In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn
```

In [3]:
```
df = pd.read_csv("matches.csv")
```

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

Out[4]:

	id	Season	city	date	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by_wickets	player_of_match	venue	umpire1	umpire2	umpire3
0	1	IPL-2017	Hyderabad	05-04-2017	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad	35	0	Yuvraj Singh	Rajiv Gandhi International Stadium, Uppal	AY Dandekar	NJ Llong	NaN
1	2	IPL-2017	Pune	06-04-2017	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0	Rising Pune Supergiant	0	7	SPD Smith	Maharashtra Cricket Association Stadium	A Nand Kishore	S Ravi	NaN
2	3	IPL-2017	Rajkot	07-04-2017	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	0	Kolkata Knight Riders	0	10	CA Lynn	Saurashtra Cricket Association Stadium	Nitin Menon	CK Nandan	NaN
3	4	IPL-2017	Indore	08-04-2017	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	0	Kings XI Punjab	0	6	GJ Maxwell	Holkar Cricket Stadium	AK Chaudhary	C Shamshuddin	NaN
4	5	IPL-2017	Bangalore	09-04-2017	Royal Challengers Bangalore	Delhi Daredevils	Royal Challengers Bangalore	bat	normal	0	Royal Challengers Bangalore	15	0	KM Jadhav	M Chinnaswamy Stadium	NaN	NaN	NaN

In [5]:
```
df.shape
```

Out[5]:
```
(756, 18)
```

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	0.156530	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.isnull().sum() #echaecking is there any null values or not
```

