In [165]:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import KFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn import metrics
import warnings
warnings.filterwarnings('ignore')
```

In [166]:

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 636 entries, 0 to 635
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   id               636 non-null    int64  
 1   season           636 non-null    int64  
 2   city             629 non-null    object 
 3   date             636 non-null    object 
 4   team1            636 non-null    object 
 5   team2            636 non-null    object 
 6   toss_winner      636 non-null    object 
 7   toss_decision    636 non-null    object 
 8   result           636 non-null    object 
 9   dl_applied       636 non-null    int64  
 10  winner           633 non-null    object 
 11  win_by_runs      636 non-null    int64  
 12  win_by_wickets   636 non-null    int64  
 13  player_of_match  633 non-null    object 
 14  venue            636 non-null    object 
 15  umpire1          635 non-null    object 
 16  umpire2          635 non-null    object 
 17  umpire3          0 non-null      float64
dtypes: float64(1), int64(5), object(12)
memory usage: 89.6+ KB
```

In [167]:

```
matches[pd.isnull(matches['winner'])]
```

Out[167]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	res
300	301	2011	Delhi	2011-05-21	Delhi Daredevils	Pune Warriors	Delhi Daredevils	bat	res
545	546	2015	Bangalore	2015-04-29	Royal Challengers Bangalore	Rajasthan Royals	Rajasthan Royals	field	res
570	571	2015	Bangalore	2015-05-17	Delhi Daredevils	Royal Challengers Bangalore	Royal Challengers Bangalore	field	res

In [168]:

```
matches['winner'].fillna('Draw', inplace=True)
```

In [169]:

```
matches.replace(['Mumbai Indians','Kolkata Knight Riders','Royal Challengers Bangalore'
,'Deccan Chargers','Chennai Super Kings',
                 'Rajasthan Royals','Delhi Daredevils','Gujarat Lions','Kings XI Punja
b',
                 'Sunrisers Hyderabad','Rising Pune Supergiants','Rising Pune Supergian
t','Kochi Tuskers Kerala','Pune Warriors']
                ,['MI','KKR','RCB','DC','CSK','RR','DD','GL','KXIP','SRH','RPS','RPS',
'KTK','PW'],inplace=True)

encode = {'team1': {'MI':1,'KKR':2,'RCB':3,'DC':4,'CSK':5,'RR':6,'DD':7,'GL':8,'KXIP':9
,'SRH':10,'RPS':11,'KTK':12,'PW':13},
          'team2': {'MI':1,'KKR':2,'RCB':3,'DC':4,'CSK':5,'RR':6,'DD':7,'GL':8,'KXIP':9
,'SRH':10,'RPS':11,'KTK':12,'PW':13},
          'toss_winner': {'MI':1,'KKR':2,'RCB':3,'DC':4,'CSK':5,'RR':6,'DD':7,'GL':8,'K
XIP':9,'SRH':10,'RPS':11,'KTK':12,'PW':13},
          'winner': {'MI':1,'KKR':2,'RCB':3,'DC':4,'CSK':5,'RR':6,'DD':7,'GL':8,'KXIP':
9,'SRH':10,'RPS':11,'KTK':12,'PW':13,'Draw':14}}
matches.replace(encode, inplace=True)
matches.head(2)
```

Out[169]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_appli
0	1	2017	Hyderabad	2017-04-05	10	3	3	field	normal	
1	2	2017	Pune	2017-04-06	1	11	11	field	normal	

In [170]:

```
matches[pd.isnull(matches['city'])]
```

Out[170]:

	id	season	city	date	team1	team2	toss_winner	toss_decision	result	dl_appliec
461	462	2014	NaN	2014-04-19	1	3	3	field	normal	(
462	463	2014	NaN	2014-04-19	2	7	2	bat	normal	(
466	467	2014	NaN	2014-04-23	5	6	6	field	normal	(
468	469	2014	NaN	2014-04-25	10	7	10	bat	normal	(
469	470	2014	NaN	2014-04-25	1	5	1	bat	normal	(
474	475	2014	NaN	2014-04-28	3	9	9	field	normal	(
476	477	2014	NaN	2014-04-30	10	1	1	field	normal	(

In [171]:

```
matches['city'].fillna('Dubai',inplace=True)
matches.describe()
```

