In [63]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading the dataset

In [64]:

```
data=pd.read_csv("/abalone.csv")
data
```

Out[64]:

	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	0.1500	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7
...	...	...	...	...	...	...	...	...	...
4172	F	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11
4173	M	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	10
4174	M	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9
4175	F	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10
4176	M	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12

4177 rows × 9 columns

In [65]:

```
data.head()
```

Out[65]:

	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

Performing visualization

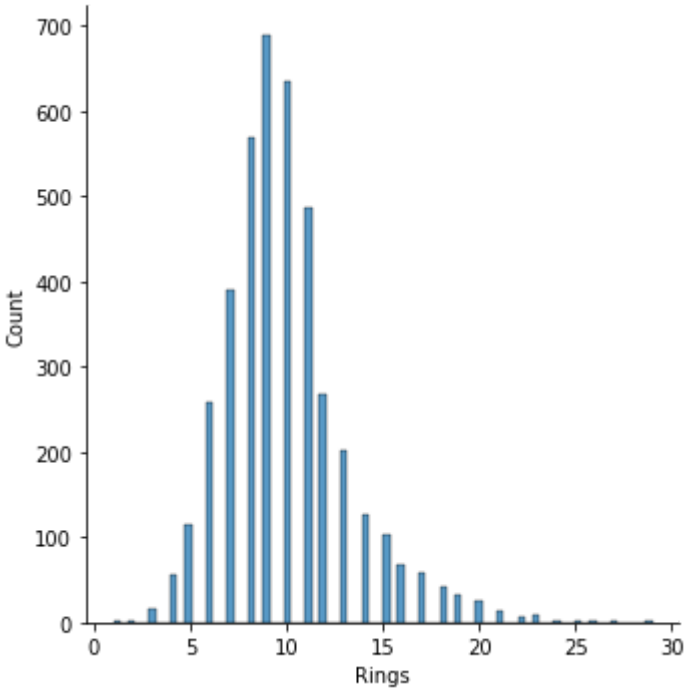
Univariant Analysis

In [66]:

```
sns.displot(data.Rings)
```

Out[66]:

```
<seaborn.axisgrid.FacetGrid at 0x7fe75dbb2090>
```

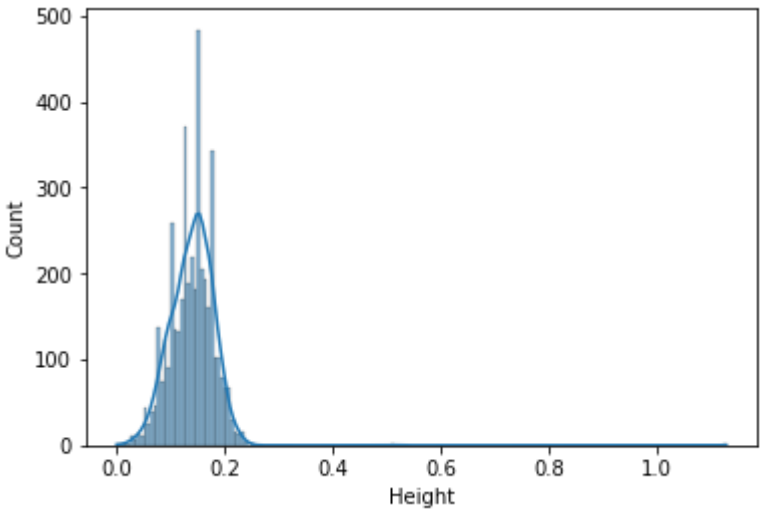


In [67]:

```
sns.histplot(data.Height,kde=True)
```

Out[67]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe75b694810>
```



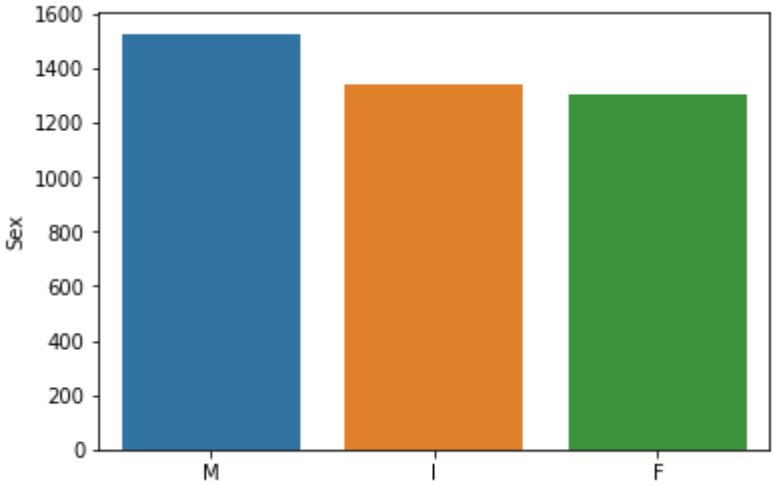
In [68]:

```
sns.barplot(data.Sex.value_counts().index,data.Sex.value_counts())
```

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

Out[68]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe75b2d6cd0>
```



Bivariant Analysis

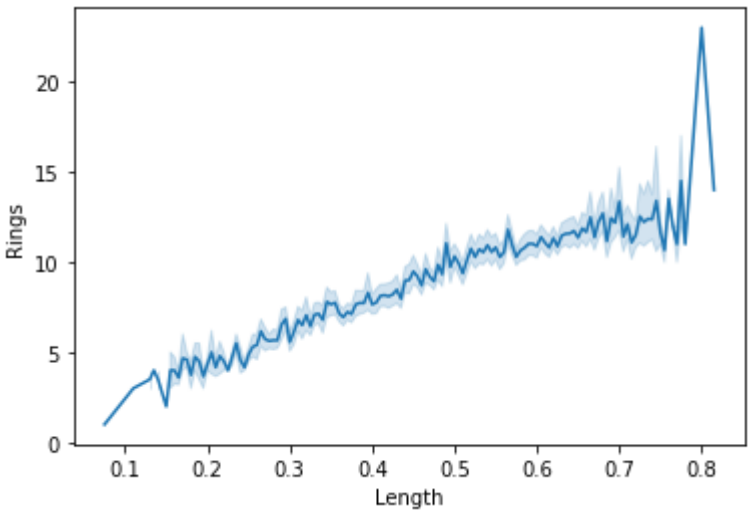
In [69]:

```
sns.lineplot(data.Length,data.Rings)
```

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

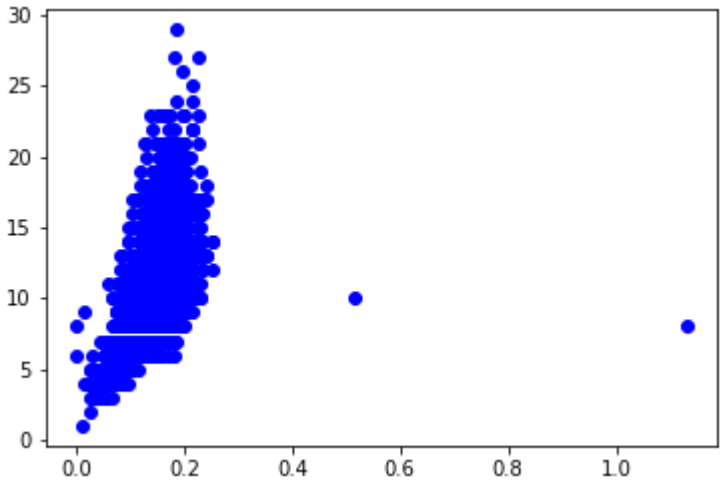
Out[69]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe75b2b46d0>
```



In [70]:

```
plt.scatter(data.Height,data.Rings,c='blue')
plt.show()
```

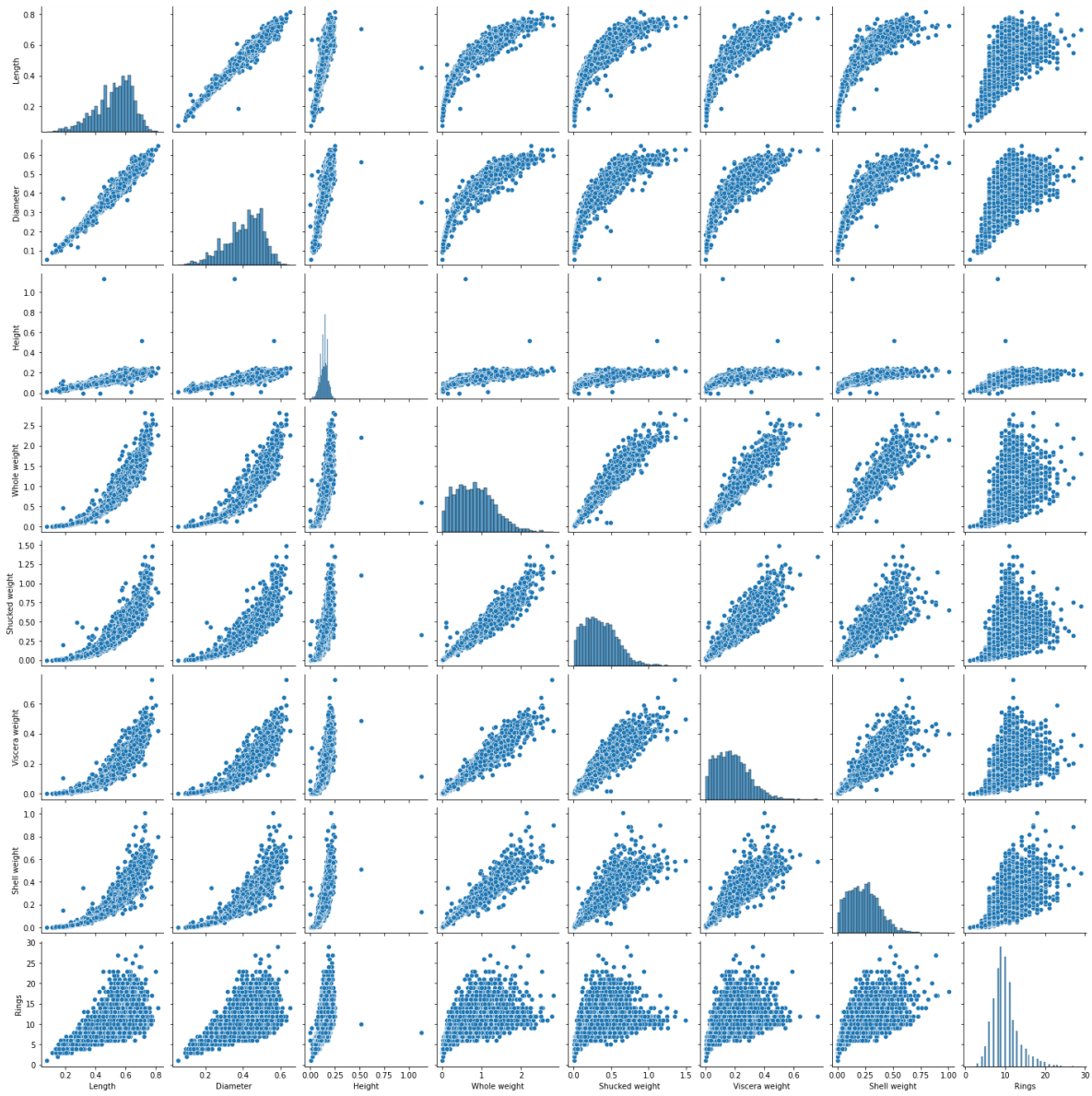


In [71]:

```
sns.pairplot(data)
```

Out[71]:

```
<seaborn.axisgrid.PairGrid at 0x7fe75b1fb750>
```



Multivariant Analysis

In [72]:

```
data.corr()
```

Out[72]:

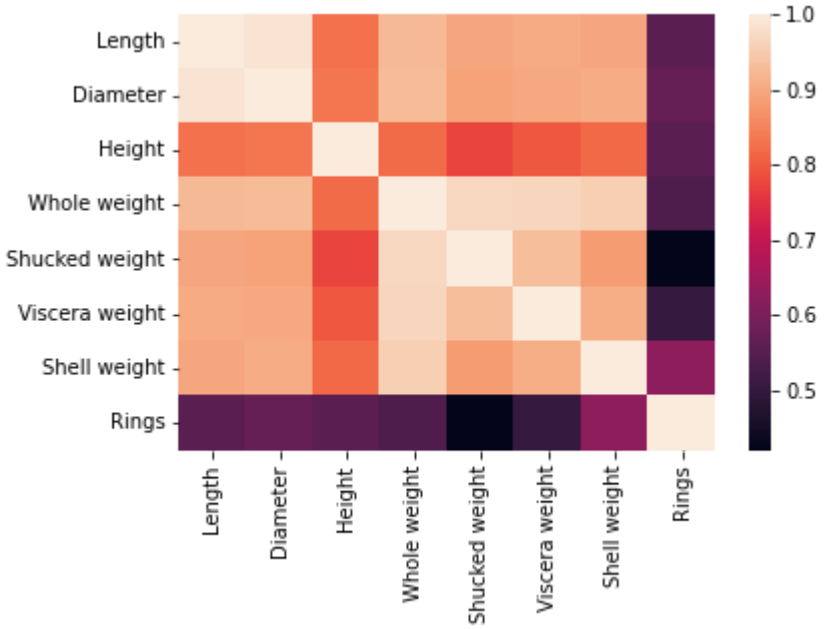
	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
Length	1.000000	0.986812	0.827554	0.925261	0.897914	0.903018	0.897706	0.556720
Diameter	0.986812	1.000000	0.833684	0.925452	0.893162	0.899724	0.905330	0.574660
Height	0.827554	0.833684	1.000000	0.819221	0.774972	0.798319	0.817338	0.557467
Whole weight	0.925261	0.925452	0.819221	1.000000	0.969405	0.966375	0.955355	0.540390
Shucked weight	0.897914	0.893162	0.774972	0.969405	1.000000	0.931961	0.882617	0.420884
Viscera weight	0.903018	0.899724	0.798319	0.966375	0.931961	1.000000	0.907656	0.503819
Shell weight	0.897706	0.905330	0.817338	0.955355	0.882617	0.907656	1.000000	0.627574
Rings	0.556720	0.574660	0.557467	0.540390	0.420884	0.503819	0.627574	1.000000

In [73]:

```
sns.heatmap(data.corr())
```

Out[73]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe759b07310>
```

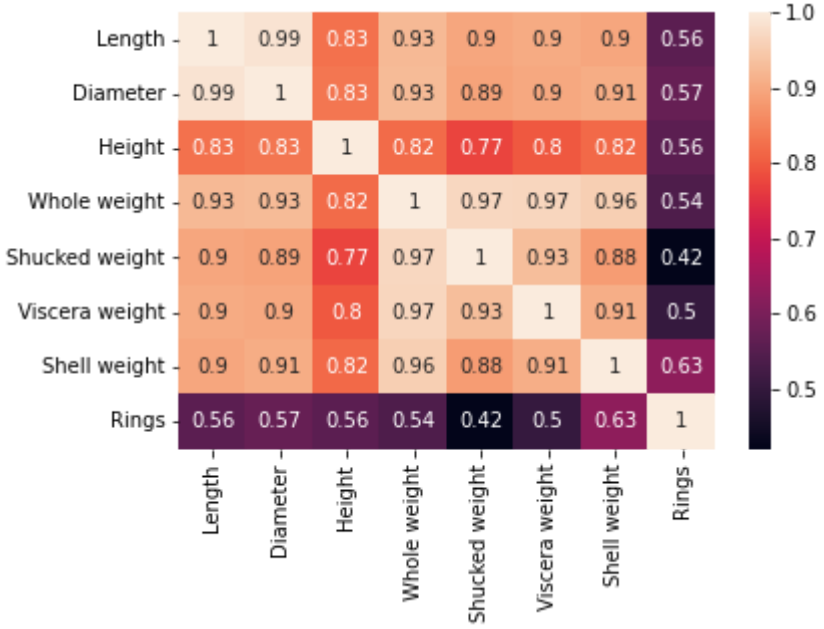


In [74]:

```
sns.heatmap(data.corr(),annot=True)
```

Out[74]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe759267d50>
```



Descriptive Analysis of Data

In [75]:

```
data.describe()
```

Out[75]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell we
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0.238
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.139
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.001
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.130
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.234
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.329
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.005

In [76]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Sex             4177 non-null   object 
 1   Length          4177 non-null   float64
 2   Diameter        4177 non-null   float64
 3   Height          4177 non-null   float64
 4   Whole weight    4177 non-null   float64
 5   Shucked weight  4177 non-null   float64
 6   Viscera weight  4177 non-null   float64
 7   Shell weight    4177 non-null   float64
 8   Rings           4177 non-null   int64  
dtypes: float64(7), int64(1), object(1)
memory usage: 293.8+ KB
```

In [77]:

```
data.shape
```

Out[77]:

```
(4177, 9)
```

In [78]:

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

Out[78]:

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

In [79]:

```
data.Sex.value_counts()
```

Out[79]:

```
M    1528
I    1342
F    1307
Name: Sex, dtype: int64
```

In [80]:

```
data.Length.value_counts()
```

Out[80]:

```
0.625    94
0.550    94
0.575    93
0.580    92
0.600    87
         ..
0.075     1
0.815     1
0.110     1
0.150     1
0.800     1
Name: Length, Length: 134, dtype: int64
```

Checking for Missing values

In [81]:

```
data.isnull().sum()
```

Out[81]:

```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
Rings             0
dtype: int64
```

In [82]:

```
data['Sex'].fillna(data['Sex'].mode(),inplace=True)
```

In [83]:

```
np.where(data['Shell weight']>0.7)
```

Out[83]:

```
(array([ 129,  163,  164,  165,  166,  168,  334,  891, 1428, 2108, 2157,
        2161, 3008, 3149, 3151]),)
```

Performing label encoding for Gender

In [84]:

```
from sklearn.preprocessing import LabelEncoder
gender=LabelEncoder()
```

In [85]:

```
gender.fit(data['Sex'])
```

Out[85]:

```
LabelEncoder()
```

In [86]:

```
values=gender.transform(data['Sex'])
```

In [87]:

```
values
```

Out[87]:

```
array([2, 2, 0, ..., 2, 0, 2])
```

In [88]:

```
data['Sex'].unique()
```

Out[88]:

```
array(['M', 'F', 'I'], dtype=object)
```

In [89]:

```
values[:10]
```

Out[89]:

```
array([2, 2, 0, 2, 1, 1, 0, 0, 2, 0])
```

In [90]:

```
data['Sex'][:10]
```

Out[90]:

```
0    M
1    M
2    F
3    M
4    I
5    I
6    F
7    F
8    M
9    F
Name: Sex, dtype: object
```

In [91]:

```
data['new_sex']=values
```

In [92]:

```
data
```

Out[92]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings	new_sex
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500	15	2
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7	2
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9	0
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10	2
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7	1
...	...	...	...	...	...	...	...	...	...	...
4172	F	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11	0
4173	M	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	10	2
4174	M	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9	2
4175	F	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10	0
4176	M	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12	2

4177 rows × 10 columns

Spliting dependent and independent variables

In [93]:

```
x=data.loc[:,data.columns.difference(['Sex','Rings'])]
x
```

Out[93]:

	Diameter	Height	Length	Shell weight	Shucked weight	Viscera weight	Whole weight	new_sex
0	0.365	0.095	0.455	0.1500	0.2245	0.1010	0.5140	2
1	0.265	0.090	0.350	0.0700	0.0995	0.0485	0.2255	2
2	0.420	0.135	0.530	0.2100	0.2565	0.1415	0.6770	0
3	0.365	0.125	0.440	0.1550	0.2155	0.1140	0.5160	2
4	0.255	0.080	0.330	0.0550	0.0895	0.0395	0.2050	1
...	...	...	...	...	...	...	...	...
4172	0.450	0.165	0.565	0.2490	0.3700	0.2390	0.8870	0
4173	0.440	0.135	0.590	0.2605	0.4390	0.2145	0.9660	2
4174	0.475	0.205	0.600	0.3080	0.5255	0.2875	1.1760	2
4175	0.485	0.150	0.625	0.2960	0.5310	0.2610	1.0945	0
4176	0.555	0.195	0.710	0.4950	0.9455	0.3765	1.9485	2

4177 rows × 8 columns

In [94]:

```
y=data.iloc[:,8:9]
y
```

Out[94]:

	Rings
0	15
1	7
2	9
3	10
4	7
...	...
4172	11
4173	10
4174	9
4175	10
4176	12

4177 rows × 1 columns

Spliting Training and Testing Data

In [95]:

```
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=10)
```

In [96]:

```
x_train.shape
```

Out[96]:

```
(3341, 8)
```

In [97]:

```
y_train.shape
```

Out[97]:

```
(3341, 1)
```

Building the model

In [98]:

```
from sklearn import linear_model
```

In [99]:

```
regr = linear_model.LinearRegression()
regr.fit(x_train,y_train)
```

Out[99]:

```
LinearRegression()
```

In [100]:

```
pred=regr.predict(x_test)
```

In [101]:

```
pred
```

Out[101]:

```
array([[ 7.90577325],
       [10.94927208],
       [ 9.90587762],
       [ 6.65525637],
       [ 9.00223145],
       [ 7.04180093],
       [12.5651143 ],
       [13.84757893],
       [ 8.43762198],
       [10.30188958],
       [ 9.15437232],
       [12.45795602],
       [ 9.36745087],
       [ 7.85119455],
       [11.15978466],
       [ 5.95616867],
       [13.8170673 ],
       [ 6.44047461],
```

In [102]:

```
y_test
```

Out[102]:

	Rings
1080	7
1371	10
2591	8
235	9
2143	10
...	...
1058	4
1323	9
784	5
2291	10
725	17

836 rows × 1 columns

In [103]:

```
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

In [104]:

```
# model evaluation
print(
'mean_squared_error : ', mean_squared_error(y_test, pred))
print(
'mean_absolute_error : ', mean_absolute_error(y_test, pred))
from sklearn.metrics import r2_score
score=r2_score(y_test,pred)
score
```

```
mean_squared_error :  4.568012913619505
mean_absolute_error :  1.581377031426174
```

Out[104]:

```
0.5387176640137976
```

In [105]:

```
from sklearn.ensemble import RandomForestRegressor
```

In [106]:

```
regressor = RandomForestRegressor(n_estimators = 100, random_state = 0)
```

In [107]:

```
regressor.fit(x_train,y_train)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: DataConversi
onWarning: A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples,), for example using ravel().
  """Entry point for launching an IPython kernel.
```

Out[107]:

```
RandomForestRegressor(random_state=0)
```

In [108]:

```
pred1=regressor.predict(x_test)
```

In [109]:

```
pred1
```

Out[109]:

```
array([ 8.13, 10.59,  8.54,  6.65,  9.59,  5.89, 11.81, 12.94,  9.38,
        9.35,  9.35, 10.79,  9.11,  7.67,  9.82,  5.44, 12.89,  5.75,
       10.67, 14.51, 10.04, 10.96,  9.12, 10.67,  9.28,  4.66, 10.87,
       10.38, 16.22, 11.79,  8.77, 12.17, 10.54, 16.38, 10.11,  7.16,
       10.43, 12.41, 10.81, 14.22,  4.35,  9.39,  6.72,  9.22,  9.11,
       10.9 , 10.74, 10.74, 13.33, 10.38, 17.01, 11.51,  9.77, 11.02,
        5.76,  9.69,  9.16, 11.62,  4.72, 16.23, 12.29, 11.29, 13.75,
        5.52, 11.8 , 12.51, 11.34,  7.81,  9.35,  9.57,  7.13, 13.95,
        7.62, 11.05,  8.74, 10.7 , 13.79,  9.53, 10.47, 10.78, 13.7 ,
       12.77,  8.61, 11.53, 10.96, 14.82,  3.62,  9.99,  9.04, 10.21,
        9.18,  8.36,  9.1 , 12.39, 11.1 , 14.36, 11.19,  9.22,  7.61,
        6.15,  9.38, 10.23, 10.83, 10.08,  9.76,  8.6 ,  8.64,  8.62,
        5.91,  9.08, 10.93,  7.28,  6.43, 12.16, 11.99, 15.03,  8.95,
        9.56, 10.01, 13.86,  7.54,  4.82, 15.13,  6.91, 10.4 , 10.07,
        7.64, 13.51, 13.67,  9.76,  8.36, 11.26, 12.89, 11.46, 13.23,
        6.96, 11.44, 10.67,  8.56, 13.21, 11.06, 10.33, 11.14, 15.85,
        9.79,  6.75,  9.06, 11.07,  9.94, 10.39,  7.55,  8.38,  6.64,
       11.24,  8.94,  7.96,  6.69,  8.78,  5.66, 11.73,  9.76, 10.52,
        9.69, 13.2 ,  8.68, 11.03,  6.97,  8.27,  6.4 , 11.32,  9.85,
        7.86,  6.23, 11.77,  5.72, 10.65,  5.91,  9.85, 10.54,  9.4 ,
        9.07,  8.95, 13.72,  5.62, 12.81,  9.52, 12.56, 13.9 ,  9.11,
        7.48,  6.52,  8.85,  7.16, 10.08,  7.69, 11.38,  8.54,  9.02,
        9.31,  5.83,  2.88, 10.25,  5.2 ,  8.76, 12.  ,  7.63,  8.91,
        4.26, 12.32, 16.13, 10.19, 10.8 ,  6.59, 11.34,  8.94, 11.25,
        9.05, 10.71,  8.82, 13.33,  8.75,  9.09, 11.95, 13.95, 11.04,
        6.63,  7.36,  9.86, 10.15,  7.62,  8.63,  6.32, 10.19, 11.12,
       11.42, 10.11, 11.93,  7.08, 10.51,  6.67, 10.29,  8.7 ,  6.58,
       11.77, 13.54,  9.02,  7.78,  7.7 ,  9.65,  6.9 ,  9.59, 10.54,
        8.27,  8.69, 15.13,  9.59, 11.69, 10.17,  7.62,  9.09, 11.53,
       10.76, 10.37, 14.25,  8.72, 13.48,  7.37,  9.51,  8.69, 11.12,
        9.8 , 13.22, 13.9 , 10.48, 10.26,  7.18, 18.16, 12.64,  9.95,
        4.23,  6.27,  6.52, 10.92, 10.36,  9.09,  8.3 , 10.32,  7.64,
       13.83, 10.13,  8.82, 10.  ,  9.98, 12.63, 11.01,  9.89, 10.76,
       12.43,  9.76,  8.68,  9.26,  7.95,  8.66, 10.25,  8.58,  9.1 ,
        8.52,  7.02,  6.13, 14.72,  7.51, 10.39,  7.31,  8.73,  9.85,
       10.04,  6.39,  5.35,  7.96, 10.03, 11.3 ,  8.06, 12.59, 10.74,
        9.02, 10.87,  8.6 ,  6.39,  6.04, 10.45,  9.63, 11.45, 15.2 ,
        8.04,  8.51,  8.16,  8.83, 10.13, 11.08, 13.09,  8.64,  8.32,
       12.37, 10.11,  7.41, 10.49, 11.12, 12.46,  8.74, 16.62,  8.72,
        7.69, 10.26,  7.76, 11.35, 12.48, 12.28,  6.9 ,  6.43, 11.04,
       10.8 , 11.52,  8.46, 11.67, 10.6 ,  8.54, 13.63,  9.34,  9.23,
        7.12,  4.38, 11.91, 10.68, 10.28,  6.18,  8.32,  7.65,  8.33,
       12.54,  8.09,  7.56, 12.01,  6.52, 13.36, 10.74,  6.78,  7.08,
        9.73, 14.3 , 12.12, 11.06,  9.9 ,  8.22,  7.04,  8.97,  9.38,
        9.31,  4.85, 13.29, 10.04,  6.55,  8.79,  6.99,  9.58, 17.25,
        9.58,  8.95,  7.21, 10.23, 13.13, 10.35,  6.62,  7.03,  6.98,
       10.63, 11.13, 14.97, 10.29,  7.53,  6.61, 15.42,  5.94, 11.4 ,
       11.46, 10.53,  8.81, 12.28,  8.82, 11.42, 10.07,  8.28,  8.52,
       10.07,  8.88, 10.36, 10.98, 13.55,  9.01,  8.94,  9.51,  7.56,
       13.25, 13.06, 10.24, 10.88,  4.9 ,  9.5 ,  9.32,  7.59,  9.08,
        7.91, 11.74,  7.74,  8.08,  5.7 , 10.4 ,  9.28, 15.5 ,  7.08,
       10.07,  9.08,  9.65, 10.3 , 14.9 ,  7.18,  5.6 , 10.93,  8.48,
       13.85,  7.75,  8.93, 13.51, 16.67,  9.25,  9.23,  8.43,  5.8 ,
        9.54, 11.01, 10.34, 11.99,  9.39, 10.04,  7.46,  9.63, 13.73,
        8.79, 10.1 ,  9.24, 15.31,  9.28, 10.  ,  8.24,  8.68,  9.19,
```

```
       11.26, 10.41,  9.93,  9.35, 10.87,  9.88, 11.72, 11.35,  6.98,
        8.61,  9.86,  7.93,  7.49, 16.51,  6.  ,  8.59, 11.21,  8.94,
        7.75,  5.64, 10.53, 10.86, 15.19,  3.82, 10.53,  6.99,  6.17,
        6.8 , 10.69,  4.2 ,  8.81,  9.63, 10.88,  8.57,  7.85,  9.05,
       10.54,  9.86,  6.54, 10.66, 10.37,  6.19,  8.42, 12.05,  9.25,
        8.37, 10.47, 17.07, 10.41, 11.24, 11.87, 10.03,  5.55, 12.9 ,
       10.29,  9.14, 12.45,  9.72,  7.15, 15.77, 11.99, 12.5 , 11.75,
       14.93,  9.76, 10.57,  9.93, 10.33, 10.5 , 10.2 , 14.64, 13.04,
       10.46,  7.62, 11.15, 12.  ,  9.02, 12.4 , 11.52, 10.64, 12.6 ,
       10.35,  8.39, 11.31, 11.1 , 13.27,  8.81,  7.64, 10.35, 10.6 ,
        8.  ,  6.39, 11.53,  9.15, 14.46, 10.85,  6.2 , 10.91,  8.93,
        9.97, 10.82, 15.14,  8.44,  4.28, 10.3 , 10.05, 12.07, 12.72,
        7.11, 10.24, 10.05,  8.88,  4.66, 10.17,  3.84,  8.59, 12.45,
        8.79, 10.8 , 11.04, 13.06, 11.05,  9.93, 16.32, 11.49,  9.65,
        8.92,  9.95, 12.15, 16.62, 10.27,  9.38, 14.46,  9.08, 10.3 ,
        9.22, 10.7 ,  9.26,  8.54,  8.89, 11.39,  7.8 , 11.64, 11.25,
        9.87, 16.07,  7.63,  8.94,  9.27,  7.54,  7.99, 11.1 , 15.89,
       11.58,  9.59,  8.33, 10.03,  9.88, 11.41,  7.51,  8.32,  8.95,
        2.88,  8.58,  7.75, 12.97,  6.92, 13.51, 12.98, 10.02, 10.35,
       11.79,  7.55, 12.72,  9.56, 15.97,  8.41,  6.49, 10.45,  8.06,
       12.88, 10.69,  3.72, 10.59,  9.62,  9.98, 11.48, 11.52,  9.78,
       10.37, 13.83, 10.25, 10.52, 12.33, 10.09, 13.03,  9.21, 13.42,
        4.36, 11.03,  4.29, 11.39, 13.73,  9.28, 14.97,  6.61,  7.62,
        7.74,  5.69,  6.47, 13.34, 16.57, 12.25, 12.44,  9.96, 10.46,
        9.03,  8.43, 10.67,  7.35,  7.66,  7.07,  9.12,  6.69,  7.81,
       10.33, 10.41, 12.05, 10.86,  9.6 , 11.21, 11.28,  6.42, 15.04,
        7.04, 15.38, 10.48, 12.6 , 12.69,  6.67,  8.76, 11.41,  6.68,
        8.77, 10.19,  9.24,  8.78,  8.79,  9.78, 10.57,  9.77,  8.74,
        9.73, 16.83, 10.27, 11.23,  8.35,  6.05,  9.4 ,  7.53,  8.68,
       11.3 , 16.25, 11.09,  8.  ,  4.81, 16.18, 10.19,  9.73,  9.92,
        9.39, 10.42,  7.22, 12.16, 10.76, 12.67, 11.61, 15.74,  8.22,
        8.8 , 10.64, 13.25,  8.75, 12.04, 12.32,  9.81,  8.24, 10.83,
       13.29,  4.43,  7.47,  8.7 ,  6.24,  6.76,  6.74,  9.13,  9.33,
        9.53,  9.46, 10.28,  5.93,  5.26, 13.14, 10.05,  6.52,  9.06,
        8.74,  5.9 , 11.77, 10.45, 12.34,  8.53,  7.01,  6.88,  7.9 ,
       10.27,  5.56, 11.11,  9.62,  9.95,  7.87,  5.68,  8.48, 10.43,
        8.79,  6.15,  9.36,  6.34,  8.69, 14.12, 10.58,  5.97, 10.47,
        8.67,  8.65, 15.12,  4.48,  9.08,  4.44,  9.18, 13.2 ])
```

In [110]:

```
y_test
```

Out[110]:

	Rings
1080	7
1371	10
2591	8
235	9
2143	10
...	...
1058	4
1323	9
784	5
2291	10
725	17

836 rows × 1 columns

In [111]:

```
print(
'mean_squared_error : ', mean_squared_error(y_test, pred1))
```

```
mean_squared_error :  4.340792105263159
```

In [112]:

```
print(np.sqrt(mean_squared_error(y_test,pred1)))
```

```
2.0834567682731406
```

In [113]:

```
from sklearn.metrics import r2_score
score=r2_score(y_test,pred1)
score
```

Out[113]:

```
0.5616626397056985
```

User input value for random Forset

In [117]:

```
import math
```

In [120]:

```
pred_val=regressor.predict([[0.365,0.095,0.455,0.65,0.2245,0.1010,0.5140,0]])
print("The Predicted Number of Rings: ",*pred_val)
print("The Predicted age of the Abalone: ",math.floor(*pred_val+1.5))
```

```
The Predicted Number of Rings:  18.24
The Predicted age of the Abalone:  19

/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X d
oes not have valid feature names, but RandomForestRegressor was fitted with
feature names
  "X does not have valid feature names, but"
```