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         637
dtype: int64
```

In [8]:
```
df.drop("umpire3",axis=1,inplace=True)
df.dropna(inplace=True)
```

In [9]:
```
df["team1"].unique()
```

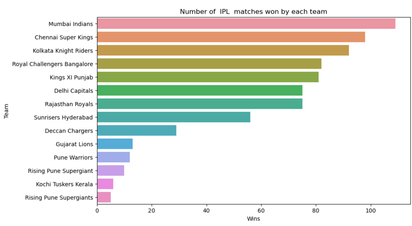
Out[9]:
```
array(['Sunrisers Hyderabad', 'Mumbai Indians', 'Gujarat Lions',
       'Rising Pune Supergiant', 'Kolkata Knight Riders',
       'Royal Challengers Bangalore', 'Delhi Daredevils',
       'Kings XI Punjab', 'Chennai Super Kings', 'Rajasthan Royals',
       'Deccan Chargers', 'Kochi Tuskers Kerala', 'Pune Warriors',
       'Rising Pune Supergiants', 'Delhi Capitals'], dtype=object)
```

Here you can see the name Delhi Daredevils and Delhi Capitals; Delhi Daredevils is the old name of the Delhi Capitals. Similarly, Decan Chargers is the old name of Sunrisers Hyderabad. So we are changing the old name to the newer one.

In [10]:
```
#for Delhi Capitals
df['team1'].str.replace('Delhi Daredevils','Delhi Capitals')
df['team2'].str.replace('Delhi Daredevils','Delhi Capitals')
df['winner'].str.replace('Delhi Daredevils','Delhi Capitals')
#for sunrisers Hyderabad
df['team1'].str.replace('Deccan Chargers','Sunrisers Hyderabad')
df['team2'].str.replace('Deccan Chargers','Sunrisers Hyderabad')
```

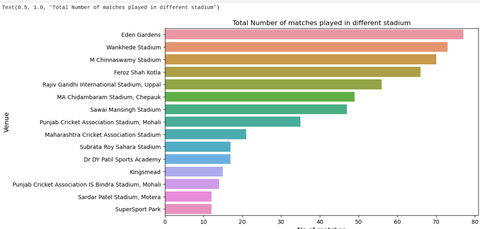
In [11]:
```
plt.figure(figsize = (10,6))
sns.countplot(y = 'winner',data = df,order= df['winner'].value_counts().index)
plt.xlabel('Wins')
plt.ylabel('Team')
plt.title('Number of  IPL  matches won by each team')
```

Out[11]:
```
Text(0.5, 1.0, 'Number of  IPL  matches won by each team')
```



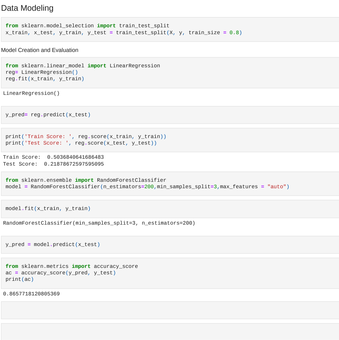
In [12]:
```
plt.figure(figsize = (10,6))
sns.countplot(y = 'venue',data = df,order= df['venue'].value_counts().iloc[:15].index)
plt.xlabel('No of matches',fontsize=12)
plt.ylabel('Venue',fontsize=12)
plt.title('Total Number of matches played in different stadium')
```

Out[12]:
```
Text(0.5, 1.0, 'Total Number of matches played in different stadium')
```



In [13]:
```
plt.figure(figsize = (10,6))
sns.countplot(x = "toss_decision", data=df)
plt.xlabel('Toss Decision',fontsize=12)
plt.ylabel('Count',fontsize=12)
plt.title('Toss Decision')
```

Out[13]:
```
Text(0.5, 1.0, 'Toss Decision')
```



In [14]:
```
x = ["city", "toss_decision", "result", "dl_applied"]
for i in x:
  print("----------")
  print(df[i].unique())
  print(df[i].value_counts())
```

Out[14]:
```
----------
['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          15
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    724
tie         9
Name: result, dtype: int64
----------
[0 1]
0    724
1     19
Name: dl_applied, dtype: int64
```

We don't need all the features or columns in order to create the model. It will reduce model accuracy, so we are dropping some of the features that don't affect our result.

In [15]:
```
df.drop(["id", "Season","city","date", "player_of_match", "umpire1", "venue", "umpire2"], axis=1, inplace=True)
```

In [16]:
```
df.head()
```

Out[16]:

	team1	team2	toss_winner	toss_decision	result	dl_applied	winner	win_by_runs	win_by_wickets
0	Sunrisers Hyderabad	Royal Challengers Bangalore	Royal Challengers Bangalore	field	normal	0	Sunrisers Hyderabad	35	0
1	Mumbai Indians	Rising Pune Supergiant	Rising Pune Supergiant	field	normal	0	Rising Pune Supergiant	0	7
2	Gujarat Lions	Kolkata Knight Riders	Kolkata Knight Riders	field	normal	0	Kolkata Knight Riders	0	10
3	Rising Pune Supergiant	Kings XI Punjab	Kings XI Punjab	field	normal	0	Kings XI Punjab	0	6
5	Gujarat Lions	Sunrisers Hyderabad	Sunrisers Hyderabad	field	normal	0	Sunrisers Hyderabad	0	9

In [17]:
```
X = df.drop(["winner"], axis=1)
y = df["winner"]
```

Several categorical values are present in the input data, so we are converting them into numerical values using the pandas, get_dummies method.

In [18]:
```
X = pd.get_dummies(X, ["team1","team2", "toss_winner", "toss_decision", "result"], drop_first = True)
```

The output data is also a categorical value, so we are converting it into numerical using LabelEncoder of sklearn.

In [19]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
```

Data Modeling

In [20]:
```
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)
```

Model Creation and Evaluation

In [21]:
```
from sklearn.linear_model import LinearRegression
regr LinearRegression()
reg.fit(x_train, y_train)
```

Out[21]:
```
LinearRegression()
```

In [22]:
```
y_pred= reg.predict(x_test)
```

In [23]:
```
print("Train Score: ", reg.score(x_train, y_train))
print("Test Score: ", reg.score(x_test, y_test))
```

Out[23]:
```
Train Score:  0.5036840641686483
Test Score:  0.21878672597595095
```

In [24]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=200,min_samples_split=3,max_features = "auto")
```

In [25]:
```
model.fit(x_train, y_train)
```

Out[25]:
```
RandomForestClassifier(min_samples_split=3, n_estimators=200)
```

In [26]:
```
y_pred = model.predict(x_test)
```

In [27]:
```
from sklearn.metrics import accuracy_score
ac = accuracy_score(y_pred, y_test)
print(ac)
```

Out[27]:
```
0.8657718120805369
```

In [ ]:

In [ ]: