In [1]:
```
import pandas as pd
df=pd.read_csv('abalone.csv')
print (df)
```

```
      Sex  Length  Diameter  Height  Whole weight  Shucked weight  \
0       M   0.455     0.365   0.095        0.5140          0.2245
1       M   0.350     0.265   0.090        0.2255          0.0995
2       F   0.530     0.420   0.135        0.6770          0.2565
3       M   0.440     0.365   0.125        0.5160          0.2155
4       I   0.330     0.255   0.080        0.2050          0.0895
...    ..     ...       ...     ...           ...             ...
4172    F   0.565     0.450   0.165        0.8870          0.3700
4173    M   0.590     0.440   0.135        0.9660          0.4390
4174    M   0.600     0.475   0.205        1.1760          0.5255
4175    F   0.625     0.485   0.150        1.0945          0.5310
4176    M   0.710     0.555   0.195        1.9485          0.9455

      Viscera weight  Shell weight  Rings
0             0.1010        0.1500     15
1             0.0485        0.0700      7
2             0.1415        0.2100      9
3             0.1140        0.1550     10
4             0.0395        0.0550      7
...              ...           ...    ...
4172          0.2390        0.2490     11
4173          0.2145        0.2605     10
4174          0.2875        0.3080      9
4175          0.2610        0.2960     10
4176          0.3765        0.4950     12

[4177 rows x 9 columns]
```

In [2]:
```
df.shape
```
Out[2]: (4177, 9)

In [3]:
```
df.head()
```
Out[3]:

	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

In [4]:
```
import numpy as np
df.isnull().sum()
```
Out[4]:
```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
Rings             0
dtype: int64
```

obsevation-no null value

In [6]:
```
df.describe()
```
Out[6]:

	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

observation= mean median values are different

data is not normally distributed

outliers present

In [7]:
```
df.info()
```

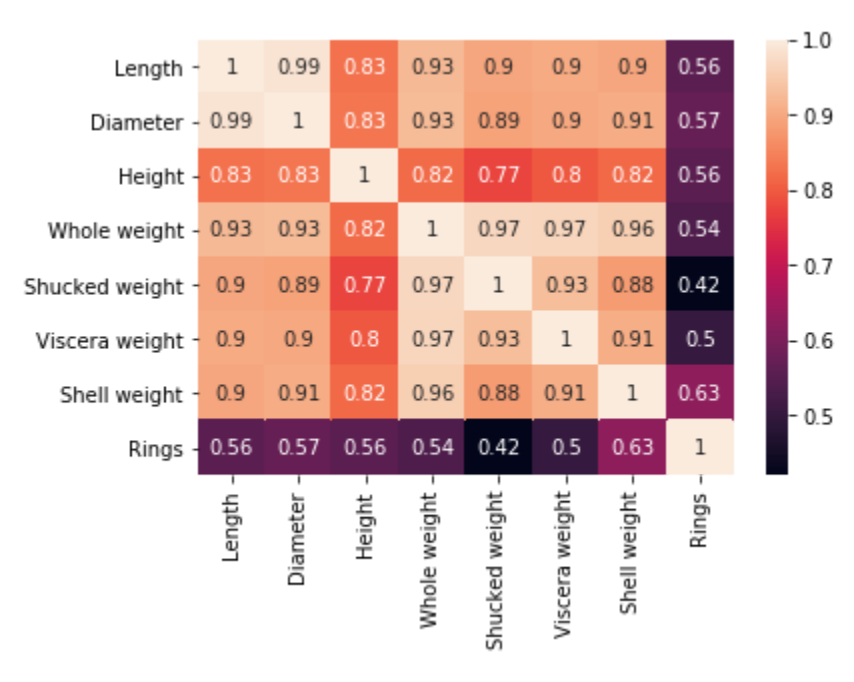
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Sex             4177 non-null   object
 1   Length          4177 non-null   float64
 2   Diameter        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 [8]:
```
dfcor=df.corr()
dfcor
```
Out[8]:

	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

CORR MAP

In [9]:
```
import numpy as np
import matplotlib as plt
import seaborn as sns
sns.heatmap(df.corr(),annot=True)
```
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x24aba2acb88>



First thing to note here is high correlation in data.

