# ipl

June 4, 2021

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
df = pd.read_csv('data/ipl.csv')
```

[2]:
```
df.head()
```

[2]:

| | mid | date | venue | bat_team | bowl_team | batsman | bowler | runs | wickets | overs | runs_last_5 | wickets_last_5 | striker | non-striker | total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2008-04-18 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | SC Ganguly | P Kumar | 1 | 0 | 0.1 | 1 | 0 | 0 | 0 | 222 |
| 1 | 1 | 2008-04-18 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | BB McCullum | P Kumar | 1 | 0 | 0.2 | 1 | 0 | 0 | 0 | 222 |
| 2 | 1 | 2008-04-18 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | BB McCullum | P Kumar | 2 | 0 | 0.2 | 2 | 0 | 0 | 0 | 222 |
| 3 | 1 | 2008-04-18 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | BB McCullum | P Kumar | 2 | 0 | 0.3 | 2 | 0 | 0 | 0 | 222 |
| 4 | 1 | 2008-04-18 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | BB McCullum | P Kumar | 2 | 0 | 0.4 | 2 | 0 | 0 | 0 | 222 |

[3]:
```
df=df.drop(['date'],axis=1)
```

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

[4]:

| | mid | venue | bat_team | bowl_team | batsman | bowler | runs | wickets | overs | runs_last_5 | wickets_last_5 | striker | non-striker | total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | SC Ganguly | P Kumar | 1 | 0 | 0.1 | 1 | 0 | 0 | 0 | 222 |
| 1 | 1 | M Chinnaswamy Stadium | Kolkata Knight Riders | Royal Challengers Bangalore | BB McCullum | P Kumar | 1 | 0 | 0.2 | 1 | 0 | 0 | 0 | 222 |

```
2    1  M Chinnaswamy Stadium  Kolkata Knight Riders
3    1  M Chinnaswamy Stadium  Kolkata Knight Riders
4    1  M Chinnaswamy Stadium  Kolkata Knight Riders

                    bowl_team      batsman   bowler  runs  wickets  overs  \
0  Royal Challengers Bangalore   SC Ganguly  P Kumar     1        0    0.1
1  Royal Challengers Bangalore  BB McCullum  P Kumar     1        0    0.2
2  Royal Challengers Bangalore  BB McCullum  P Kumar     2        0    0.2
3  Royal Challengers Bangalore  BB McCullum  P Kumar     2        0    0.3
4  Royal Challengers Bangalore  BB McCullum  P Kumar     2        0    0.4

   runs_last_5  wickets_last_5  striker  non-striker  total
0            1               0        0            0    222
1            1               0        0            0    222
2            2               0        0            0    222
3            2               0        0            0    222
4            2               0        0            0    222
```

# 1 Data Visuaisation

[5]:
```
df.describe()
```

[5]:
```
                mid          runs       wickets         overs   runs_last_5  \
count  76014.000000  76014.000000  76014.000000  76014.000000  76014.000000
mean     308.627740     74.889349      2.415844      9.783068     33.216434
std      178.156878     48.823327      2.015207      5.772587     14.914174
min        1.000000      0.000000      0.000000      0.000000      0.000000
25%      154.000000     34.000000      1.000000      4.600000     24.000000
50%      308.000000     70.000000      2.000000      9.600000     34.000000
75%      463.000000    111.000000      4.000000     14.600000     43.000000
max      617.000000    263.000000     10.000000     19.600000    113.000000

