odi

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/odi.csv')
```

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

[2]:
```
   mid        date                                 venue bat_team bowl_team  \
0    1  2006-06-13  Civil Service Cricket Club, Stormont  England   Ireland
1    1  2006-06-13  Civil Service Cricket Club, Stormont  England   Ireland
2    1  2006-06-13  Civil Service Cricket Club, Stormont  England   Ireland
3    1  2006-06-13  Civil Service Cricket Club, Stormont  England   Ireland
4    1  2006-06-13  Civil Service Cricket Club, Stormont  England   Ireland

          batsman       bowler  runs  wickets  overs  runs_last_5  \
0  ME Trescothick  DT Johnston     0        0    0.1            0
1  ME Trescothick  DT Johnston     0        0    0.2            0
2  ME Trescothick  DT Johnston     4        0    0.3            4
3  ME Trescothick  DT Johnston     6        0    0.4            6
4  ME Trescothick  DT Johnston     6        0    0.5            6

   wickets_last_5  striker  non-striker  total
0               0        0            0    301
1               0        0            0    301
2               0        0            0    301
3               0        0            0    301
4               0        0            0    301
```

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

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

[4]:
```
   mid                                 venue bat_team bowl_team  \
0    1  Civil Service Cricket Club, Stormont  England   Ireland
1    1  Civil Service Cricket Club, Stormont  England   Ireland
```

```
2    1  Civil Service Cricket Club, Stormont  England   Ireland
3    1  Civil Service Cricket Club, Stormont  England   Ireland
4    1  Civil Service Cricket Club, Stormont  England   Ireland

          batsman       bowler  runs  wickets  overs  runs_last_5  \
0  ME Trescothick  DT Johnston     0        0    0.1            0
1  ME Trescothick  DT Johnston     0        0    0.2            0
2  ME Trescothick  DT Johnston     4        0    0.3            4
3  ME Trescothick  DT Johnston     6        0    0.4            6
4  ME Trescothick  DT Johnston     6        0    0.5            6

   wickets_last_5  striker  non-striker  total
0               0        0            0    301
1               0        0            0    301
2               0        0            0    301
3               0        0            0    301
4               0        0            0    301
```

1 Data Visuaisation

[5]:

```
df.describe()
```

[5]:

```
                 mid           runs        wickets          overs  \
count  350899.000000  350899.000000  350899.000000  350899.000000
mean      594.360426     114.801661       2.974970      24.052899
std       343.605128      77.665959       2.298959      14.235439
min         1.000000       0.000000       0.000000       0.000000
25%       296.000000      51.000000       1.000000      11.600000
50%       596.000000     105.000000       3.000000      23.600000
75%       893.000000     168.000000       4.000000      36.200000
max      1188.000000     444.000000      10.000000      49.600000

         runs_last_5  wickets_last_5        striker    non-striker  \
count  350899.000000   350899.000000  350899.000000  350899.000000
mean       23.548303        0.669814      35.180129      12.427944
std        11.042974        0.833895      28.115264      15.019181
min         0.000000        0.000000       0.000000       0.000000
25%        17.000000        0.000000      13.000000       2.000000
50%        23.000000        0.000000      29.000000       7.000000
75%        29.000000        1.000000      50.000000      18.000000
max       101.000000        7.000000     264.000000     149.000000