Out[171]:

	id	season	team1	team2	toss_winner	dl_applied	winne
count	636.000000	636.000000	636.000000	636.000000	636.000000	636.000000	636.00000
mean	318.500000	2012.490566	5.540881	5.511006	5.371069	0.025157	5.30974
std	183.741666	2.773026	3.329169	3.341677	3.293140	0.156726	3.28872
min	1.000000	2008.000000	1.000000	1.000000	1.000000	0.000000	1.00000
25%	159.750000	2010.000000	3.000000	3.000000	2.000000	0.000000	2.00000
50%	318.500000	2012.000000	5.000000	5.000000	5.000000	0.000000	5.00000
75%	477.250000	2015.000000	9.000000	8.000000	7.250000	0.000000	8.00000
max	636.000000	2017.000000	13.000000	13.000000	13.000000	1.000000	14.00000

In [172]:

```
dicVal = encode['winner']
print(dicVal['MI'])
print(list(dicVal.keys())[list(dicVal.values()).index(1)])
```

```
1
MI
```

In [173]:

```
matches = matches[['team1','team2','city','toss_decision','toss_winner','venue','winne
r']]
matches.head(2)
```

Out[173]:

	team1	team2	city	toss_decision	toss_winner	venue	winner
0	10	3	Hyderabad	field	3	Rajiv Gandhi International Stadium, Uppal	10
1	1	11	Pune	field	11	Maharashtra Cricket Association Stadium	11

In [174]:

```
df = pd.DataFrame(matches)
df.describe()
```

Out[174]:

	team1	team2	toss_winner	winner
count	636.000000	636.000000	636.000000	636.000000
mean	5.540881	5.511006	5.371069	5.309748
std	3.329169	3.341677	3.293140	3.288726
min	1.000000	1.000000	1.000000	1.000000
25%	3.000000	3.000000	2.000000	2.000000
50%	5.000000	5.000000	5.000000	5.000000
75%	9.000000	8.000000	7.250000	8.000000
max	13.000000	13.000000	13.000000	14.000000

In [175]:

```
temp1=df['toss_winner'].value_counts(sort=True)
temp2=df['winner'].value_counts(sort=True)
#Mumbai won most toss and also won most matches
print('No of toss winners by each team')
for idx, val in temp1.iteritems():
   print('{} -> {}'.format(list(dicVal.keys())[list(dicVal.values()).index(idx)],val))
print('No of match winners by each team')
for idx, val in temp2.iteritems():
   print('{} -> {}'.format(list(dicVal.keys())[list(dicVal.values()).index(idx)],val))
```

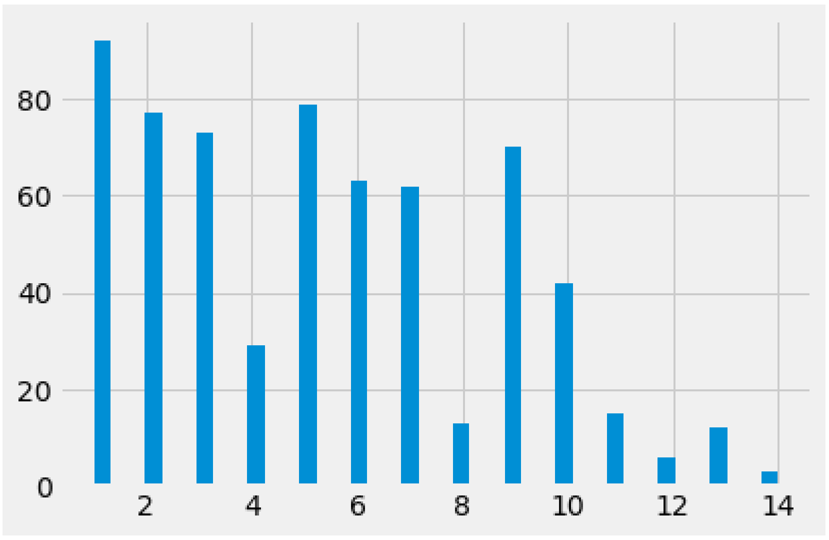
```
No of toss winners by each team
MI -> 85
KKR -> 78
DD -> 72
RCB -> 70
KXIP -> 68
CSK -> 66
RR -> 63
DC -> 43
SRH -> 35
PW -> 20
GL -> 15
RPS -> 13
KTK -> 8
No of match winners by each team
MI -> 92
CSK -> 79
KKR -> 77
RCB -> 73
KXIP -> 70
RR -> 63
DD -> 62
SRH -> 42
DC -> 29
RPS -> 15
GL -> 13
PW -> 12
KTK -> 6
Draw -> 3
```