       wickets_last_5       striker   non-striker         total
count    76014.000000  76014.000000  76014.000000  76014.000000
mean         1.120307     24.962283      8.869287    160.901452
std          1.053343     20.079752     10.795742     29.246231
min          0.000000      0.000000      0.000000     67.000000
25%          0.000000     10.000000      1.000000    142.000000
50%          1.000000     20.000000      5.000000    162.000000
75%          2.000000     35.000000     13.000000    181.000000
max          7.000000    175.000000    109.000000    263.000000
```

[6]:
```
df.dtypes
```

[6]:
```
mid                int64
venue             object
```

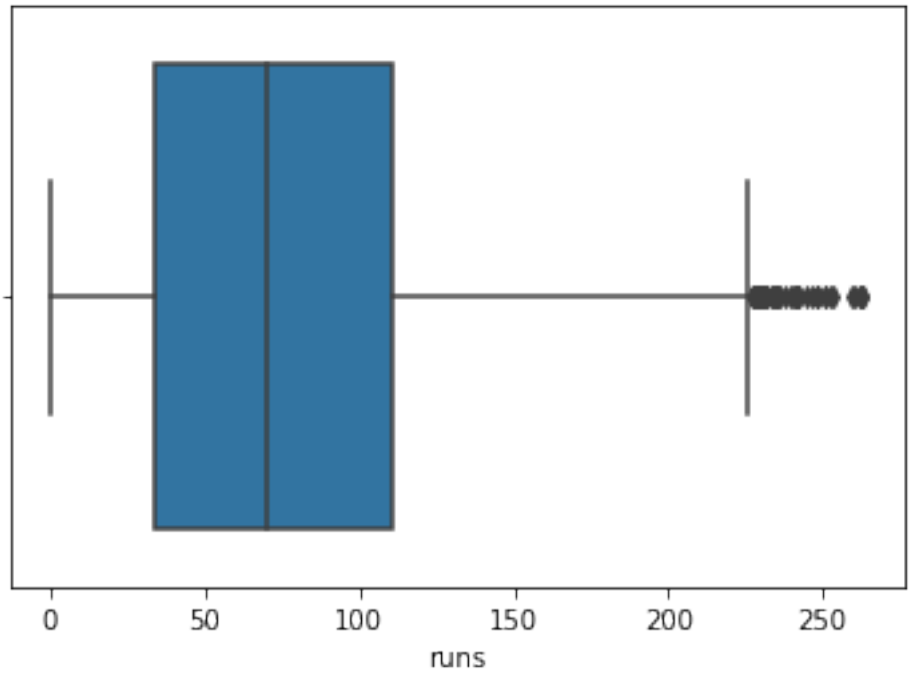
```
bat_team           object
bowl_team          object
batsman            object
bowler             object
runs                int64
wickets             int64
overs             float64
runs_last_5         int64
wickets_last_5      int64
striker             int64
non-striker         int64
total               int64
dtype: object
```

[7]:
```
df.shape
```

[7]:
```
(76014, 14)
```

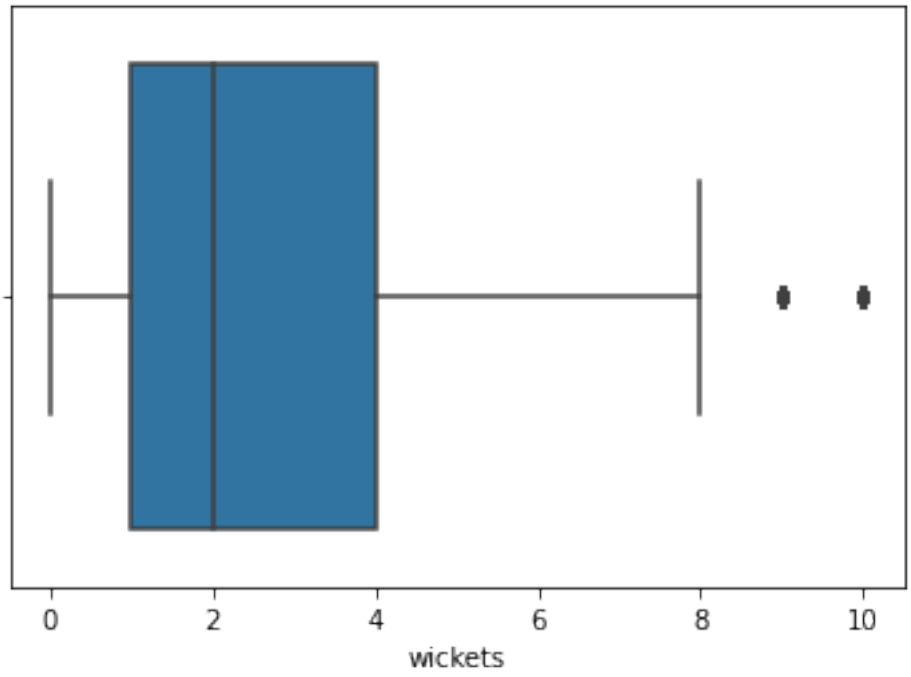
[8]:
```
sns.boxplot(x=df['runs'])
```

[8]:
```
<AxesSubplot:xlabel='runs'>
```



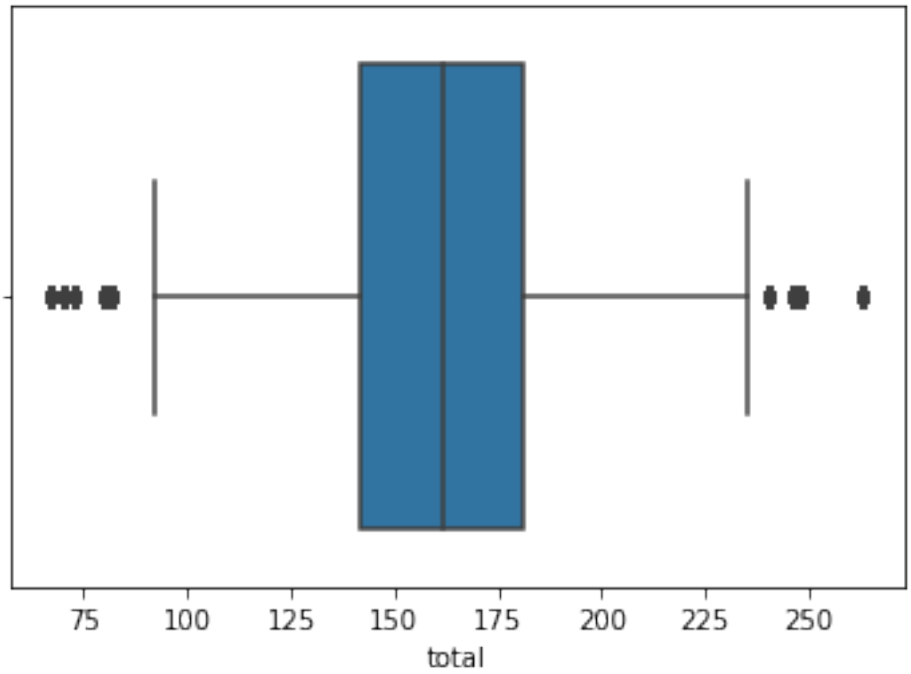

[9]:
```
sns.boxplot(x=df['wickets'])
```

[9]:
```
<AxesSubplot:xlabel='wickets'>
```




[10]:
```
sns.boxplot(x=df['total'])
```

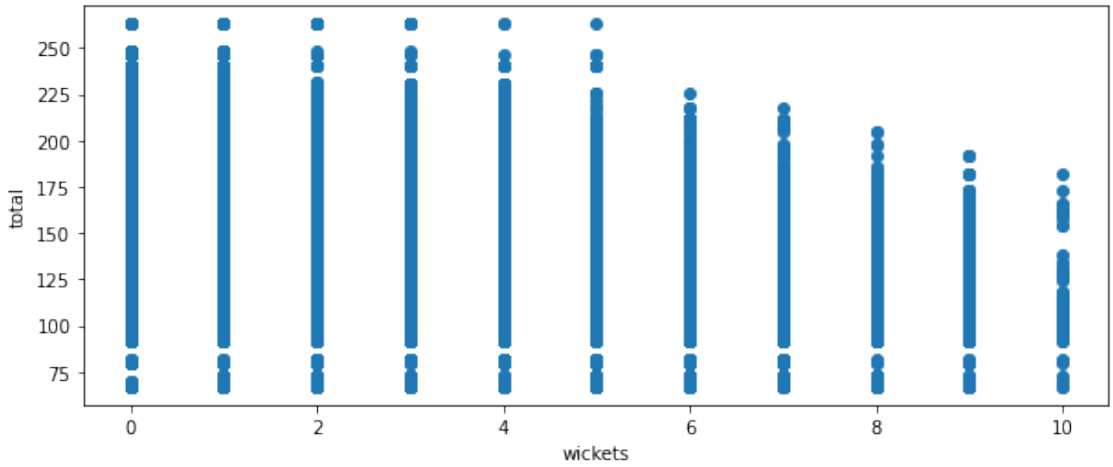
[10]: <AxesSubplot:xlabel='total'>

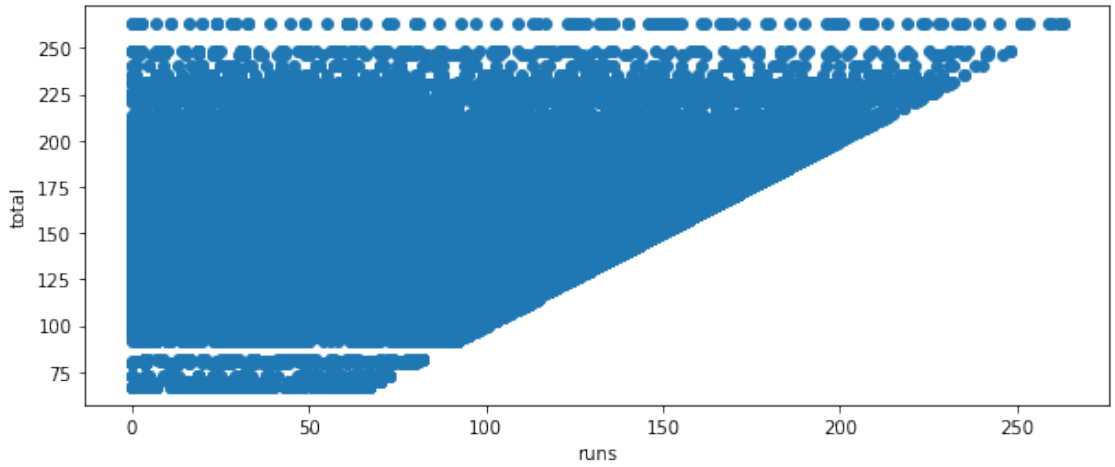


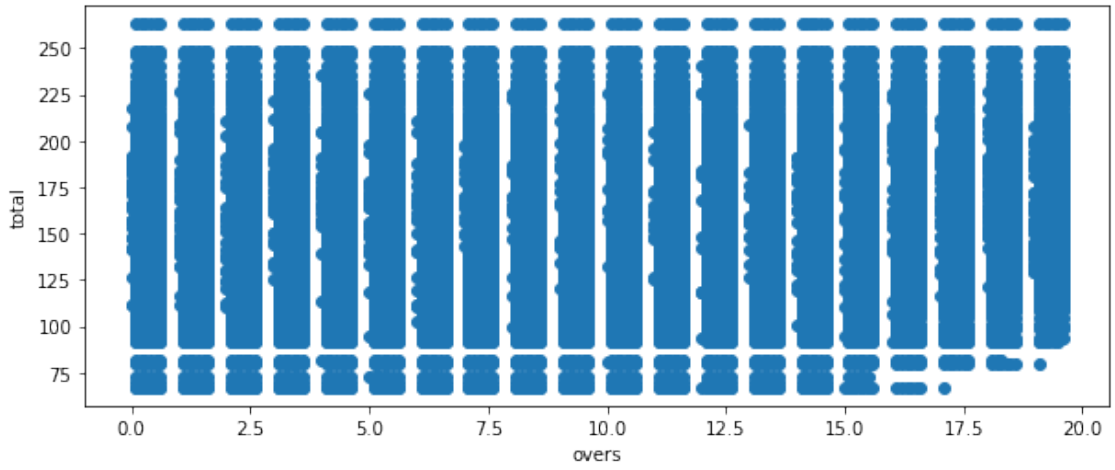
[11]:
```
fig, ax = plt.subplots(figsize=(10,4))
ax.scatter(df['wickets'] , df['total'])
ax.set_xlabel('wickets')
ax.set_ylabel('total')
plt.show()

