#IBM-Project

#ASSIGNMENT 4

```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

data =pd.read_csv('/abalone.csv')

data.head()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

```
data.describe()
```

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0.238831	9.933684
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.139203	3.224169
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.001500	1.000000
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.130000	8.000000
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.234000	9.000000
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.329000	11.000000
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.005000	29.000000

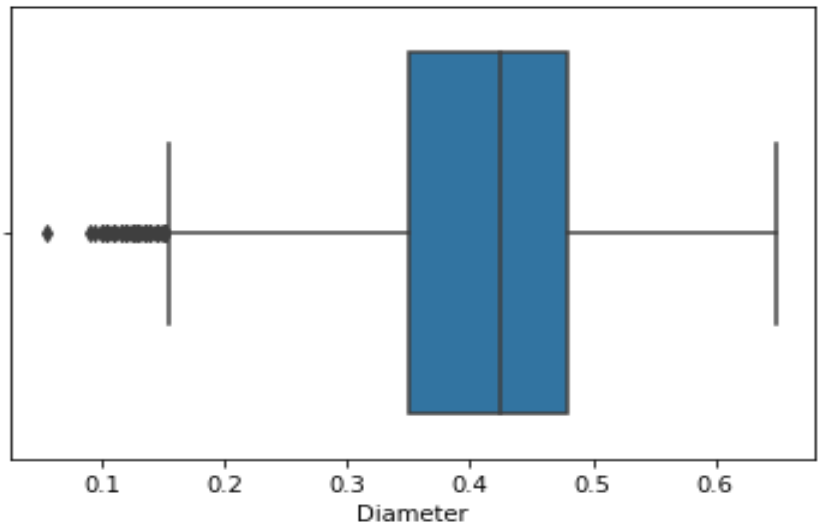
#Perform Visualizations.

####Univariate Analysis

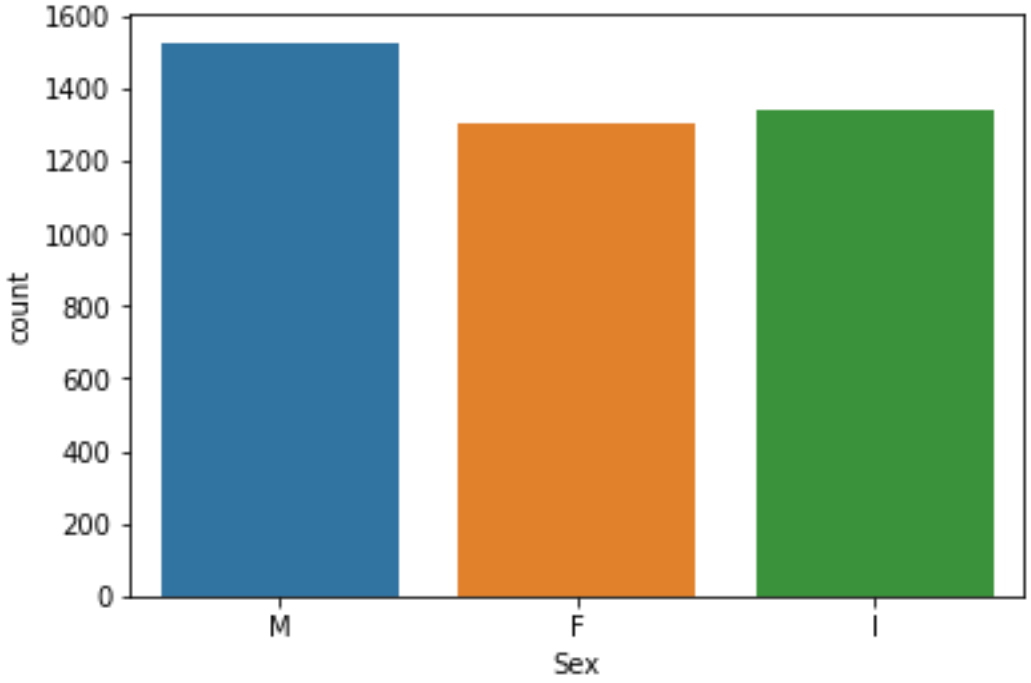
sns.boxplot(data.Diameter)

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

FutureWarning

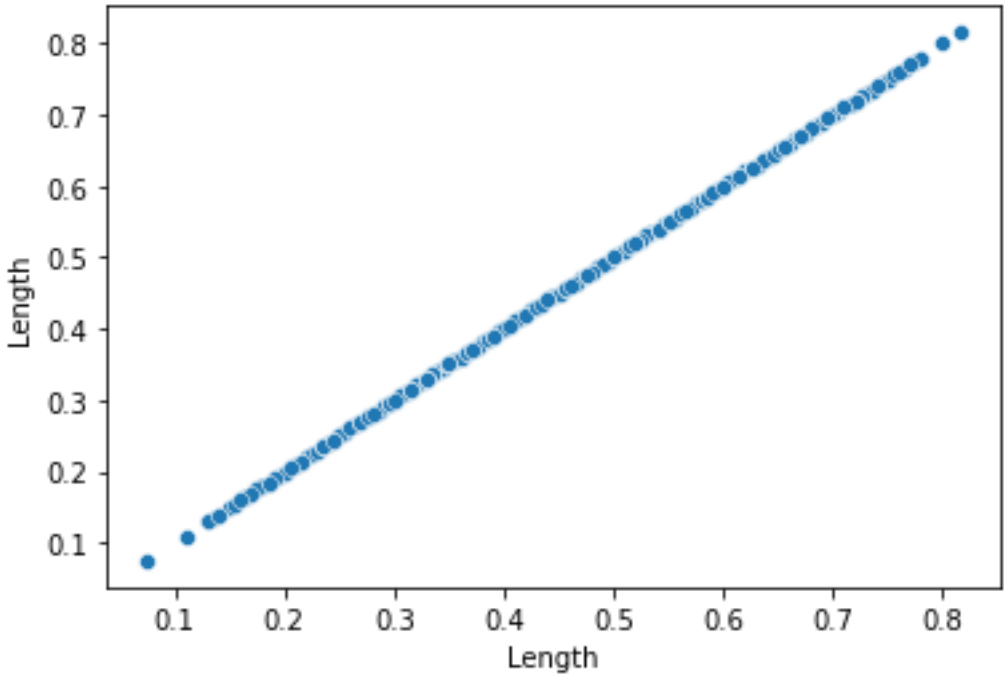


sns.countplot(x='Sex', data=data)

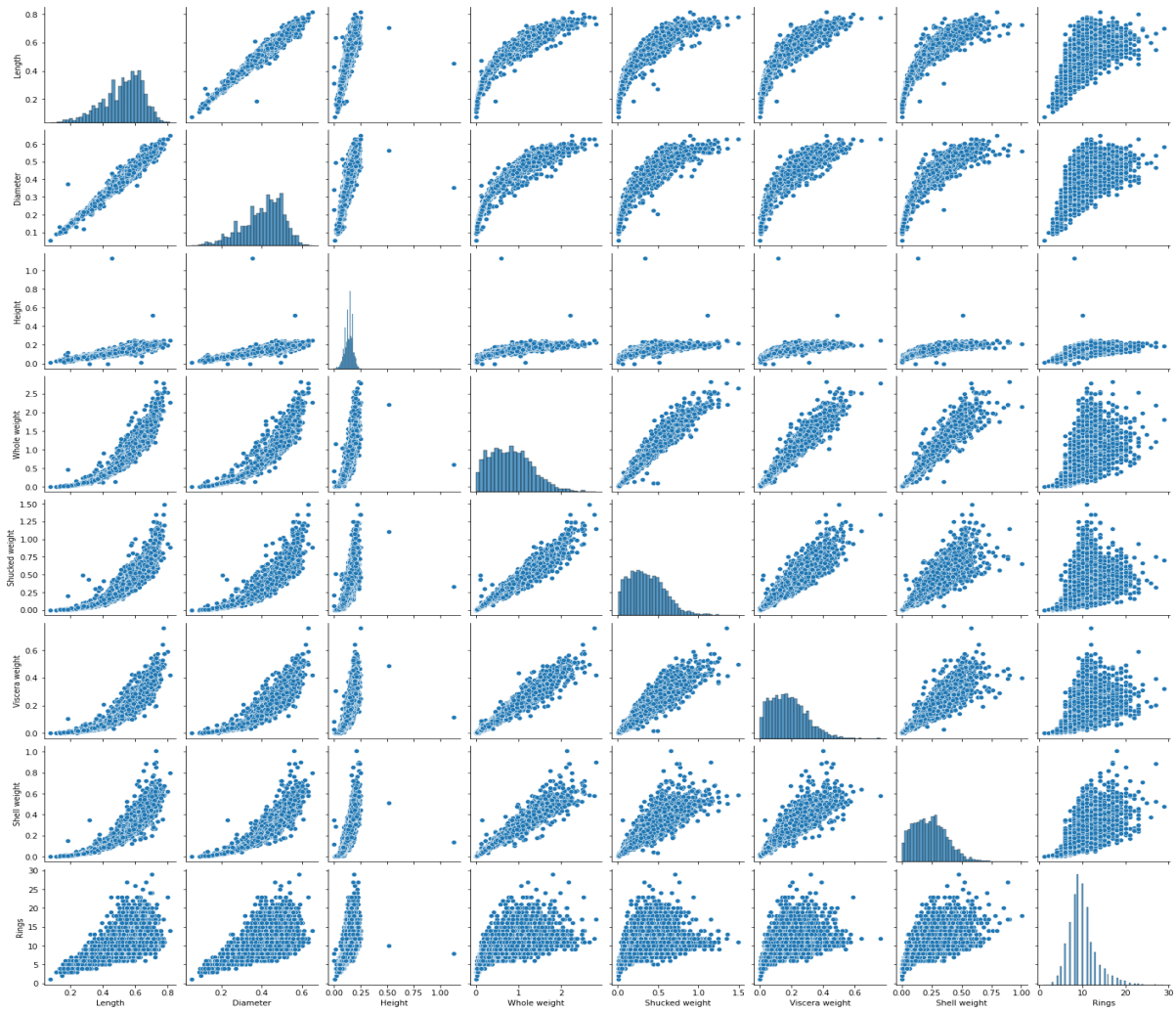


#bivariate analysis

sns.scatterplot(x='Length', y='Length', data=data)



sns.pairplot(data)



##Perform descriptive statistics on the dataset.

data['Height'].describe()

count 4177.000000

mean 0.139516

std 0.041827

min 0.000000

25% 0.115000

50% 0.140000

75% 0.165000

max 1.130000

Name: Height, dtype: float64

data['Height'].mean()

0.13951639932966242

```
data['Length'].mode()
```

```
0    0.550
1    0.625
dtype: float64
```

```
data['Shell weight'].sum()
```

997.5964999999999

```
data['Rings'].max()
```

29

```
data['Whole weight'].min()
```

0.002

```
data[data.Height <= 0]
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
1257	I	0.430	0.34	0.0	0.428	0.2065	0.0860	0.1150	8
3996	I	0.315	0.23	0.0	0.134	0.0575	0.0285	0.3505	6

```
data['Diameter'].median()
```

0.425

```
data['Shucked weight'].skew()
```

0.7190979217612694

##Check for Missing values and deal with them.

```
data.isnull().any()
```

```
Sex               False
Length            False
Diameter          False
Height            False
Whole weight      False
Shucked weight    False
Viscera weight    False
Shell weight      False
Rings             False
```

dtype: bool

```
missing_values = data.isnull().sum().sort_values(ascending = False)

percentage_missing_values = (missing_values/len(data))*100

pd.concat([missing_values, percentage_missing_values], axis = 1, keys= ['Missing values', '% Missing'])
```

	Missing values	% Missing
Sex	0	0.0
Length	0	0.0
Diameter	0	0.0
Height	0	0.0
Whole weight	0	0.0
Shucked weight	0	0.0
Viscera weight	0	0.0
Shell weight	0	0.0
Rings	0	0.0

##Find the outliers and replace them outliers

```
q1=data.Rings.quantile(0.25)

q2=data.Rings.quantile(0.75)

iqr=q2-q1

print(iqr)
```

3.0

#Check for categorical columns and perform encoding

```
data['Sex'].replace({'M':1, 'F':0, 'I':2},inplace=True)

data.head()
```

Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	1	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	1	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	2	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

#Split the data into dependent and independent variables.

```
x = data.iloc[:,:-1].values
y = data.iloc[:, -1].values
##Scale the independent variables
from sklearn.preprocessing import StandardScaler
std=StandardScaler()
x=std.fit_transform(x)
print(x)
[[-0.0105225  -0.57455813 -0.43214879 ... -0.60768536 -0.72621157
  -0.63821689]
 [-0.0105225  -1.44898585 -1.439929   ... -1.17090984 -1.20522124
  -1.21298732]
 [-1.26630752  0.05003309  0.12213032 ... -0.4634999  -0.35668983
  -0.20713907]
 ...
 [-0.0105225   0.6329849   0.67640943 ...  0.74855917  0.97541324
   0.49695471]
 [-1.26630752  0.84118198  0.77718745 ...  0.77334105  0.73362741
   0.41073914]
 [-0.0105225   1.54905203  1.48263359 ...  2.64099341  1.78744868
   1.84048058]]
##Split the data into training and testing
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
x_train
array([[ 1.24526253,  0.21659075,  0.37407537, ..., -0.32156733,
        -0.4023098 , -0.20713907],
       [ 1.24526253, -0.40800047, -0.53292681, ..., -0.47701729,
        -0.81745151, -0.71006319],
       [-1.26630752, -1.82374058, -1.84304108, ..., -1.35564747,
        -1.34208115, -1.39260308],
       ...,
```

```
       [-0.0105225 , -0.11652457, -0.12981473, ..., -0.51982235,
        -0.42968178, -0.36520094],
       [-0.0105225 ,  0.42478783,  0.57563141, ...,  0.70575411,
         0.34585768, -0.02752331],
       [-1.26630752,  0.59134549,  0.67640943, ...,  0.84543378,
         0.4599076 ,  0.23112338]])
print(y_train)
[11  8  7 ... 11  9  6]
print(x_test)
[[-0.0105225   0.67462432  0.47485339 ...  0.27770351  1.10314916
   0.61909342]
 [-0.0105225   0.54970607  0.32368636 ...  0.12450645  0.3139237
   0.04432299]
 [-1.26630752  0.29986958  0.37407537 ... -0.2449688   0.40060164
   0.69093973]
 ...
 [ 1.24526253  0.17495134  0.22290834 ... -0.0309435  -0.20614393
  -0.22150833]
 [ 1.24526253 -0.4912793  -0.53292681 ... -0.47025859 -0.81288951
  -0.39393946]
 [ 1.24526253 -1.3240676  -1.33915098 ... -1.17766853 -1.30558517
  -1.17706417]]
print(y_test)
[ 9  8 16  9 14 11  7  6  7 10 22  7 15  9  8 18 11 14 13  9 20 12 12 11
 10  7 11  8  9 10  9 10  6 10  8  9  5  3  6  6 12 12 18  8 12 13 10 10
 18  4  6 22  8  5  7 10 15 21 10  9 10 13 11  7  9 11  4  5  7  9 10 11
 10  7  9 12 23 14 15  9 15 13 10  6  7 13  9 10 19 10 10  9 11 11 10 10
  6 15  7  7 15 11 11 13  7  9 10  8  9 14 18  8 13  9 12  5  9 12 11 13
 11 10  8 14  9 20  9  9  9 10  9  9 10  5  8  8 10 10  5 12  8 11  7  8
 10 15 10 14 10 10 10  8 11 11  8 11 12  7  8  6  9  6 10 12  7 10 17 11
  8  8 10 12  9  8  8  7  9 11  9 10 13  7  8  8  7 10  8 11  9  5  9  8
```

16 13 11 17 10 11 12 9 8 17 11 12 9 12 11 9 8 10 5 9 12 6 8 11

11 7 9 12 13 9 12 11 9 8 7 13 9 12 5 10 10 12 7 10 10 7 4 10

 8 11 10 9 10 8 9 7 7 6 7 9 9 7 15 11 9 5 12 14 19 16 9 9

 7 6 7 14 12 6 9 8 6 12 8 18 10 16 9 6 15 9 13 8 5 9 10 5

10 10 11 4 15 9 15 8 5 14 7 11 10 10 7 10 9 10 18 8 6 5 8 6

 7 14 12 10 5 23 9 9 12 7 8 8 13 6 13 17 7 8 8 7 7 9 14 10

 9 13 8 10 10 9 10 9 8 8 10 13 10 9 8 10 8 11 10 3 7 6 3 8

 8 13 15 6 14 8 9 12 8 8 15 11 9 6 10 13 13 7 7 9 9 8 7 10

11 5 10 12 8 7 9 6 8 13 7 7 10 11 23 9 11 10 8 8 7 7 10 9

 9 15 13 14 7 9 9 6 4 6 10 10 11 10 11 9 9 8 17 9 10 9 8 7

11 9 10 6 15 7 9 11 7 19 10 11 8 11 10 9 10 10 8 9 8 5 11 10

10 7 12 7 8 11 9 10 8 5 14 9 9 8 9 10 8 7 9 7 16 16 6 10

 8 17 10 18 9 10 11 10 10 7 12 14 7 11 10 12 14 9 9 10 12 14 4 12

10 7 6 7 9 12 10 12 8 10 10 8 4 11 10 16 10 10 11 10 5 6 13 11

19 8 8 9 4 19 12 8 12 10 9 11 7 4 8 10 6 6 8 7 10 9 5 9

14 10 8 6 9 7 9 7 11 6 11 11 8 9 16 9 7 9 10 8 13 9 10 11

 9 20 6 7 7 7 10 9 18 21 9 10 11 11 11 11 9 9 9 14 7 7 7 12

 6 16 13 23 11 11 14 6 10 8 8 11 4 11 10 12 12 17 7 9 14 7 7 6

 9 10 15 11 8 10 7 8 11 9 15 11 8 13 9 11 9 7 4 9 12 8 11 9

14 11 11 7 6 16 10 13 5 6 9 16 9 11 13 18 9 10 12 15 5 19 7 6

22 10 10 10 8 13 10 8 9 11 5 8 10 9 21 9 11 10 10 11 9 9 14 12

10 18 20 8 13 10 9 11 11 9 6 8 9 5 10 9 8 5 12 10 16 9 9 11

 9 9 9 9 9 9 9 12 7 7 11 8 7 9 9 6 10 17 9 5 10 9 6 15

 9 10 6 11 7 10 6 9 10 10 11 8 11 7 11 11 8 10 8 10 18 12 9 12

10 6 10 7 12 8 11 14 5 8 11 9 14 10 12 10 16 9 10 7 6 11 4 5

10 11 10 12 15 14 12 5 11 6 9 8 7 4 9 12 11 10 11 11 10 5 19 11

11 18 7 8 10 9 9 10 13 5 5 6 8 8 15 11 8 10 8 7]

##Build and train the Model

```
from sklearn.ensemble import RandomForestRegressor

model=RandomForestRegressor(n_estimators=1000, oob_score=True, n_jobs=-1,
min_samples_split=6, min_samples_leaf=4, max_features='sqrt', max_depth=120)
```

```
model.fit(x_train, y_train)
RandomForestRegressor(max_depth=120, max_features='sqrt', min_samples_leaf=4,
                      min_samples_split=6, n_estimators=1000, n_jobs=-1,
                      oob_score=True)
##Test the model
pred=model.predict(x_test)
print(pred)
[11.94473157  9.52492659 14.33955884 10.84536003 11.9982424  10.43245678
 8.92075719  8.76490086  6.72880933 10.63993945 11.64994056  7.8047387
 12.58213916  7.28950173  8.32118321 13.18811499 12.14302326 10.44545696
 13.40174628  8.04532015 14.3333229  10.62085832 10.56069455 10.51564261
 9.26325673  7.95836514  9.90105967  7.93271282  9.21533156 11.14530691
 11.72470205 10.44326274  6.91120384 11.73242601  8.64362528  9.35847723
 6.1038035   3.05632245  8.44121845  6.45678267 11.55679136 15.04404998
 11.24420358  8.908379   11.61017414 11.40841541  9.90437003  8.90318983
 12.89715199  4.46239039  6.30259639 16.56100413  9.18566148  3.91541034
 6.47980493 10.99104364 10.44425056 12.96902071  9.68966851  8.73391989
 11.13037535 14.30519877 10.13616805  7.02888586  7.9613865  10.76215261
 4.36549326  6.57031363  8.42659707 11.66126198 11.14267812 11.49863884
 9.48105399  8.36584664  8.9397942  11.08992218 15.06523157 10.6458405
 11.32273726  9.18216359 12.42133556 11.95790845 11.00735698  8.61337769
 6.8489838  12.15785946  9.48678168  9.56737526 11.30192434 12.86382261
 11.3462735  10.95393587 10.37363807 11.45126035 11.05134767  9.71958137
 6.79472658 10.39434192 11.40861704  6.57683992 13.61547693 10.84243496
 11.50178301 10.84003837  9.28764637 10.29027507 10.83194911  9.55927703
 10.05396066 10.68376174 11.37438655  9.04254695 12.30662996  9.25432218
 10.62588582  4.83582507  9.3132854  10.48594236 10.88238619 12.72530299
 10.02089278 10.24711508  9.91701948 11.0727653  11.32336203 13.34646379
 10.36469635  9.89053539 11.27695187 14.94499504  9.40135317  8.19444786
 10.36111954  4.70817512  5.80545883  8.97890066 10.01137051 11.61073539
 5.83197481 12.11975008  8.98240375  9.91013071  9.21554341 10.64927558
```

11.03398217 12.87440421 10.99653432 12.2085514 9.33016181 12.63328568
8.96215979 11.37996338 9.74139647 10.17665443 9.46184385 10.16157737
12.72201588 5.58100943 8.94777186 6.87179119 10.55668867 6.30979976
10.11546316 12.4120683 7.40067597 11.93595319 14.2682589 9.95734989
10.64963534 9.52432197 10.86630185 11.24303055 11.42071475 8.6627275
6.98790566 9.63615165 8.78671944 11.42506965 9.51334933 9.24420407
12.13257233 9.27508685 6.69459261 9.20611828 6.97057055 10.67369669
8.39316472 10.40270478 10.2470234 5.50055323 8.57383633 10.36069652
12.86663819 11.0536439 10.1914634 11.19762775 11.3221326 13.3715373
13.64890142 10.63045682 8.34697228 12.20939309 12.83098717 12.36902338
7.9467357 11.41478368 10.61802559 9.70861084 8.50506587 10.35024072
5.93944323 10.47941918 12.31503328 6.82205659 10.36283479 12.43805364
11.13175997 6.83078424 10.28078005 15.2407508 11.14702211 9.89555353
12.69082239 12.24440492 9.5154395 6.8007093 8.08556354 12.26630521
9.87170156 12.8559529 4.90807399 12.57299331 10.68112723 14.11932438
8.21330188 10.86999933 9.74125666 8.152788 5.65384709 9.80911683
10.38853136 14.79204527 9.36592176 7.89578346 9.73546427 8.59556786
10.9036502 7.66236623 8.75161817 7.48718139 7.61862261 10.95238276
11.10330926 6.79789074 11.22291277 12.34821362 10.03988145 4.70817126
13.59062547 10.72854307 13.10444809 10.20578232 9.0350292 9.23199204
8.70571959 5.8634368 5.56737304 12.76518855 10.19443707 7.19045939
9.65125425 8.99644894 7.33405161 10.34156208 9.48392456 11.06971776
11.27333513 14.51737249 10.64935688 5.0388614 14.46982248 8.20500416
10.58509146 8.73057126 5.3403121 11.4221692 10.71528171 4.22890426
11.15793167 11.98054568 10.1286364 4.22090861 12.82565136 8.14953074
12.31159298 10.91686738 4.4948195 15.18302857 7.27569239 9.88324908
10.58322186 9.80631032 6.82563962 11.67957333 9.89228671 8.7449724
13.21055831 8.05250526 5.53354291 5.59013749 9.4455929 7.15175348
7.43604011 9.47389963 9.67598171 8.85856464 4.40850572 14.98024455
9.57665174 8.51710751 11.58323531 8.12182743 7.02379302 10.07499369
9.70457263 6.59904055 12.80530731 14.13749296 9.30193165 9.70496264

8.8134583 12.28073028 7.22517329 9.64381243 11.62594477 10.0986394
9.95879348 13.19989825 9.31019535 10.90626442 9.94848437 10.30627217
11.94493133 9.79665994 8.89329914 10.83967114 10.10448181 10.96609449
10.2432141 8.75781595 11.62307063 11.00132809 9.26676461 10.75837537
10.73847053 3.59295017 7.24861341 7.09596242 3.1250182 8.82863012
10.31040155 8.59871671 13.27163888 7.56512256 9.69840121 7.76492521
10.79404903 9.60966527 9.98791428 6.20909059 15.59277441 11.53454196
7.87797522 7.19655076 11.70974426 13.57613856 8.63745038 8.67008991
8.48803062 9.90254943 9.50751903 7.76303564 9.38626989 10.41927238
10.59446468 4.7862898 13.51394983 11.81906747 9.10445355 7.91674721
10.41793586 8.6666572 10.45192738 12.40861568 8.97558944 9.08675713
9.44908787 13.11980322 11.97759543 9.38082957 11.85610244 8.79729839
11.05486734 8.54308161 6.8916259 8.37703559 11.08474315 9.31784055
11.51825761 10.53499865 13.72151262 12.99812789 7.03146141 10.83020951
12.11495544 5.79392399 4.1795482 9.87476098 10.61288852 11.00180913
9.23423734 11.66829103 11.65663205 10.6334049 10.68380985 10.57966156
11.99158838 10.23337048 11.249736 8.96411569 10.31444662 10.00626995
11.59046002 9.94159565 11.98834366 6.87786371 11.3903104 6.03299967
10.2437933 9.96165489 6.39927975 12.36682915 10.51478439 8.49349784
10.43647531 9.77640374 9.65088644 10.69564215 12.65243216 11.50969062
8.57788883 10.12623097 7.16653073 7.76419913 7.96658624 11.64707859
7.21685386 8.9753022 12.102908 6.92770291 8.61896926 11.99204179
10.83319325 10.40028895 8.60732964 5.628796 11.00462396 10.92081833
7.93914831 8.52314821 10.40911949 12.53727224 7.99877964 9.16548875
10.41878318 7.79168394 15.37631741 16.83897131 7.48734881 9.48638248
11.28843497 11.22948025 8.93001237 13.4012675 10.01226269 10.66673156
12.43832284 11.07758332 10.04265967 6.64484907 12.06994741 13.01144897
7.63619629 10.37347131 12.06796721 9.65988746 12.01833079 9.68309629
8.51957635 10.51780403 8.01189901 11.98239281 4.22503857 8.979944
9.38903563 7.84620535 7.00476423 9.87249209 12.1597806 10.68500926
9.92800004 10.84276297 9.12892174 10.13454183 8.44672367 8.81998069

4.43885168 11.22610128 9.99992279 11.15013284 9.41224257 7.87675098
12.74370086 8.42413228 6.17968222 8.295297 9.84226301 12.05580174
16.27156852 9.59448091 8.05820649 9.98941618 4.32597998 15.32353153
11.92514978 6.8019514 12.73690685 10.14862149 9.65173545 10.66465316
7.44403505 5.29667552 9.10678854 11.11792994 5.27690873 7.68875276
9.15840663 9.77645092 10.87771704 8.81828765 6.63061853 9.50634106
14.99513682 9.93116533 7.51604961 7.17768986 10.37506244 6.6348117
8.61874582 9.69776336 11.64816122 9.17272413 9.25308972 9.70944179
7.80580906 12.34730302 12.25932391 10.63279878 6.23749282 9.68108717
8.98658056 7.55304058 9.52357084 8.53115739 9.2530326 11.53090174
7.14583162 10.78297507 7.3733705 7.46935859 8.51818201 7.14664817
9.28418158 6.88595999 13.47967746 12.54846066 10.14554201 12.25829308
11.27924943 12.00030414 11.47313097 10.1781377 9.46377612 8.52621968
9.17164221 9.94064407 6.65167353 8.81320453 8.28357324 11.05763459
7.3646045 13.07845534 11.36300261 12.1477895 13.79325249 12.93047709
12.71755667 5.96128352 10.88502181 10.7820109 7.44663133 7.8909081
4.17604337 11.60029949 9.46597529 10.48076601 12.76846149 13.38322129
7.00734882 8.56732765 10.41458218 7.4376855 9.45489298 6.7978318
8.58124042 10.39480727 12.76451004 12.87727875 10.41980712 11.89687013
9.2911082 8.33492046 10.50853338 9.0534952 11.12553463 11.35529824
9.18484588 11.96948119 10.51741984 11.47736113 9.03684532 6.37730231
5.35654562 11.48966559 10.32019192 9.11677018 12.95944668 9.26842077
10.37273936 12.79623404 11.70022988 7.10594448 5.45836942 14.47416158
11.43670069 10.77609577 5.54686589 7.34874157 10.94723962 13.90796619
10.07176143 6.82479191 12.26963483 15.63537894 10.06857317 8.37949058
13.37512335 11.24860165 4.30935592 12.90822203 7.232458 9.05789272
11.72484484 8.67738642 9.87668418 13.74254049 8.26064449 13.14326441
11.34776118 9.50021562 10.33663481 10.51583929 5.57970232 9.47979598
11.87456957 9.77454975 14.61887837 10.19144873 8.02846671 7.16714883
9.6994995 10.4670837 10.82196071 10.49506982 13.8709429 7.70423128
12.17252426 11.94396317 13.93791151 9.89578405 13.7758245 11.4660944

9.22706697 10.33181462 11.15171454 11.00037304 6.36400021 7.74935767
 9.26583658 4.85339908 11.38448506 9.72383886 9.49485277 5.72247788
11.80723147 9.10203402 13.98319194 10.1361182   9.41581147 8.90653835
10.88911175 11.54842399 8.06864349 10.39605062 10.71421084 11.13198741
12.41946474 8.4633108   7.01545096 6.49467439 12.55024858 7.57174465
 9.38965174 9.46637897 12.8662497   4.60755263 10.47880078 11.38627333
 9.50770132 6.66967255 11.72893007 11.54653023 6.53107911 12.87044602
10.61853203 13.22649925 6.75958821 10.27101925 8.86564841 11.8998275
 6.25976319 10.47451848 13.04310737 15.40733411 11.3280577   8.69338984
11.27787702 7.65889956 11.04728095 9.22826477 10.18361591 11.63857463
 9.98676471 11.57190083 10.27317369 10.89599272 9.97244121 11.30511412
 8.53149666 7.50087678 9.70005568 9.12644551 11.40964433 7.86179103
11.20791818 10.78303607 4.53995388 9.02114854 12.01205207 10.87876816
15.62058844 12.18046293 11.58291044 10.54562279 14.08614396 11.58504898
13.73672139 7.93856826 6.84075831 9.94530402 3.27512974 6.37572101
15.21775433 12.68520886 9.99599269 12.19690134 13.73750356 12.61241292
10.29215838 3.58361686 11.64568934 6.42438426 12.49114688 9.81540132
 9.07383926 4.12939977 9.19301404 15.06705375 11.37127629 10.78427119
10.80530065 9.94285707 12.05637461 6.26786724 11.85359602 10.39807176
11.25217453 10.2498546   7.95469174 7.54384969 9.39819449 9.49042143
11.63646916 9.11875446 7.84949005 6.68392265 4.06435364 6.79859508
 8.20750748 10.58126607 11.77293922 10.91516048 9.78358833 9.16779968
 9.0780574   6.96849654]

##measure the performance using the metrics

```
from sklearn.metrics import r2_score
acc=r2_score(y_test,pred)
acc
```

0.5563741745933068

#END