There seems to be high multicollinearity between the predictors. for example correlation between Diameter and Length is extremely high

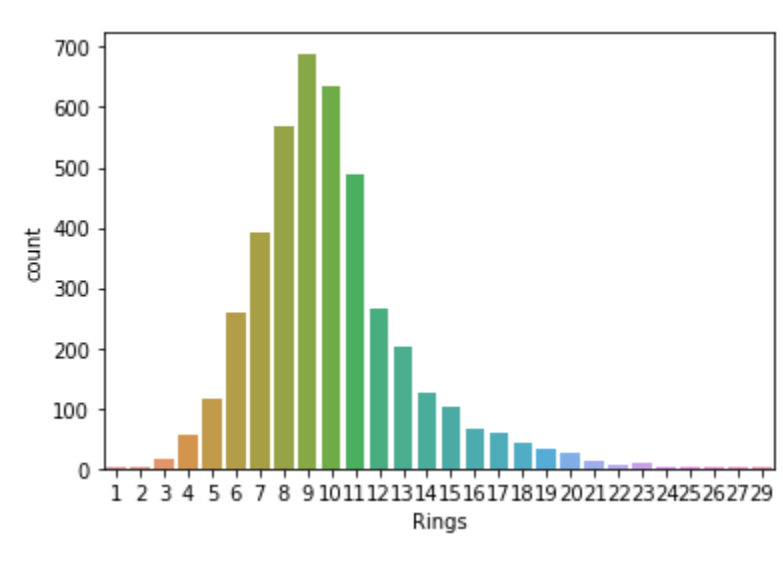
Similarly Whole_weight seems to be highly correlated with other weight predictors

In [12]:
```
df['Rings'].unique()
```
Out[12]:
```
array([15,  7,  9, 10,  8, 20, 16, 19, 14, 11, 12, 18, 13,  5,  4,  6, 21,
       17, 22,  1,  3, 26, 23, 29,  2, 27, 25, 24], dtype=int64)
```

In [13]:
```
df.Rings.value_counts().sort_index()
```
Out[13]:
```
1       1
2       1
3      15
4      57
5     115
6     259
7     391
8     568
9     689
10    634
11    487
12    267
13    203
14    126
15    103
16     67
17     58
18     42
19     32
20     26
21     14
22      6
23      9
24      2
25      1
26      1
27      2
29      1
Name: Rings, dtype: int64
```

COUNTPLOT FOR RINGS

In [14]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(x='Rings',data=df)
```
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x24abc46b708>



In [ ]:

In [15]:
```
import pandas as pd
df=pd.read_csv('abalone.csv')
from sklearn.preprocessing import LabelEncoder
LE=LabelEncoder()
df["Sex"]=LE.fit_transform(df["Sex"])
```

In [16]:
```
df["Sex"].value_counts()
```
Out[16]:
```
2    1528
1    1342
0    1307
Name: Sex, dtype: int64
```

In [17]:
```
df.head()
```
Out[17]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	2	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	2	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	2	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	1	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

REMOVING THE OUTLIERS

In [18]:
```
import numpy as np
from scipy.stats import zscore
z=np.abs(zscore(df))
z
```
Out[18]:
```
array([[1.15198011, 0.57455813, 0.43214879, ..., 0.72621157, 0.63821689,
        1.57154357],
       [1.15198011, 1.44898585, 1.439929  , ..., 1.20522124, 1.21298732,
        0.91001299],
       [1.28068972, 0.05003309, 0.12213032, ..., 0.35668983, 0.20713907,
        0.28962385],
       ...,
       [1.15198011, 0.6329849 , 0.67640943, ..., 0.97541324, 0.49695471,
        0.28962385],
       [1.28068972, 0.84118198, 0.77718745, ..., 0.73362741, 0.41073914,
        0.02057072],
       [1.15198011, 1.54905203, 1.48263359, ..., 1.78744868, 1.84048058,
        0.64095986]])
```

threshold=3 print(np.where(z>3))

In [20]:
```
df_new=df[(z<3).all(axis=1)]
```

In [21]:
```
df_new
```
Out[21]:

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

4027 rows × 9 columns

SPLITTING THE DATA

In [22]:
```
x=df_new.iloc[:,0:-1]
x.head()
```
Out[22]:

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

In [23]:
```
y=df_new.iloc[:,-1]
y.head()
```
Out[23]:
```
0    15
1     7
2     9
3    10
4     7
Name: Rings, dtype: int64
```

In [24]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33, random_state=42)
```