In [176]:

```
#shows that Mumbai won most matches followed by Chennai
df['winner'].hist(bins=40)
```

Out[176]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1d134c47640>
```



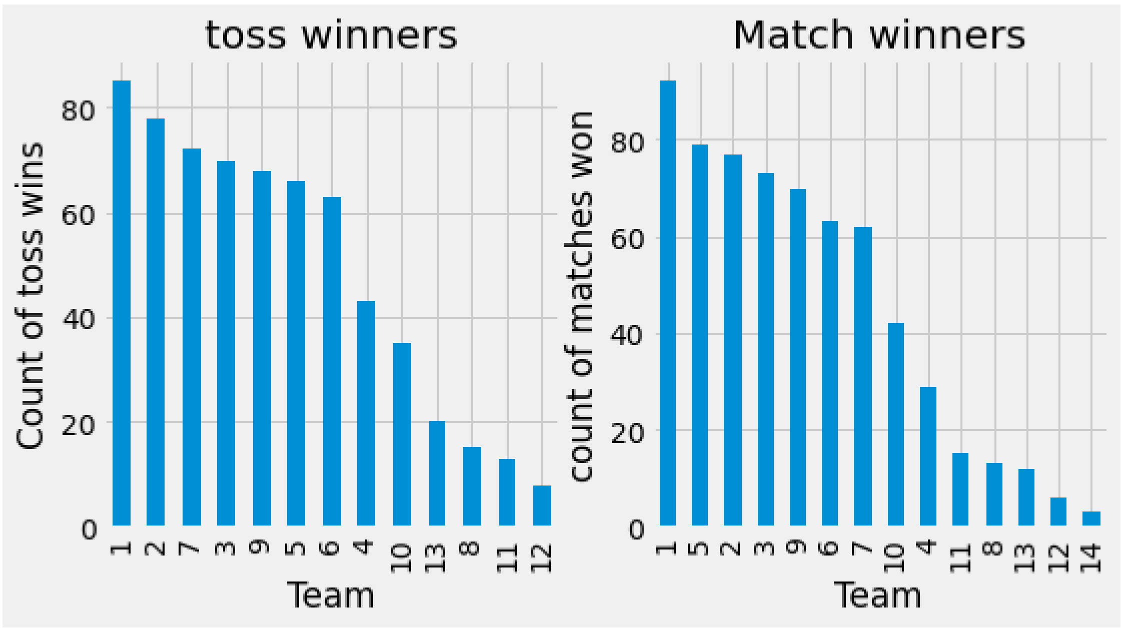
In [177]:

```
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(8,4))
ax1 = fig.add_subplot(121)
ax1.set_xlabel('Team')
ax1.set_ylabel('Count of toss wins')
ax1.set_title("toss winners")
temp1.plot(kind='bar')

ax2 = fig.add_subplot(122)
temp2.plot(kind = 'bar')
ax2.set_xlabel('Team')
ax2.set_ylabel('count of matches won')
ax2.set_title("Match winners")
```

Out[177]:

```
Text(0.5, 1.0, 'Match winners')
```



In [178]:

```
df.apply(lambda x: sum(x.isnull()),axis=0)
```

Out[178]:

```
team1            0
team2            0
city             0
toss_decision    0
toss_winner      0
venue            0
winner           0
dtype: int64
```

In [179]:

```
df[pd.isnull(df['city'])]
```

Out[179]:

team1	team2	city	toss_decision	toss_winner	venue	winner

In [180]:

```
var_mod = ['city','toss_decision','venue']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i])
df.dtypes
```

Out[180]:

```
team1            int64
team2            int64
city             int32
toss_decision    int32
toss_winner      int64
venue            int32
winner           int64
dtype: object
```

In [181]:

```
def classification_model(model, data, predictors, outcome):
  model.fit(data[predictors],data[outcome])
  predictions = model.predict(data[predictors])
  accuracy = metrics.accuracy_score(predictions,data[outcome])
  print('Accuracy : %s' % '{0:.3%}'.format(accuracy))
  kf = KFold(data.shape[0],n_splits=7)
  error = []
  for train, test in kf.split(data[predictors]):
    train_predictors = (data[predictors].iloc[train,:])
    train_target = data[outcome].iloc[train]
    model.fit(train_predictors, train_target)
    error.append(model.score(data[predictors].iloc[test,:], data[outcome].iloc[test]))

  print('Cross-Validation Score : %s' % '{0:.3%}'.format(np.mean(error)))

  model.fit(data[predictors],data[outcome])
```

In [182]:

```
model = RandomForestClassifier(n_estimators=100)
outcome_var = ['winner']
predictor_var = ['team1', 'team2', 'venue', 'toss_winner','city','toss_decision']
classification_model(model, df,predictor_var,outcome_var)
```

```
Accuracy : 89.151%
Cross-Validation Score : 48.899%
```

In [183]:

```
df.head(7)
```

Out[183]:

	team1	team2	city	toss_decision	toss_winner	venue	winner
0	10	3	14	1	3	23	10
1	1	11	25	1	11	16	11
2	8	2	27	1	2	25	2
3	11	9	15	1	9	11	9
4	3	7	2	0	3	14	3
5	8	10	14	1	10	23	10
6	2	1	22	1	1	34	1

In [184]:

```
team1='RCB'
team2='KKR'
toss_winner='RCB'
input=[dicVal[team1],dicVal[team2],'14',dicVal[toss_winner],'2','1']
input = np.array(input).reshape((1, -1))
output=model.predict(input)
print(list(dicVal.keys())[list(dicVal.values()).index(output)]) #find key by value sear
ch output
```

```
KKR
```

In [185]:

```
team1='DC'
team2='DD'
toss_winner='DC'
input=[dicVal[team1],dicVal[team2],'23',dicVal[toss_winner],'14','0']
input = np.array(input).reshape((1, -1))
output=model.predict(input)
print(list(dicVal.keys())[list(dicVal.values()).index(output)]) #find key by value sear
ch output
```

```
DD
```

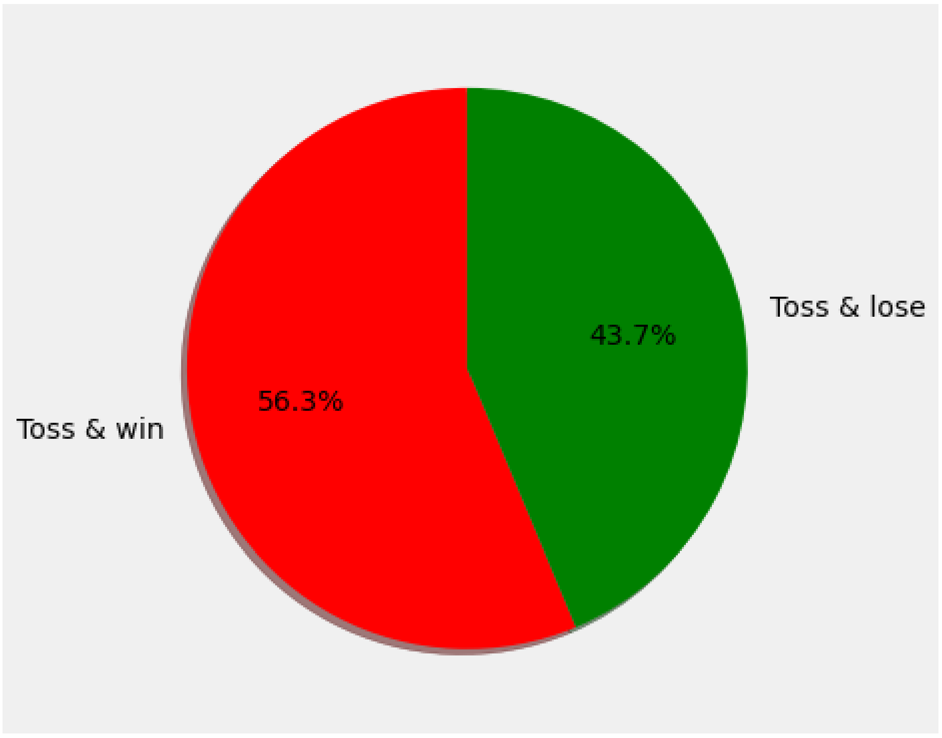
In [186]:

```
imp_input = pd.Series(model.feature_importances_, index=predictor_var).sort_values(asce
nding=False)
print(imp_input)
```

```
team2            0.253574
team1            0.222173
venue            0.168841
toss_winner      0.167574
city             0.154566
toss_decision    0.033272
dtype: float64
```

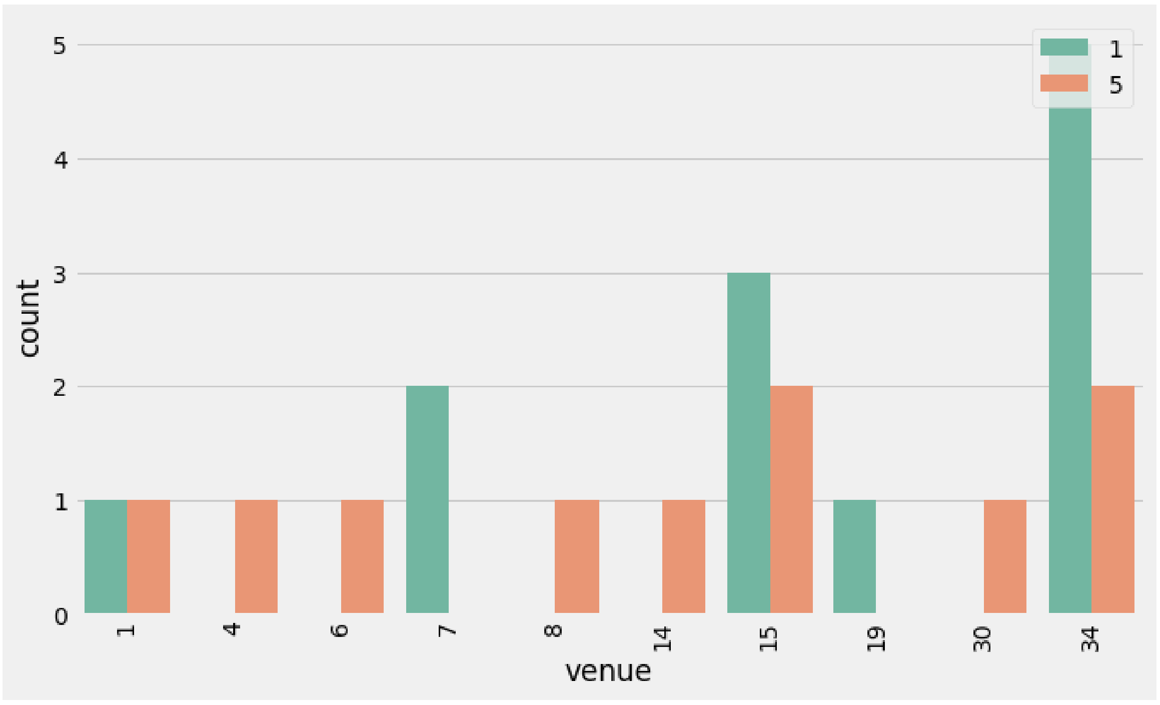
In [187]:

```
import matplotlib.pyplot as mlt
mlt.style.use('fivethirtyeight')
df_fil=df[df['toss_winner']==df['winner']]
slices=[len(df_fil),(577-len(df_fil))]
mlt.pie(slices,labels=['Toss & win','Toss & lose'],startangle=90,shadow=True,explode=(0
,0),autopct='%1.1f%%',colors=['r','g'])
fig = mlt.gcf()
fig.set_size_inches(6,6)
mlt.show()
```



In [188]:

```
import seaborn as sns
team1=dicVal['MI']
team2=dicVal['CSK']
mtemp=matches[((matches['team1']==team1)|(matches['team2']==team1))&((matches['team1']=
=team2)|(matches['team2']==team2))]
sns.countplot(x='venue', hue='winner',data=mtemp,palette='Set2')
mlt.xticks(rotation='vertical')
leg = mlt.legend( loc = 'upper right')
fig=mlt.gcf()
fig.set_size_inches(10,6)
mlt.show()
```



In [189]:

```
le.classes_[34]
```

Out[189]:

```
'Wankhede Stadium'
```

In [190]:

```
le.classes_[15]
```

Out[190]:

```
'MA Chidambaram Stadium, Chepauk'
```

In [ ]:

```

```