               total
count  350899.000000
mean      255.355387
std        62.354412
```

```
min        44.000000
25%       217.000000
50%       257.000000
75%       298.000000
max       444.000000
```

[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]:

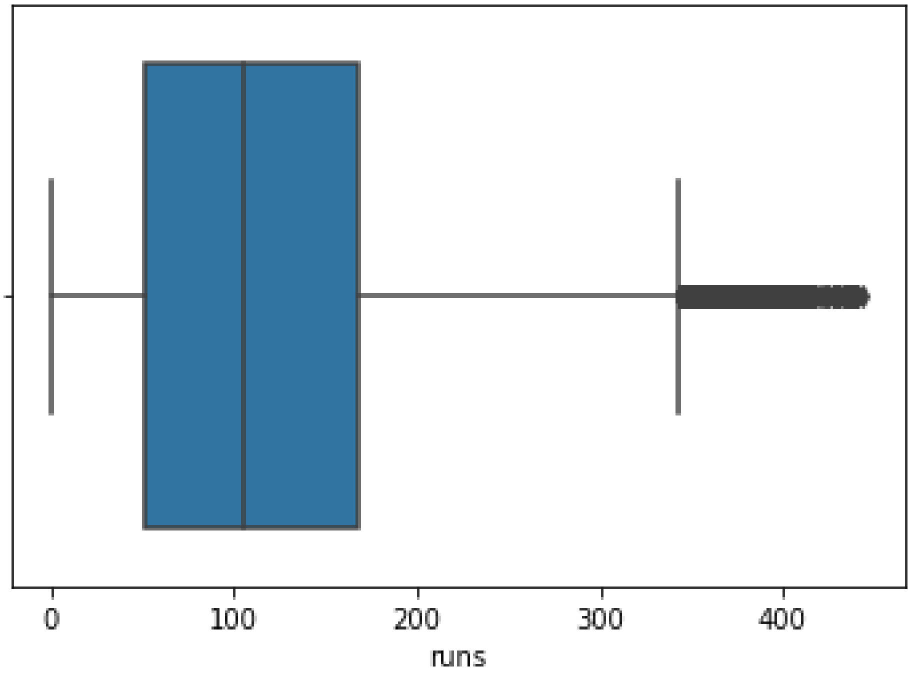
```
(350899, 14)
```

[8]:

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

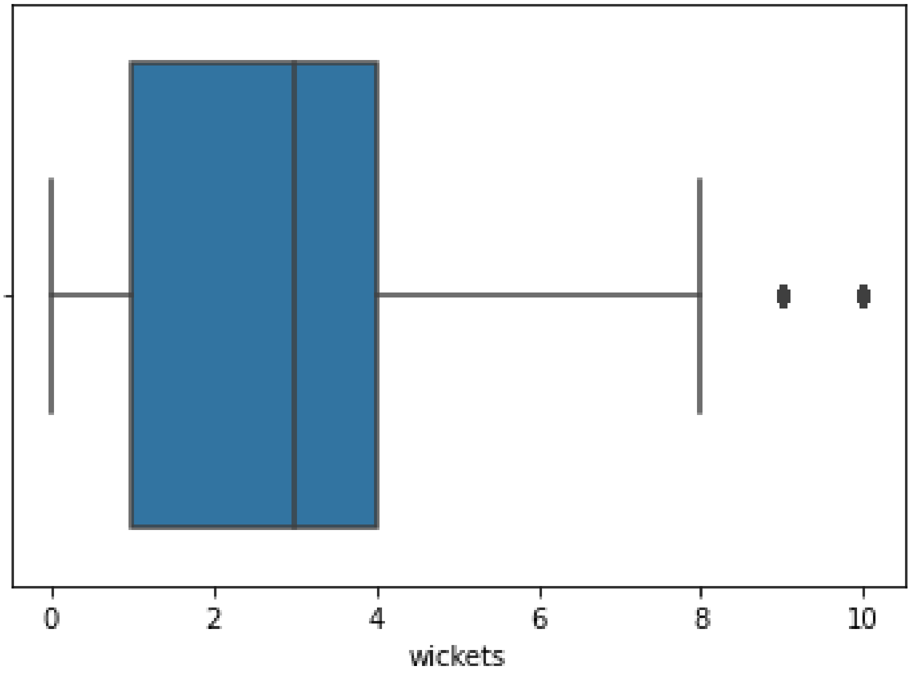
[8]:

```
<AxesSubplot:xlabel='runs'>
```



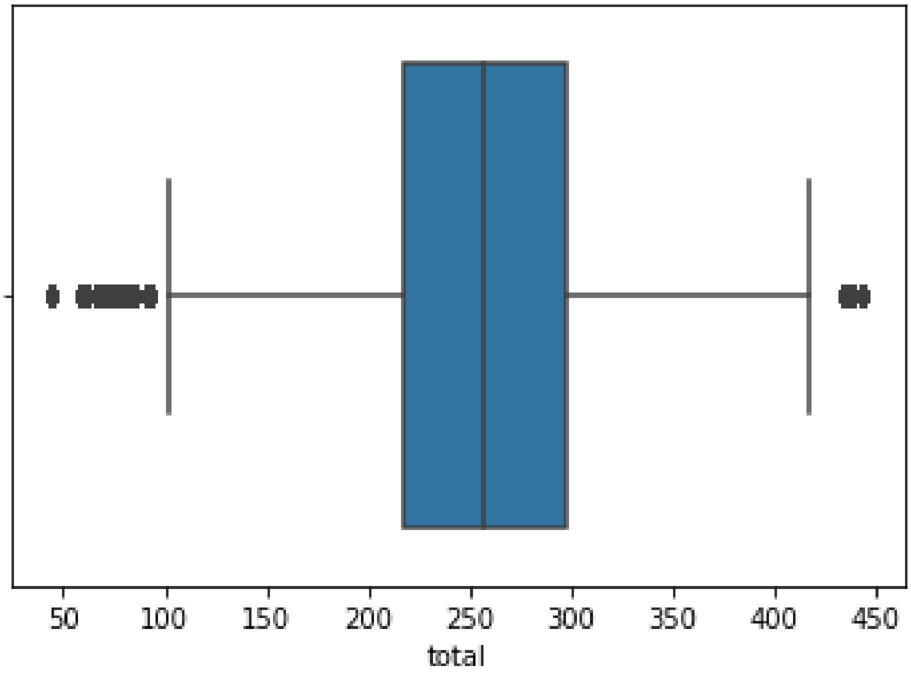
```
[9]: sns.boxplot(x=df['wickets'])
```

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



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

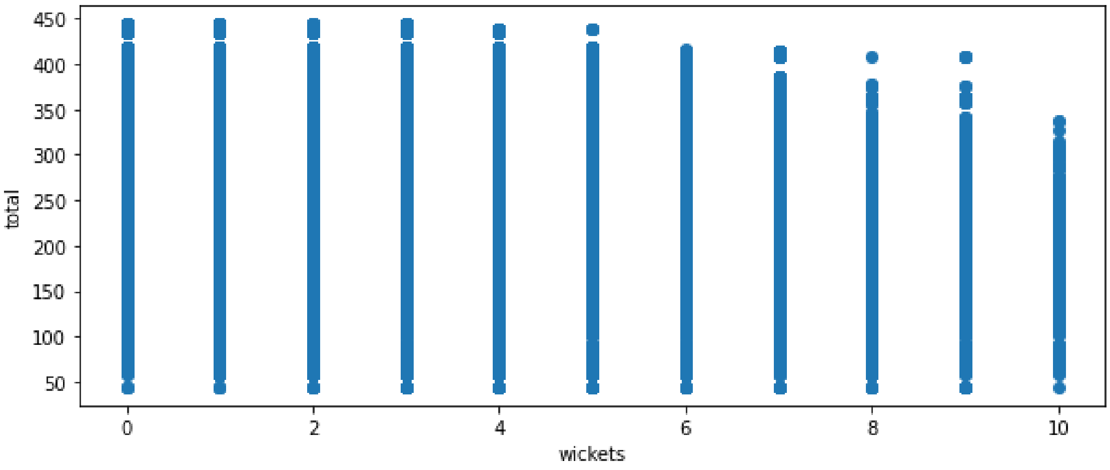
```
[10]: <AxesSubplot:xlabel='total'>
```

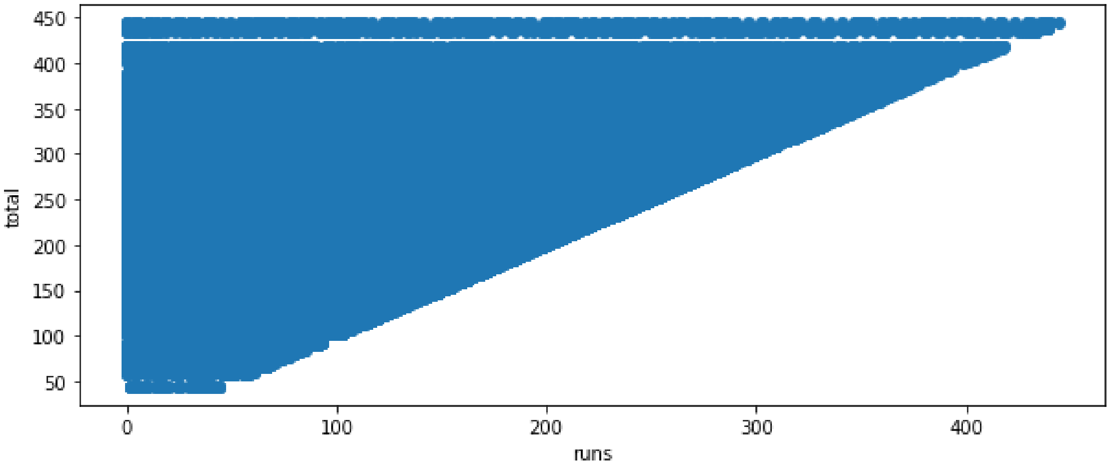


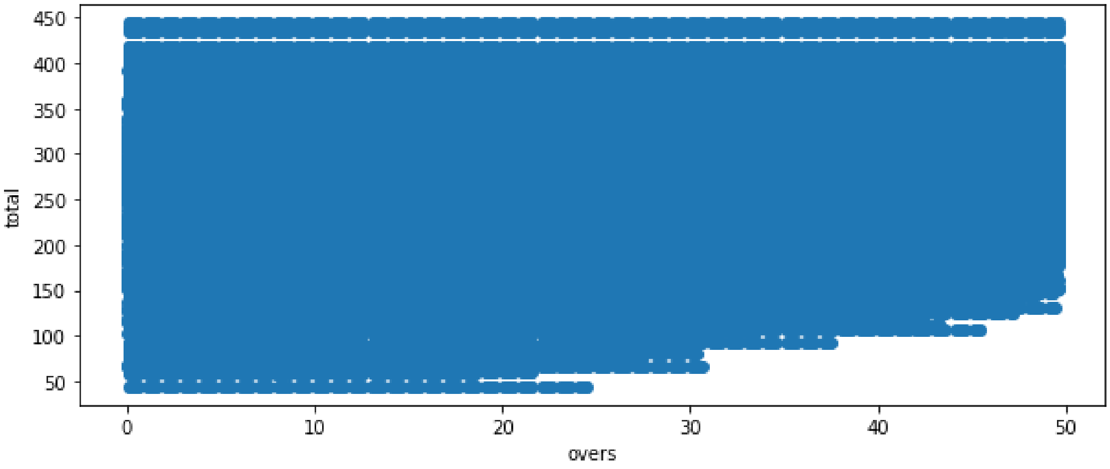
```
[11]: fig, ax = plt.subplots(figsize=(10,4))
      ax.scatter(df['wickets'] , df['total'])
      ax.set_xlabel('wickets')
      ax.set_ylabel('total')
      plt.show()

      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()
```


450
400
350
300
250
200
150
100
50
total
0
2
4
6
8
10
wickets


450
400
350
300
250
200
150
100
50
total
0
100
200
300
400
runs

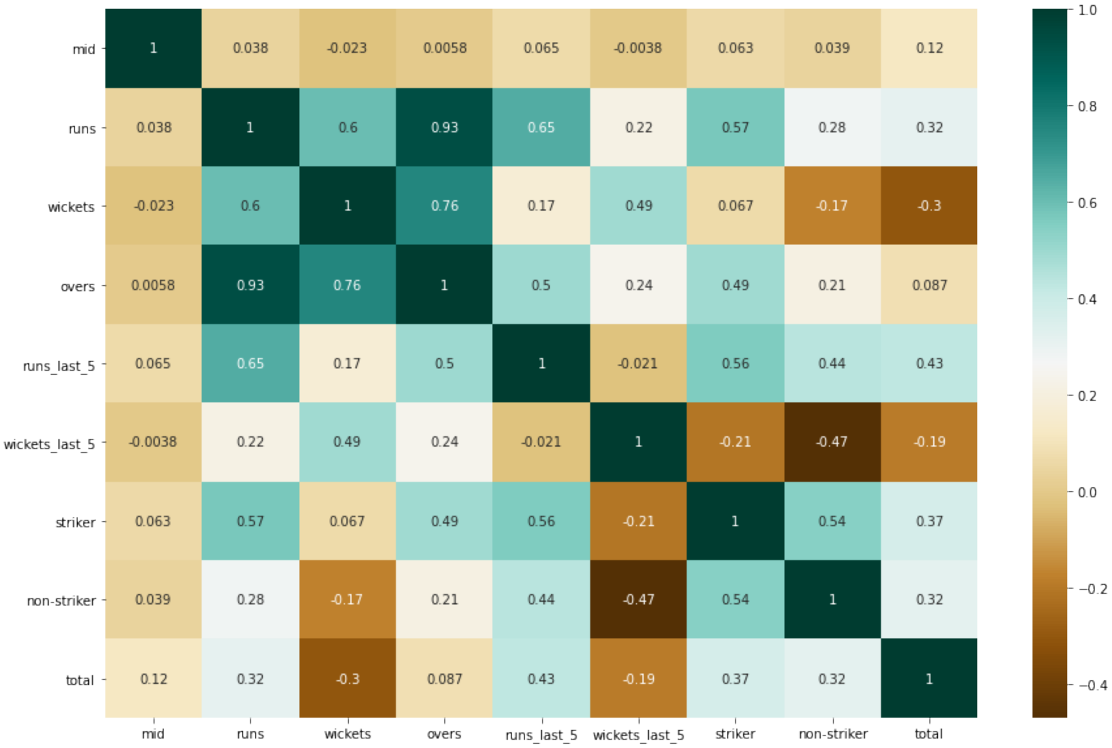

450
400
350
300
250
200
150
100
50
total
0
10
20
30
40
50
overs

[12]:

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

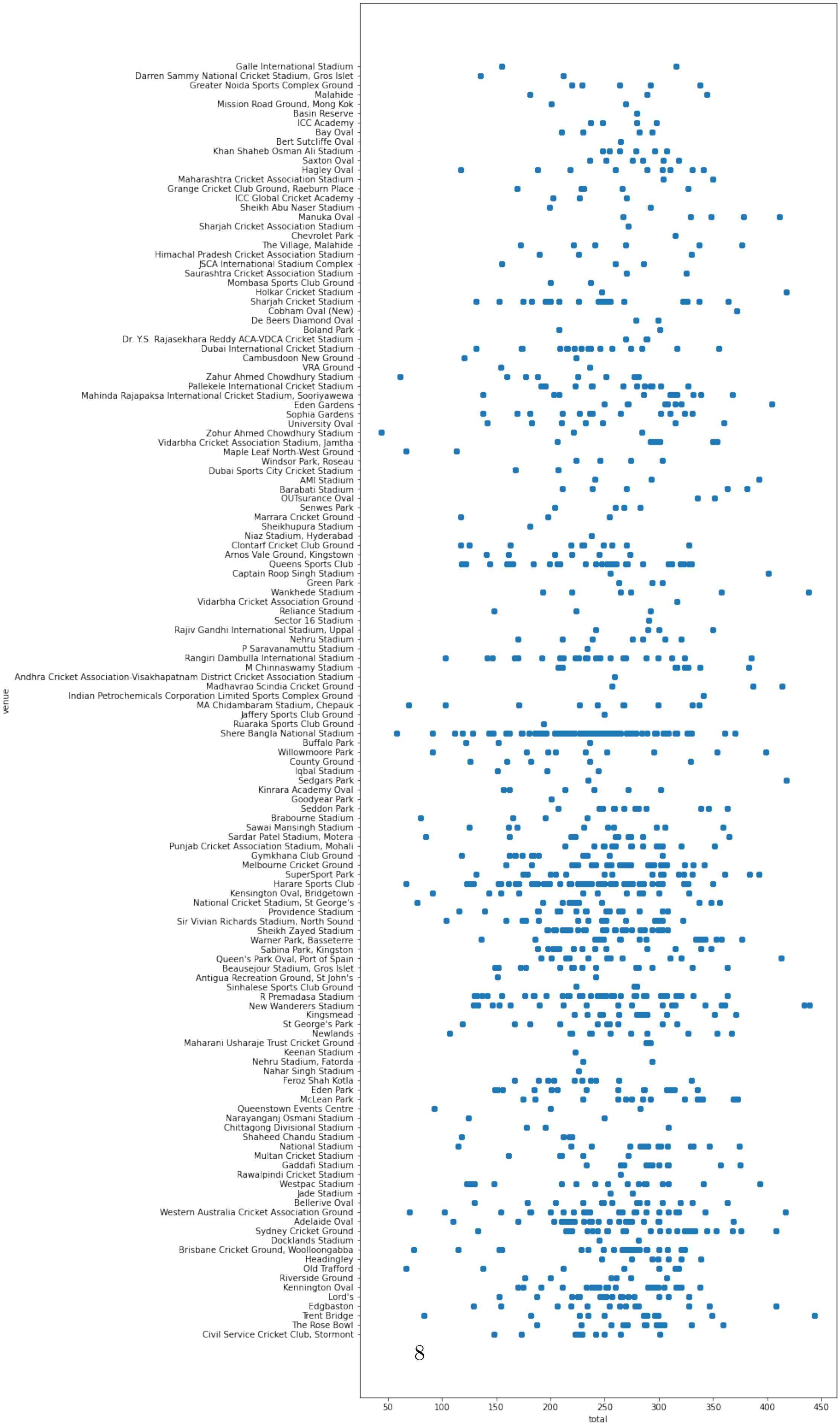
[12]:

```
<AxesSubplot:>
```



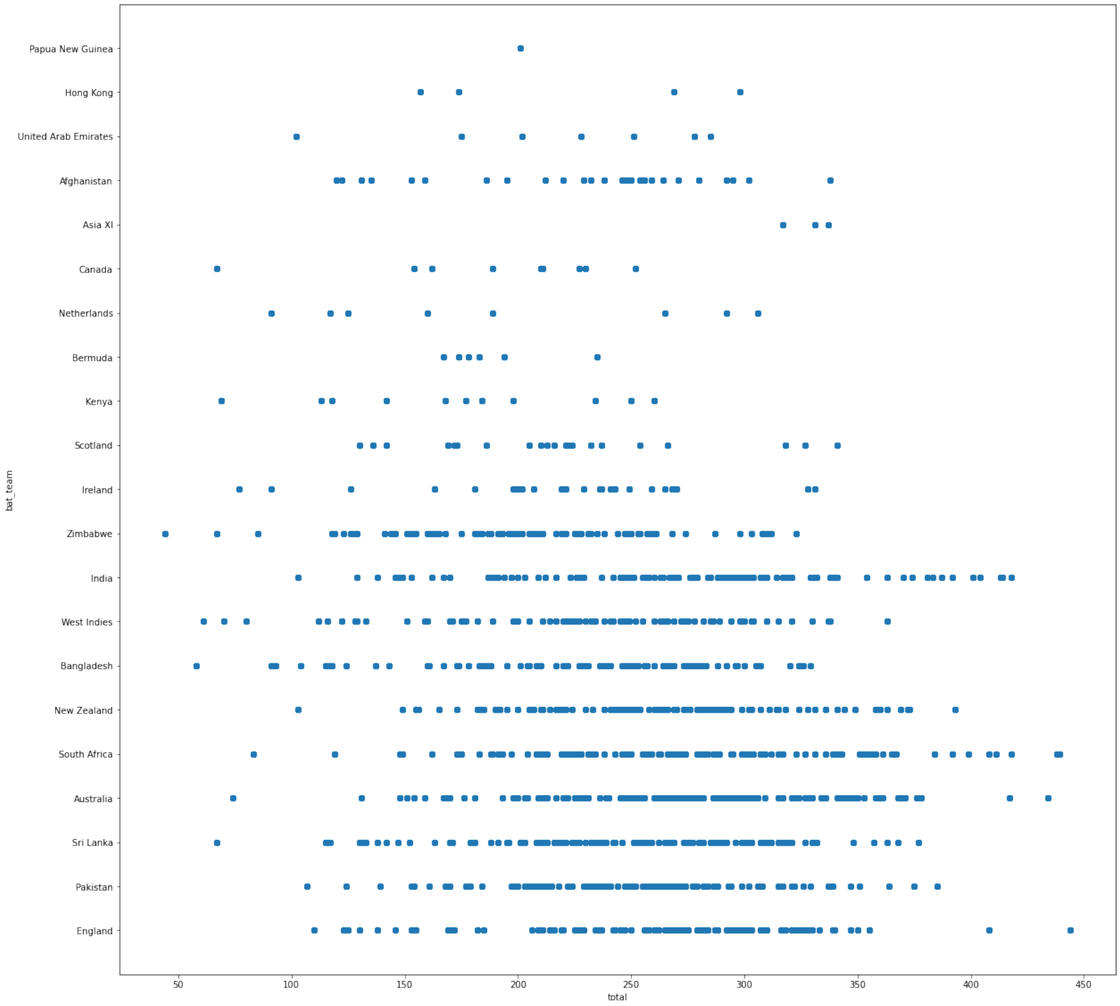
[13]:

```
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()
```


Galle International Stadium
Darren Sammy National Cricket Stadium, Gros Islet
Greater Noida Sports Complex Ground
Malahide
Mission Road Ground, Mong Kok
Basin Reserve
ICC Academy
Bay Oval
Bert Sutcliffe Oval
Khan Shaheb Osman Ali Stadium
Saxton Oval
Hagley Oval
Maharashtra Cricket Association Stadium
Grange Cricket Club Ground, Raeburn Place
ICC Global Cricket Academy
Sheikh Abu Naser Stadium
Manuka Oval
Sharjah Cricket Association Stadium
Chevrolet Park
The Village, Malahide
Himachal Pradesh Cricket Association Stadium
JSCA International Stadium Complex
Saurashtra Cricket Association Stadium
Mombasa Sports Club Ground
Holkar Cricket Stadium
Sharjah Cricket Stadium
Cobham Oval (New)
De Beers Diamond Oval
Boland Park
Dr. Y.S. Rajasekhara Reddy ACA-VDCA Cricket Stadium
Dubai International Cricket Stadium
Cambusdoon New Ground
VRA Ground
Zahur Ahmed Chowdhury Stadium
Pallekele International Cricket Stadium
Mahinda Rajapaksa International Cricket Stadium, Sooriyawewa
Eden Gardens
Sophia Gardens
University Oval
Zohur Ahmed Chowdhury Stadium
Vidarbha Cricket Association Stadium, Jamtha
Maple Leaf North-West Ground
Windsor Park, Roseau
Dubai Sports City Cricket Stadium
AMI Stadium
Barabati Stadium
OUTsurance Oval
Senwes Park
Marrara Cricket Ground
Sheikhupura Stadium
Niaz Stadium, Hyderabad
Clontarf Cricket Club Ground
Arnos Vale Ground, Kingstown
Queens Sports Club
Captain Roop Singh Stadium
Green Park
Wankhede Stadium
Vidarbha Cricket Association Ground
Reliance Stadium
Sector 16 Stadium
Rajiv Gandhi International Stadium, Uppal
Nehru Stadium
P Saravanamuttu Stadium
Rangiri Dambulla International Stadium
M Chinnaswamy Stadium
Andhra Cricket Association-Visakhapatnam District Cricket Association Stadium
Madhavrao Scindia Cricket Ground
Indian Petrochemicals Corporation Limited Sports Complex Ground
MA Chidambaram Stadium, Chepauk
Jaffery Sports Club Ground
Ruaraka Sports Club Ground
Shere Bangla National Stadium
Buffalo Park
Willowmoore Park
County Ground
Iqbal Stadium
Sedgars Park
Kinrara Academy Oval
Goodyear Park
Seddon Park
Brabourne Stadium
Sawai Mansingh Stadium
Sardar Patel Stadium, Motera
Punjab Cricket Association Stadium, Mohali
Gymkhana Club Ground
Melbourne Cricket Ground
SuperSport Park
Harare Sports Club
Kensington Oval, Bridgetown
National Cricket Stadium, St George's
Providence Stadium
Sir Vivian Richards Stadium, North Sound
Sheikh Zayed Stadium
Warner Park, Basseterre
Sabina Park, Kingston
Queen's Park Oval, Port of Spain
Beausejour Stadium, Gros Islet
Antigua Recreation Ground, St John's
Sinhalese Sports Club Ground
R Premadasa Stadium
New Wanderers Stadium
Kingsmead
St George's Park
Newlands
Maharani Usharaje Trust Cricket Ground
Keenan Stadium
Nehru Stadium, Fatorda
Nahar Singh Stadium
Feroz Shah Kotla
Eden Park
McLean Park
Queenstown Events Centre
Narayanganj Osmani Stadium
Chittagong Divisional Stadium
Shaheed Chandu Stadium
National Stadium
Multan Cricket Stadium
Gaddafi Stadium
Rawalpindi Cricket Stadium
Westpac Stadium
Jade Stadium
Bellerive Oval
Western Australia Cricket Association Ground
Adelaide Oval
Sydney Cricket Ground
Docklands Stadium
Brisbane Cricket Ground, Woolloongabba
Headingley
Old Trafford
Riverside Ground
Kennington Oval
Lord's
Edgbaston
Trent Bridge
The Rose Bowl
Civil Service Cricket Club, Stormont
venue
50
100
150
200
250
300
350
400
450
total

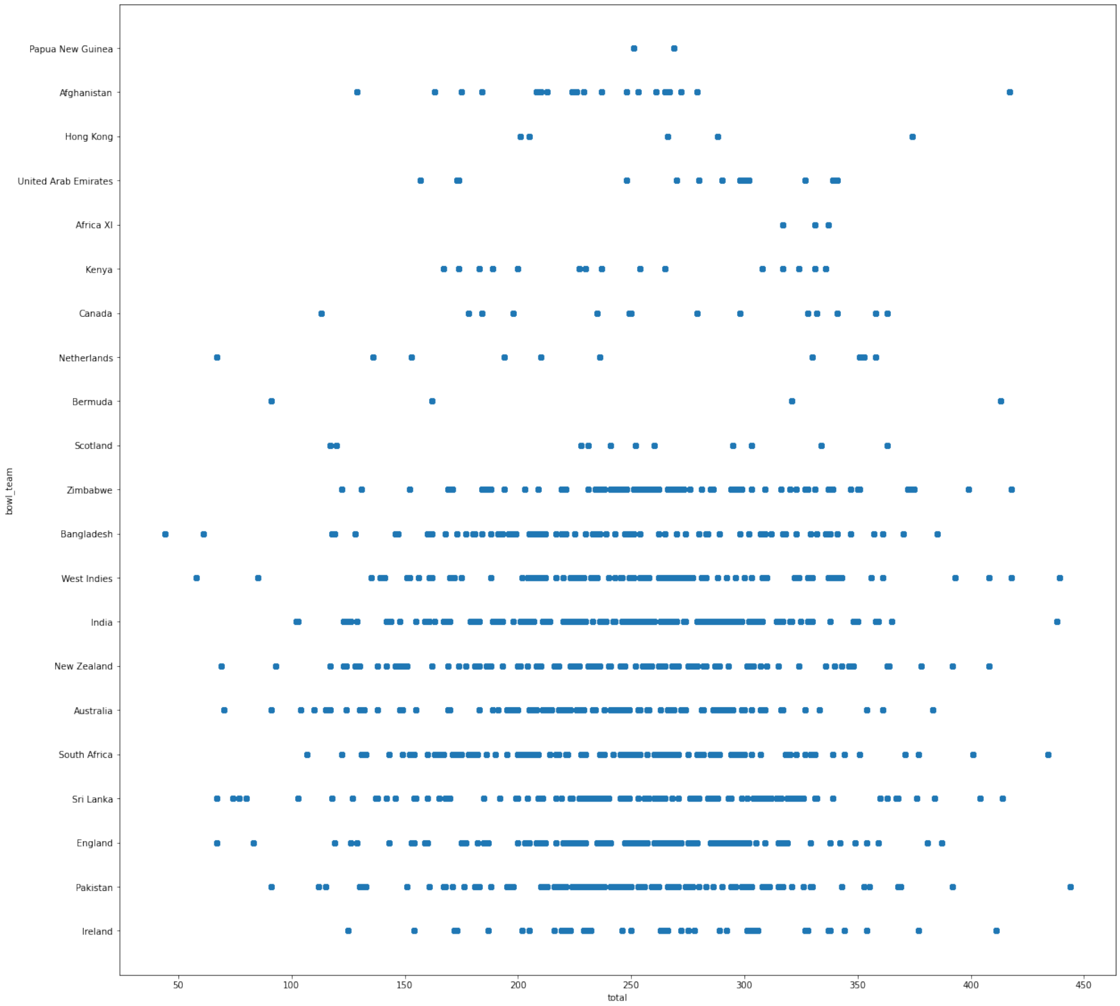
[14]:

```
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()
```



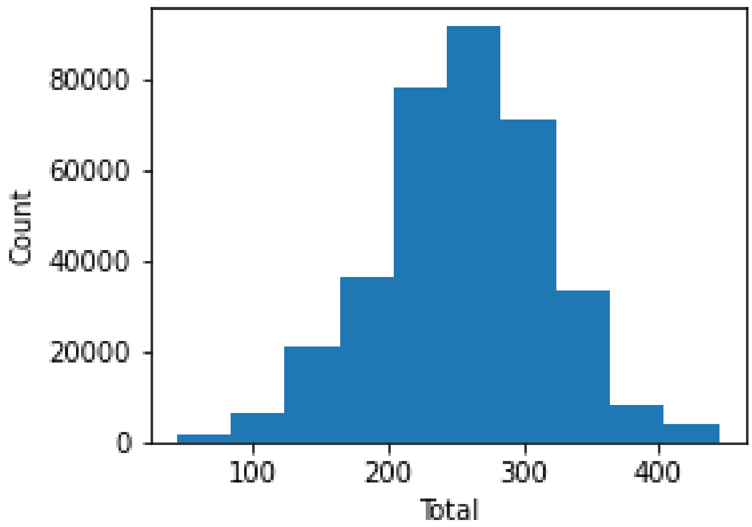
[15]:

```
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     19         6          9      520     168     0        0    0.1
1     1     19         6          9      520     168     0        0    0.2
2     1     19         6          9      520     168     4        0    0.3
3     1     19         6          9      520     168     6        0    0.4
4     1     19         6          9      520     168     6        0    0.5
5     1     19         6          9      520     168     6        0    0.6
6     1     19         6          9      242     137     6        0    1.1
7     1     19         6          9      242     137     6        0    1.2
8     1     19         6          9      242     137     6        0    1.3
9     1     19         6          9      242     137     7        0    1.3
10    1     19         6          9      242     137     8        0    1.4
11    1     19         6          9      520     137     8        0    1.5
12    1     19         6          9      520     137     9        0    1.6
```

```
13    1     19          6          9      520      168     10          0     2.0
14    1     19          6          9      520      168     10          0     2.1

    runs_last_5  wickets_last_5  striker  non-striker  total
0             0               0        0            0    301
1             0               0        0            0    301
2             4               0        0            0    301
3             6               0        0            0    301
4             6               0        0            0    301
5             6               0        0            0    301
6             6               0        0            0    301
7             6               0        0            0    301
8             6               0        0            0    301
9             7               0        0            0    301
10            8               0        1            0    301
11            8               0        1            0    301
12            9               0        1            1    301
13           10               0        1            1    301
14           10               0        1            1    301
```

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])
```

```
[130.    6.    2.   51.  210.   87.    0.   12.1  44.   36. ] 316
```

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: 0.5282371229584477
```

[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: 43.45017573857699
```

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:  316
Predicted Score:  [274.3488753]
Margin of Error:  41.65112470263381
Margin of Error percent 13.180735665390445

Actual Score:  332
```

```
Predicted Score:  [263.5871279]
Margin of Error:  68.41287210288186
Margin of Error percent 20.606286777976464

Actual Score:  253
Predicted Score:  [259.68268178]
Margin of Error:  6.682681782120824
Margin of Error percent 2.6413761984667286

Actual Score:  307
Predicted Score:  [246.1332842]
Margin of Error:  60.86671580112974
Margin of Error percent 19.82629179189894
```

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: [234.46596015]
```

[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: [319.0187636]
```

[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: [250.67227471]
```

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: 0.9911370760900542
```

[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: 99.39298945568538 %
```

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:  316
Predicted Score:  [272]
Margin of Error:  44
Margin of Error percent 13.924050632911392

Actual Score:  332
Predicted Score:  [272]
Margin of Error:  60
Margin of Error percent 18.072289156626507

Actual Score:  253
Predicted Score:  [217]
Margin of Error:  36
Margin of Error percent 14.229249011857707

Actual Score:  307
Predicted Score:  [217]
Margin of Error:  90
Margin of Error percent 29.315960912052116
```

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: [246]
```

[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: [381]
```

[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: [246]
```

14 Random Forest Regressor

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

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)
```

[37]:
```
RandomForestRegressor(max_features=None)
```

```
R square value: 0.9782971377097056
```

[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: 96.09575377600456 %
```

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:  316
Predicted Score:  [220.73]
Margin of Error:  95.27000000000001
Margin of Error percent 30.148734177215196

Actual Score:  332
Predicted Score:  [187.38]
Margin of Error:  144.62
Margin of Error percent 43.56024096385542

Actual Score:  253
Predicted Score:  [224.4]
Margin of Error:  28.599999999999994
Margin of Error percent 11.304347826086955

Actual Score:  307
Predicted Score:  [194.08]
Margin of Error:  112.91999999999999
Margin of Error percent 36.781758957654716
```

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: [165.26]
```

[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: [346.02]
```

[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: [186.59]
```