In [25]:
```
x_train.shape
```
Out[25]: (2698, 8)

In [26]:
```
y_train.shape
```
Out[26]: (2698,)

In [27]:
```
x_test.shape
```
Out[27]: (1329, 8)

In [28]:
```
y_test.shape
```
Out[28]: (1329,)

BUILDING MODEL

In [33]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(x_train)
x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)
```

In [35]:
```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(x_train, y_train)
```
Out[35]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

In [36]:
```
y_pred = classifier.predict(x_test)
```

In [ ]:

In [38]:
```
from sklearn.metrics import classification_report, confusion_matrix
print (classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           3       0.00      0.00      0.00         3
           4       0.43      0.32      0.36        19
           5       0.20      0.35      0.25        26
           6       0.22      0.29      0.25        76
           7       0.21      0.27      0.24       115
           8       0.26      0.30      0.28       191
           9       0.19      0.26      0.22       218
          10       0.18      0.20      0.19       212
          11       0.19      0.18      0.19       158
          12       0.11      0.05      0.07       102
          13       0.00      0.00      0.00        65
          14       0.20      0.04      0.07        45
          15       0.08      0.03      0.04        36
          16       0.00      0.00      0.00        22
          17       0.00      0.00      0.00        16
          18       0.00      0.00      0.00         8
          19       0.00      0.00      0.00        17

    accuracy                           0.20      1329
   macro avg       0.13      0.13      0.13      1329
weighted avg       0.18      0.20      0.18      1329

[[ 0  2  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  6  9  4  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 0  5  9  8  3  0  1  0  0  0  0  0  0  0  0  0  0]
 [ 0  1 18 22 30  4  1  0  0  0  0  0  0  0  0  0  0]
 [ 0  0  2 36 31 26 13  3  3  1  0  0  0  0  0  0  0]
 [ 0  0  4 11 43 58 54 10  7  3  1  0  0  0  0  0  0]
 [ 0  0  2  8 17 48 56 53 26  4  3  0  1  0  0  0  0]
 [ 0  0  1  6 10 29 69 43 36 10  3  0  3  1  1  0  0]
 [ 0  0  1  2  6 16 48 42 29  8  3  1  1  0  1  0  0]
 [ 0  0  0  1  1 18 16 30 20  5  3  3  2  1  1  0  1]
 [ 0  0  0  2  2  9 14 18  6  8  0  3  1  0  1  0  1]
 [ 0  0  0  0  1  6  6 14 10  1  3  2  1  0  0  1  0]
 [ 0  0  0  0  0  4 11  7  4  2  5  1  1  1  0  0  0]
 [ 0  0  0  1  2  3  6  3  5  0  1  0  1  0  0  0  0]
 [ 0  0  0  0  0  1  2  6  3  2  2  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  2  4  2  0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  4  0  3  4  2  2  0  1  0  1  0  0]]
```

```
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1272: UndefinedM
etricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no pr
edicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [ ]: