', 'Rajasthan Royals',
       'Deccan Chargers', 'Kochi Tuskers Kerala', 'Pune Warriors',
       'Rising Pune Supergiants', 'Delhi Capitals'], dtype=object)
```

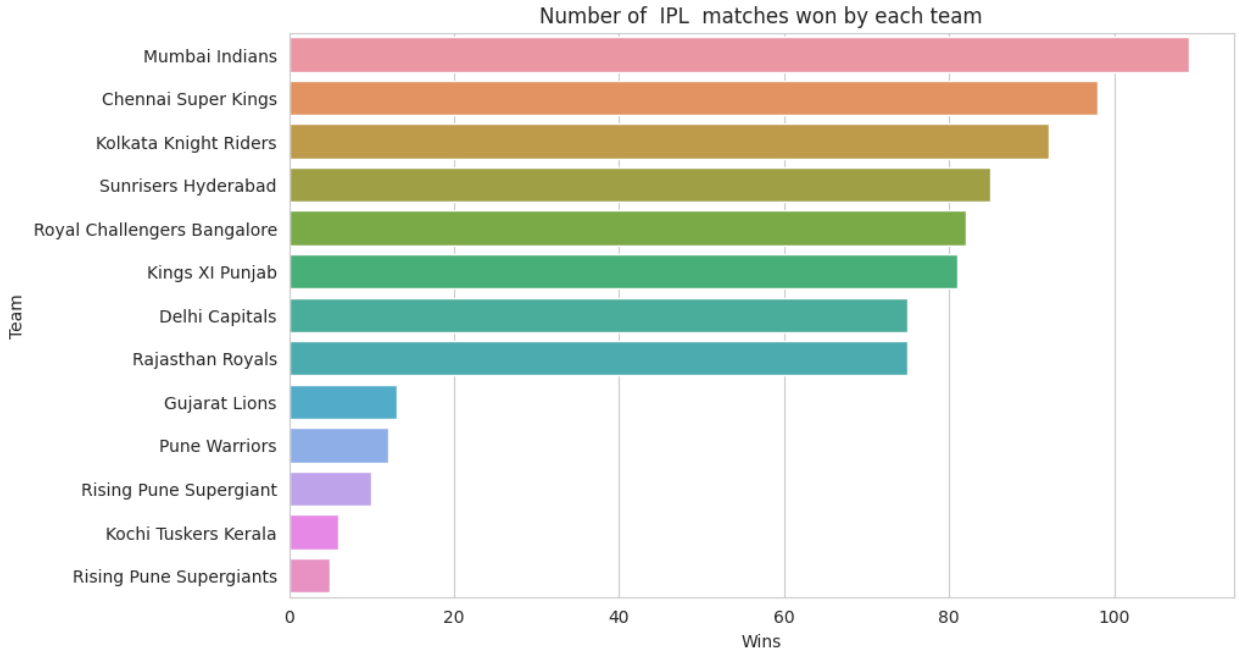
```
1 data['team1']=data['team1'].str.replace('Delhi Daredevils','Delhi Capitals')
2 data['team2']=data['team2'].str.replace('Delhi Daredevils','Delhi Capitals')
3 data['winner']=data['winner'].str.replace('Delhi Daredevils','Delhi Capitals')
```

```
1 data['team1']=data['team1'].str.replace('Deccan Chargers','Sunrisers Hyderabad')
2 data['team2']=data['team2'].str.replace('Deccan Chargers','Sunrisers Hyderabad')
3 data['winner']=data['winner'].str.replace('Deccan Chargers','Sunrisers Hyderabad')
```

```
1 plt.figure(figsize = (10,6))
2 sns.countplot(y = 'winner',data = data,order= data['winner'].value_counts().index)
3 plt.xlabel('Wins')
```

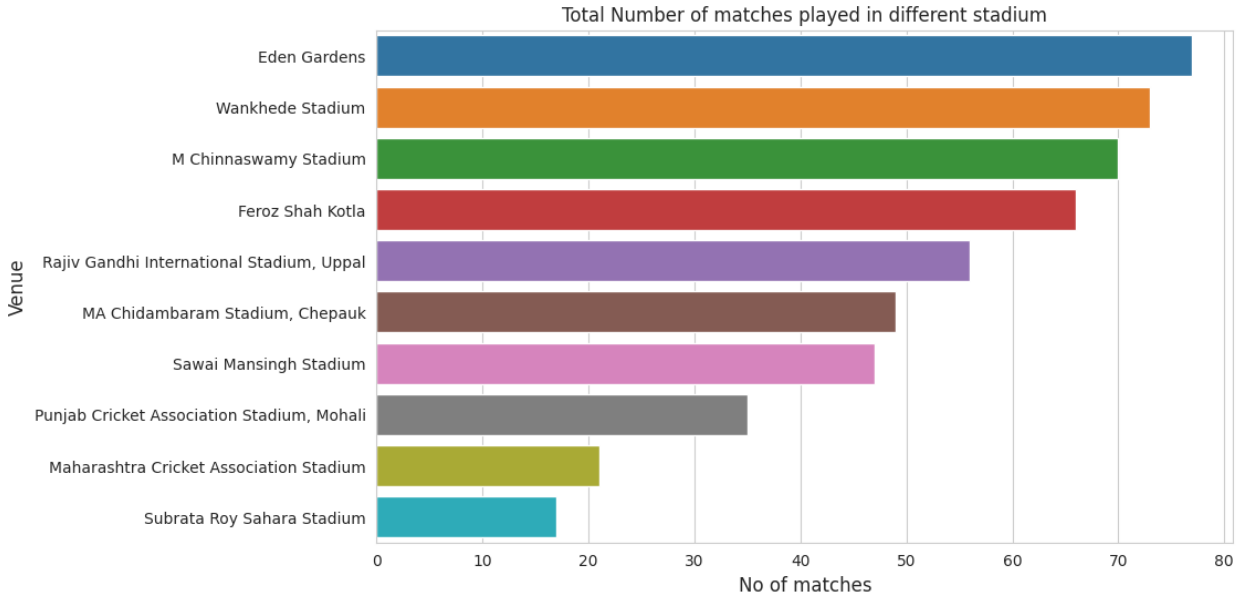
```
4 plt.ylabel('Team')
```

```
Text(0.5, 1.0, 'Number of  IPL  matches won by each team')
```



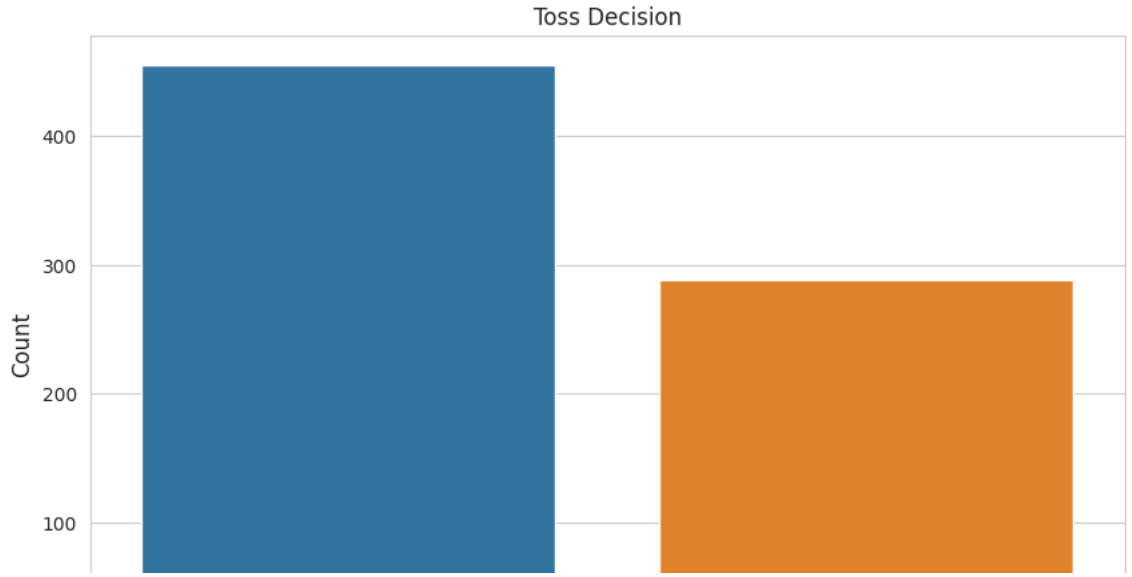

```
1 plt.figure(figsize = (10,6))
2 sns.countplot(y = 'venue',data = data,order = data['venue'].value_counts().iloc[:10].index)
3 plt.xlabel('No of matches',fontsize=12)
4 plt.ylabel('Venue',fontsize=12)
5 plt.title('Total Number of matches played in different stadium')
```

```
Text(0.5, 1.0, 'Total Number of matches played in different stadium')
```




```
1 plt.figure(figsize = (10,6))
2 sns.countplot(x = "toss_decision", data=data)
3 plt.xlabel('Toss Decision',fontsize=12)
4 plt.ylabel('Count',fontsize=12)
5 plt.title('Toss Decision')
```

```
Text(0.5, 1.0, 'Toss Decision')
```




```
1 x = ["city", "toss_decision", "result", "dl_applied"]
2 for i in x:
3   print("------------")
4   print(data[i].unique())
5   print(data[i].value_counts())
```

```
------------
['Hyderabad' 'Pune' 'Rajkot' 'Indore' 'Mumbai' 'Kolkata' 'Bangalore'
 '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' 'Mohali'
 'Bengaluru']
Mumbai            101
Kolkata            77
Delhi              73
Hyderabad          64
Bangalore          63
Chennai            57
Jaipur             47
Chandigarh         46
Pune               38
Durban             15
Bengaluru          13
Centurion          12
Ahmedabad          12
Visakhapatnam      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    455
bat      288
Name: toss_decision, dtype: int64
------------
['normal' 'tie']
normal    734
tie         9
Name: result, dtype: int64
```

```
------------
[0 1]
0    724
1     19
Name: dl_applied, dtype: int64
```

```
data.drop(["id", "season","city","date", "player_of_match", 'umpire1', "venue", "umpire2"], axis=1, inplace=True)
```

```
X = data.drop(["winner"], axis=1)
y = data["winner"]
```

```
X = pd.get_dummies(X, ["team1","team2", "toss_winner", "toss_decision", "result"], drop_first = True)
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, y, train_size = 0.8)
```

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=200,min_samples_split=3,
                               max_features = "auto")
```

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

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'` has been dep
  warn(
```

RandomForestClassifier

```
RandomForestClassifier(max_features='auto', min_samples_split=3,
                       n_estimators=200)
```

```
y_pred=model.predict(x_test)
from sklearn.metrics import accuracy_score
ac=accuracy_score(y_pred, y_test)
ac
y_pred
```

```
array([ 3, 11, 12,  6,  0,  1,  6,  6, 12,  6, 12, 12,  6, 11,  8,  3,  7,
        0,  1,  8,  6,  3, 12,  0,  1,  6, 12,  3,  0,  5, 12,  3, 11,  0,
        5,  8,  3,  3,  5,  5,  8,  6,  0,  8, 11, 12, 11,  5,  9,  8,  6,
       12, 12,  0,  5, 11,  5,  8,  8,  3,  3, 11,  1,  5,  0,  0,  0,  3,
        3, 11, 11,  5, 12,  0,  6,  0,  6,  6, 11,  6, 12, 12,  5, 12,  1,
        0,  0,  5,  0, 11,  1, 12,  6, 12,  3,  3,  0,  3,  0,  3,  6,  6,
       12,  0,  0,  3,  0,  1,  5,  1, 11,  0,  5,  1,  3,  1,  2,  6,  6,
       12,  3,  0,  3,  8,  1, 11,  3,  6,  4,  1,  6,  1, 11,  6,  8, 12,
        6,  5, 11,  0,  2, 11,  6,  5,  6,  1,  5,  6,  5])
```

```
from sklearn.metrics import make_scorer, f1_score, precision_score, recall_score, accuracy_score

scorers = {
    'f1_score': make_scorer(f1_score, average='micro'),
    'precision_score': make_scorer(precision_score, average='micro'),
    'recall_score': make_scorer(recall_score, average='micro'),
    'accuracy_score': make_scorer(accuracy_score)
}

```

```
x_test.head()
```

	dl_applied	win_by_runs	win_by_wickets	team1_Delhi Capitals	team1_Gujarat Lions	team1_Kings XI Punjab	team1_Kochi Tuskers Kerala	team1_Kolkata Knight Riders	team1_M In
603	0	23	0	0	0	1	0	0	
391	0	0	8	0	0	0	0	1	
393	0	0	3	1	0	0	0	0	

```
from sklearn.ensemble import RandomForestClassifier

# Create and train the RandomForestClassifier
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(x_train, y_train)
```

```
RandomForestClassifier(random_state=42)
```

```
from sklearn.metrics import accuracy_score, classification_report

y_pred = clf.predict(x_test)  # X_test contains feature data for the test set
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print("Accuracy:", accuracy)
print("Classification Report:\n", report)

```

```
).9261744966442953
:ion Report:
     precision    recall  f1-score   support

 0       0.77      0.94      0.85        18
 1       1.00      0.86      0.92        14
 2       1.00      0.67      0.80         3
 3       0.95      0.95      0.95        20
 4       1.00      0.67      0.80         3
 5       1.00      0.94      0.97        18
 6       0.96      1.00      0.98        24
 7       1.00      0.50      0.67         2
 8       0.91      1.00      0.95        10
 9       1.00      1.00      1.00         1
10       0.00      0.00      0.00         2
11       1.00      0.94      0.97        17
12       0.85      1.00      0.92        17

:y                           0.93       149
/g       0.88      0.81      0.83       149
/g       0.92      0.93      0.92       149

'lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and
f(average, modifier, msg_start, len(result))
'lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and
f(average, modifier, msg_start, len(result))
'lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and
f(average, modifier, msg_start, len(result))
```

```
upcoming_match = {
    'team1': 'Mumbai Indians',
    'team2': 'Kolkata Knight Riders',
    'toss_winner': 'Mumbai Indians',
    'toss_decision': 'field',
    'result': 'normal',
    'dl_applied': 0,
    'win_by_runs': 10,
    'win_by_wickets': 0
}

```

```
best_clf = clf
```

```
 import pandas as pd
```

```
from sklearn.ensemble import RandomForestClassifier

# Load your trained model (use the same model you trained earlier)
model = RandomForestClassifier(n_estimators=200, min_samples_split=3, max_features="auto")

# Assuming you have the dataset loaded in 'data' (you can adjust this as needed)
# Prepare the dataset (remove unnecessary columns, perform one-hot encoding, etc.)

# Create a DataFrame from the upcoming match details
upcoming_match_df = pd.DataFrame([upcoming_match])

# Perform one-hot encoding on the upcoming match DataFrame
upcoming_match_df = pd.get_dummies(upcoming_match_df, columns=["team1", "team2", "toss_winner", "toss

# Ensure all columns in the training data are present in the test data
missing_columns = set(x_train.columns) - set(upcoming_match_df.columns)
for col in missing_columns:
    upcoming_match_df[col] = 0

# Reorder columns to match the order in x_train
upcoming_match_df = upcoming_match_df[x_train.columns]

# Fit the RandomForestClassifier model with your training data
model.fit(x_train, y_train)

# Make predictions for the upcoming match
predicted_winner_index = model.predict(upcoming_match_df)[0]

# Assuming you have a label encoder 'le' to transform predictions back to team names
predicted_winner = le.inverse_transform([predicted_winner_index])[0]

print("Predicted Winner:", predicted_winner)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning: `max_features='auto'` has been deprecated in 1.
  warn(
Predicted Winner: Mumbai Indians
```

```
```