fig, ax = plt.subplots(figsize=(10,4))
ax.scatter(df['runs'] , df['total'])
ax.set_xlabel('runs')
ax.set_ylabel('total')
plt.show()

fig, ax = plt.subplots(figsize=(10,4))
ax.scatter(df['overs'] , df['total'])
ax.set_xlabel('overs')
ax.set_ylabel('total')
plt.show()
```

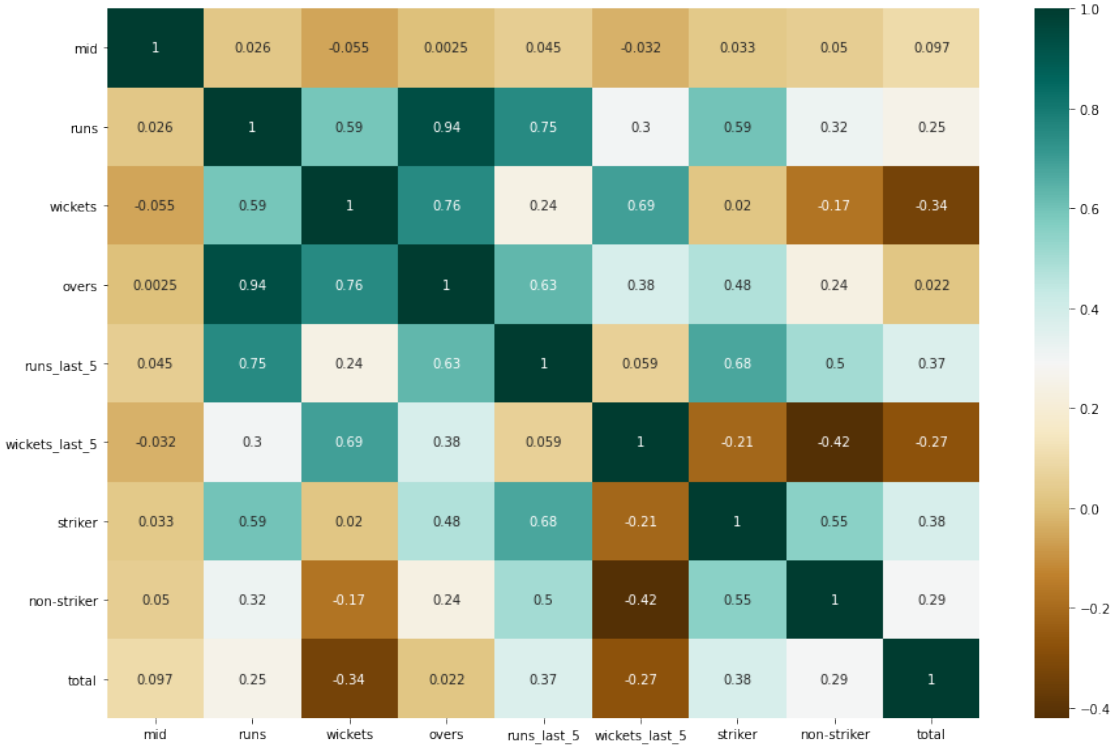




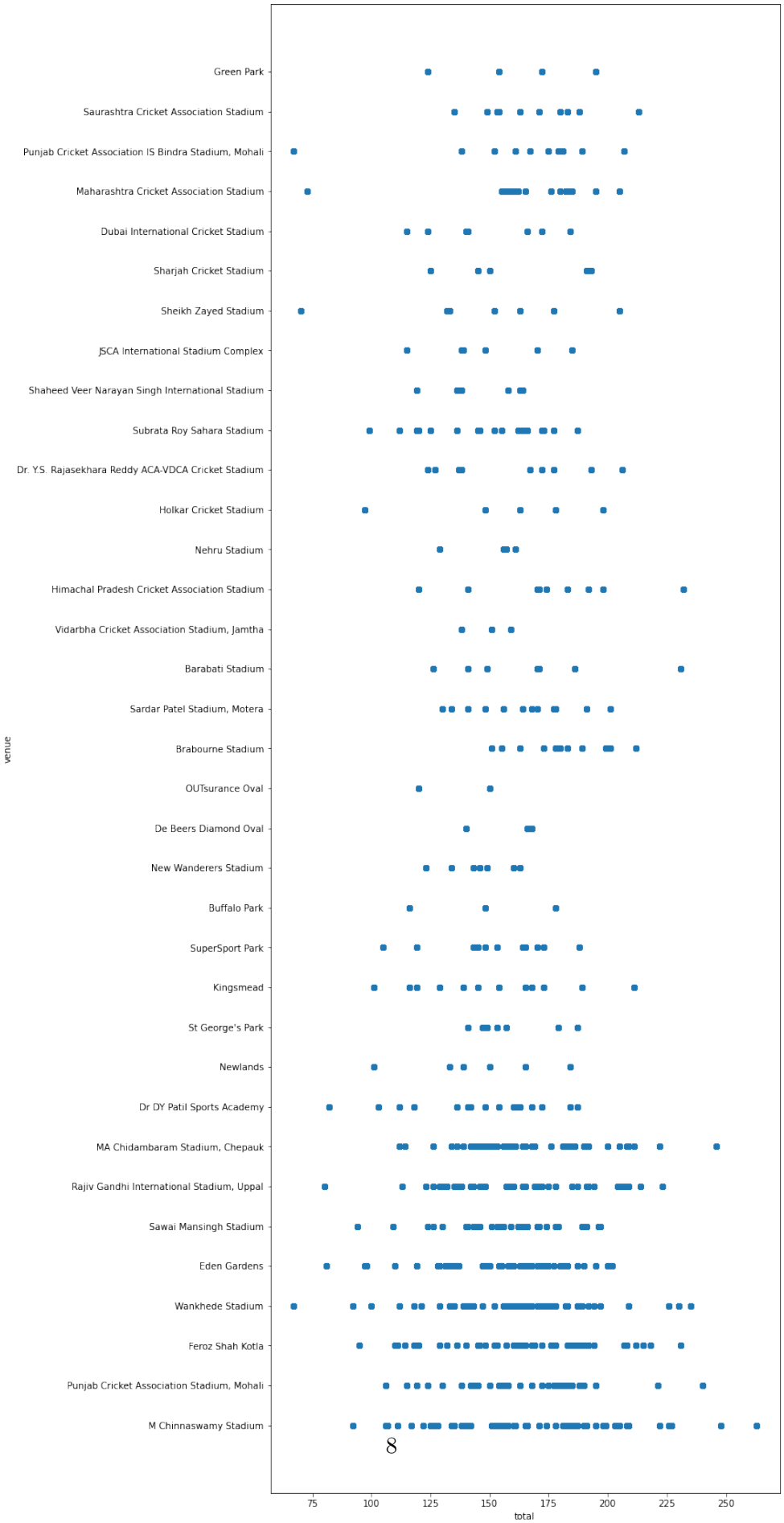




[12]:
```
plt.figure(figsize=(15,10))
c= df.corr()
sns.heatmap(c,cmap='BrBG',annot=True)
```

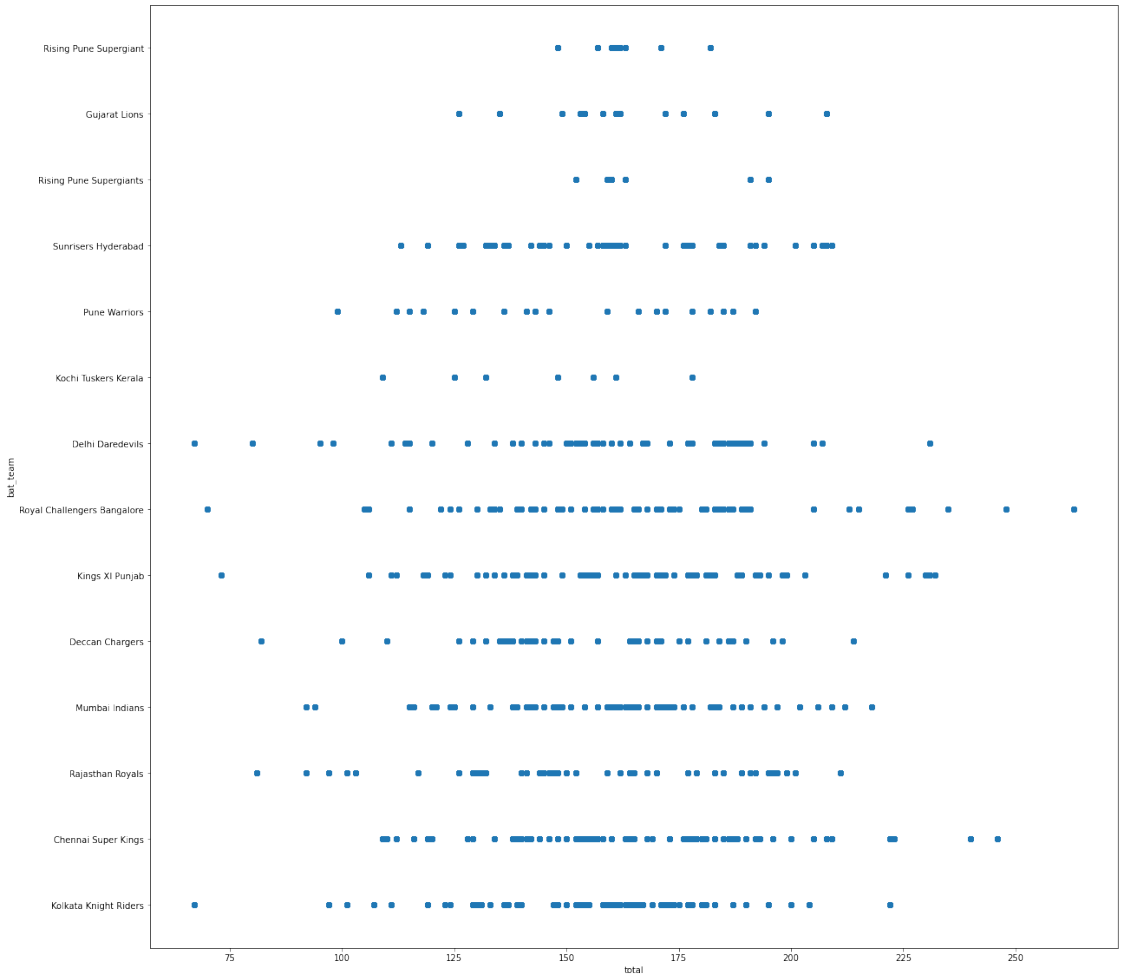
[12]: <AxesSubplot:>



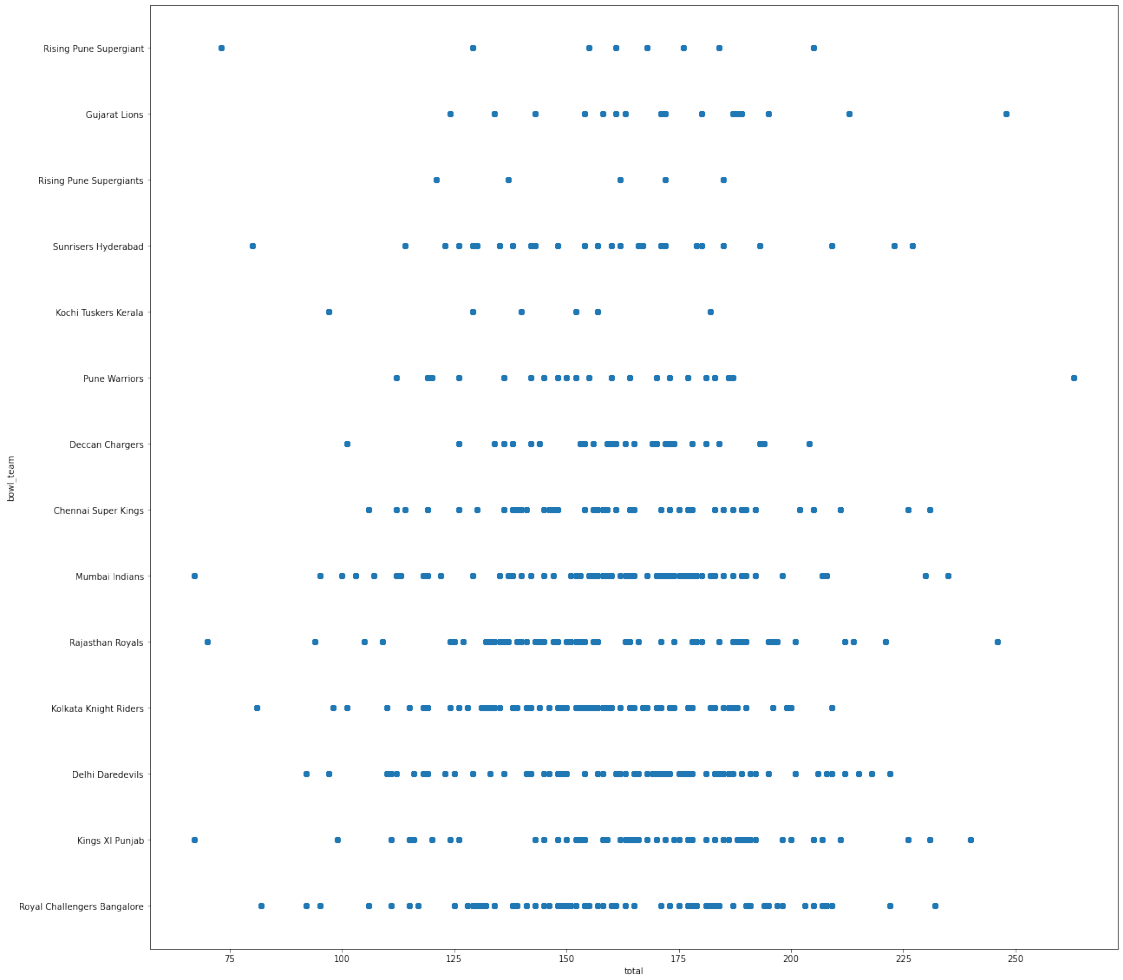

[13]:
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(10,30))
ax.scatter(df['total'] , df['venue'])
ax.set_xlabel('total')
ax.set_ylabel('venue')
plt.show()
```


Green Park
Saurashtra Cricket Association Stadium
Punjab Cricket Association IS Bindra Stadium, Mohali
Maharashtra Cricket Association Stadium
Dubai International Cricket Stadium
Sharjah Cricket Stadium
Sheikh Zayed Stadium
JSCA International Stadium Complex
Shaheed Veer Narayan Singh International Stadium
Subrata Roy Sahara Stadium
Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket Stadium
Holkar Cricket Stadium
Nehru Stadium
Himachal Pradesh Cricket Association Stadium
Vidarbha Cricket Association Stadium, Jamtha
Barabati Stadium
Sardar Patel Stadium, Motera
Brabourne Stadium
OUTsurance Oval
De Beers Diamond Oval
New Wanderers Stadium
Buffalo Park
SuperSport Park
Kingsmead
St George's Park
Newlands
Dr DY Patil Sports Academy
MA Chidambaram Stadium, Chepauk
Rajiv Gandhi International Stadium, Uppal
Sawai Mansingh Stadium
Eden Gardens
Wankhede Stadium
Feroz Shah Kotla
Punjab Cricket Association Stadium, Mohali
M Chinnaswamy Stadium
venue
75
100
125
150
175
200
225
250
total

[14]:
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(20,20))
ax.scatter(df['total'] , df['bat_team'])
ax.set_xlabel('total')
ax.set_ylabel('bat_team')
plt.show()
```

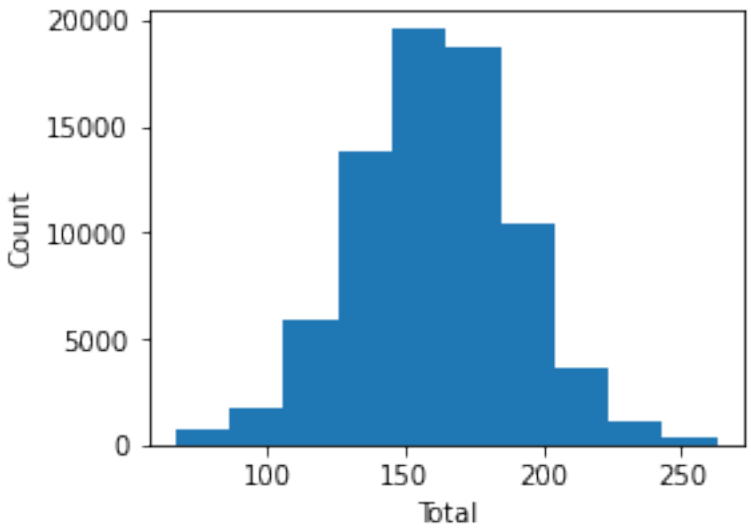



[15]:
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(20,20))
ax.scatter(df['total'] , df['bowl_team'])
ax.set_xlabel('total')
ax.set_ylabel('bowl_team')
plt.show()
```




[16]:
```
plt.figure(figsize=(4,3))
plt.hist(df.total)
plt.xlabel('Total')
plt.ylabel('Count')
plt.tight_layout
```

[16]:
```
<function matplotlib.pyplot.tight_layout(*, pad=1.08, h_pad=None, w_pad=None,
rect=None)>
```




## 2 We will now convert the textual data to numeric data so that those columns can be used for prediciton

[17]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['bat_team']=le.fit_transform(df['bat_team'])
df['venue']=le.fit_transform(df['venue'])
df['bowl_team']=le.fit_transform(df['bowl_team'])
df['batsman']=le.fit_transform(df['batsman'])
df['bowler']=le.fit_transform(df['bowler'])
```

[18]:
```
df.head(15)
```

[18]:

| | mid | venue | bat_team | bowl_team | batsman | bowler | runs | wickets | overs | \ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 14 | 6 | 12 | 328 | 201 | 1 | 0 | 0.1 | |
| 1 | 1 | 14 | 6 | 12 | 61 | 201 | 1 | 0 | 0.2 | |
| 2 | 1 | 14 | 6 | 12 | 61 | 201 | 2 | 0 | 0.2 | |
| 3 | 1 | 14 | 6 | 12 | 61 | 201 | 2 | 0 | 0.3 | |
| 4 | 1 | 14 | 6 | 12 | 61 | 201 | 2 | 0 | 0.4 | |
| 5 | 1 | 14 | 6 | 12 | 61 | 201 | 2 | 0 | 0.5 | |
| 6 | 1 | 14 | 6 | 12 | 61 | 201 | 3 | 0 | 0.6 | |
| 7 | 1 | 14 | 6 | 12 | 61 | 328 | 3 | 0 | 1.1 | |
| 8 | 1 | 14 | 6 | 12 | 61 | 328 | 7 | 0 | 1.2 | |
| 9 | 1 | 14 | 6 | 12 | 61 | 328 | 11 | 0 | 1.3 | |
| 10 | 1 | 14 | 6 | 12 | 61 | 328 | 17 | 0 | 1.4 | |
| 11 | 1 | 14 | 6 | 12 | 61 | 328 | 21 | 0 | 1.5 | |
| 12 | 1 | 14 | 6 | 12 | 61 | 328 | 21 | 0 | 1.6 | |

```
13    1     14         6         12      328     201    21        0    2.1
14    1     14         6         12      328     201    21        0    2.2
```

| | runs_last_5 | wickets_last_5 | striker | non-striker | total |
|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 222 |
| 1 | 1 | 0 | 0 | 0 | 222 |
| 2 | 2 | 0 | 0 | 0 | 222 |
| 3 | 2 | 0 | 0 | 0 | 222 |
| 4 | 2 | 0 | 0 | 0 | 222 |
| 5 | 2 | 0 | 0 | 0 | 222 |
| 6 | 3 | 0 | 0 | 0 | 222 |
| 7 | 3 | 0 | 0 | 0 | 222 |
| 8 | 7 | 0 | 4 | 0 | 222 |
| 9 | 11 | 0 | 8 | 0 | 222 |
| 10 | 17 | 0 | 14 | 0 | 222 |
| 11 | 21 | 0 | 18 | 0 | 222 |
| 12 | 21 | 0 | 18 | 0 | 222 |
| 13 | 21 | 0 | 18 | 0 | 222 |
| 14 | 21 | 0 | 18 | 0 | 222 |

## 3 Dividing Dependent and Independent Variables

[19]:
```
x = df.iloc[:,[1,2,3,4,5,6,7,8,11,12]].values
y = df.iloc[:, 13].values
```

## 4 Train Test Split

[20]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3,␣
 ↪random_state = 42)
print(x_test[0], y_test[0])
```

```
[ 18.    2.   12.  122.  128.    1.    2.    1.2   1.    0. ] 134
```

## 5 Scaling the Dependent and Independent Variables

[21]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

## 6 Linar Regression

[22]:
```
from sklearn.linear_model import LinearRegression
lin = LinearRegression()
lin.fit(x_train,y_train)
```

[22]:
```
LinearRegression()
```

## 7 Testing the Accuracy

[23]:
```
# Testing the dataset on trained model
from sklearn.metrics import r2_score,accuracy_score
y_pred = lin.predict(x_test)
score = lin.score(x_test,y_test)
print("R square value:" , score)
```

```
R square value: 0.5177231535552973
```

[24]:
```
def custom_accuracy(y_test,y_pred,thresold):
    right = 0

    l = len(y_pred)
    for i in range(0,l):
        if(abs(y_pred[i]-y_test[i]) <= thresold):
            right += 1
    return ((right/l)*100)
print("Custom accuracy:" , custom_accuracy(y_test,y_pred,20))
```

```
Custom accuracy: 74.17233062924797
```

## 8 Test Case Using Present Data

[25]:
```
for i in range(4):
    pred = lin.predict(sc.transform(np.array([x_test[i]])))
    print("Actual Score: ", y_test[i])
    print("Predicted Score: ", pred)
    print("Margin of Error: ", abs(y_test[i] - pred[0]))
    print("Margin of Error percent", ((abs(y_test[i] - pred[0]))*100)/y_test[i])
    print("")
```

```
Actual Score:  134
Predicted Score:  [169.20976448]
Margin of Error:  35.2097644796859
Margin of Error percent 26.27594364155664

Actual Score:  195
```

```
Predicted Score:  [156.71856818]
Margin of Error:  38.28143182138632
Margin of Error percent 19.63150349814683

Actual Score:  183
Predicted Score:  [162.78339884]
Margin of Error:  20.21660116181866
Margin of Error percent 11.047323039245168

Actual Score:  183
Predicted Score:  [157.19424085]
Margin of Error:  25.805759148437602
Margin of Error percent 14.101507731386667
```

## 9 Test Case using Random Data

[26]:
```
lin1 = lin.predict(sc.transform(np.array([[100,12,2,391,283,50,4,5.2,11,12]])))
print("Prediction score:" , lin1)
```

```
Prediction score: [155.76378686]
```

[27]:
```
lin2 = lin.predict(sc.transform(np.array([[5,10,7,112,127,100,1,8.5,80,10]])))
print("Prediction score:" , lin2)
```

```
Prediction score: [200.92768]
```

[28]:
```
lin3 = lin.predict(sc.transform(np.array([[56,13,4,14,24,75,5,4.3,34,12]])))
print("Prediction score:" , lin3)
```

```
Prediction score: [178.29172526]
```

## 10 Decision Tree Regressor

[29]:
```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier()
clf = clf.fit(x_train,y_train)
y_pred1 = clf.predict(x_test)
```

## 11 Testing the accuracy

[30]:
```
y_pred1 = clf.predict(x_test)
score1 = clf.score(x_test,y_test)
print("R square value:" , score1)
```

```
R square value: 0.9748739311554484
```

[31]:
```
def custom_accuracy1(y_test,y_pred1,thresold):
    right = 0
    l = len(y_pred1)
    for i in range(0,l):
        if(abs(y_pred1[i]-y_test[i]) <= thresold):
            right += 1
    return ((right/l)*100)

print("Custom accuracy:" , custom_accuracy1(y_test,y_pred1,20),'%')
```

```
Custom accuracy: 98.53540890155668 %
```

## 12 Test Case using Present Data

[32]:
```
for i in range(4):
    predclf = clf.predict(sc.transform(np.array([x_test[i]])))
    print("Actual Score: ", y_test[i])
    print("Predicted Score: ", predclf)
    print("Margin of Error: ", abs(y_test[i] - predclf[0]))
    print("Margin of Error percent", ((abs(y_test[i] - predclf[0]))*100)/
 ↪y_test[i])
    print("")
```

```
Actual Score:  134
Predicted Score:  [178]
Margin of Error:  44
Margin of Error percent 32.83582089552239

Actual Score:  195
Predicted Score:  [185]
Margin of Error:  10
Margin of Error percent 5.128205128205129

Actual Score:  183
Predicted Score:  [185]
Margin of Error:  2
Margin of Error percent 1.092896174863388

Actual Score:  183
Predicted Score:  [185]
Margin of Error:  2
Margin of Error percent 1.092896174863388
```

## 13 Test Case using Random Data

[33]:
```
clf1 = clf.predict(sc.transform(np.array([[100,12,2,391,283,50,4,5.2,11,12]])))
print("Prediction score:" , clf1)
```

```
Prediction score: [149]
```

[34]:
```
clf2 = clf.predict(sc.transform(np.array([[5,10,7,112,127,100,1,8.5,80,10]])))
print("Prediction score:" , clf2)
```

```
Prediction score: [165]
```

[35]:
```
clf3 = clf.predict(sc.transform(np.array([[56,13,4,14,24,75,5,4.3,34,12]])))
print("Prediction score:" , clf3)
```

```
Prediction score: [159]
```

## 14 Random Forest Regressor

[36]:
```
from sklearn.ensemble import RandomForestRegressor
reg = RandomForestRegressor(n_estimators=100,max_features=None)
reg.fit(x_train,y_train)
```

[36]:
```
RandomForestRegressor(max_features=None)
```

## 15 Testing the accuracy

[37]:
```
# Testing the dataset on trained model
y_pred2 = reg.predict(x_test)
score2 = reg.score(x_test,y_test)
print("R square value:" , score2)
```

```
R square value: 0.9305247548012424
```

[38]:
```
def custom_accuracy1(y_test,y_pred2,thresold):
    right = 0
    l = len(y_pred2)
    for i in range(0,l):
        if(abs(y_pred2[i]-y_test[i]) <= thresold):
            right += 1
    return ((right/l)*100)

print("Custom accuracy:" , custom_accuracy1(y_test,y_pred2,20),'%')
```

```
Custom accuracy: 97.15413286559965 %
```

## 16 Test Case using Present Data

[39]:
```
for i in range(4):
    predreg = reg.predict(sc.transform(np.array([x_test[i]])))
    print("Actual Score: ", y_test[i])
    print("Predicted Score: ", predreg)
    print("Margin of Error: ", abs(y_test[i] - predreg[0]))
    print("Margin of Error percent", ((abs(y_test[i] - predreg[0]))*100)/
 ↪y_test[i])
    print("")
```

```
Actual Score:  134
Predicted Score:  [167.32]
Margin of Error:  33.31999999999999
Margin of Error percent 24.86567164179104

Actual Score:  195
Predicted Score:  [168.7]
Margin of Error:  26.30000000000001
Margin of Error percent 13.487179487179493

Actual Score:  183
Predicted Score:  [163.53]
Margin of Error:  19.47
Margin of Error percent 10.639344262295081

Actual Score:  183
Predicted Score:  [169.37]
Margin of Error:  13.629999999999995
Margin of Error percent 7.4480874316939865
```

## 17 Test Case

[40]:
```
reg1 = reg.predict(sc.transform(np.array([[100,12,2,391,283,50,4,5.2,11,12]])))
print("Prediction score:" , reg1)
```

```
Prediction score: [116.6]
```

[41]:
```
reg2 = reg.predict(sc.transform(np.array([[5,10,7,112,127,100,1,8.5,80,10]])))
print("Prediction score:" , reg2)
```

```
Prediction score: [200.02]
```

[42]:
```
reg3 = reg.predict(sc.transform(np.array([[56,13,4,14,24,75,5,4.3,34,12]])))
print("Prediction score:" , reg3)
```

```
Prediction score: [131.08]
```