In [1]:
```python
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]:
```python
df.shape
```
Out[2]: (4177, 9)

In [3]:
```python
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]:
```python
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
```

In [5]:
```python
df.describe()
```
Out[5]:

	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

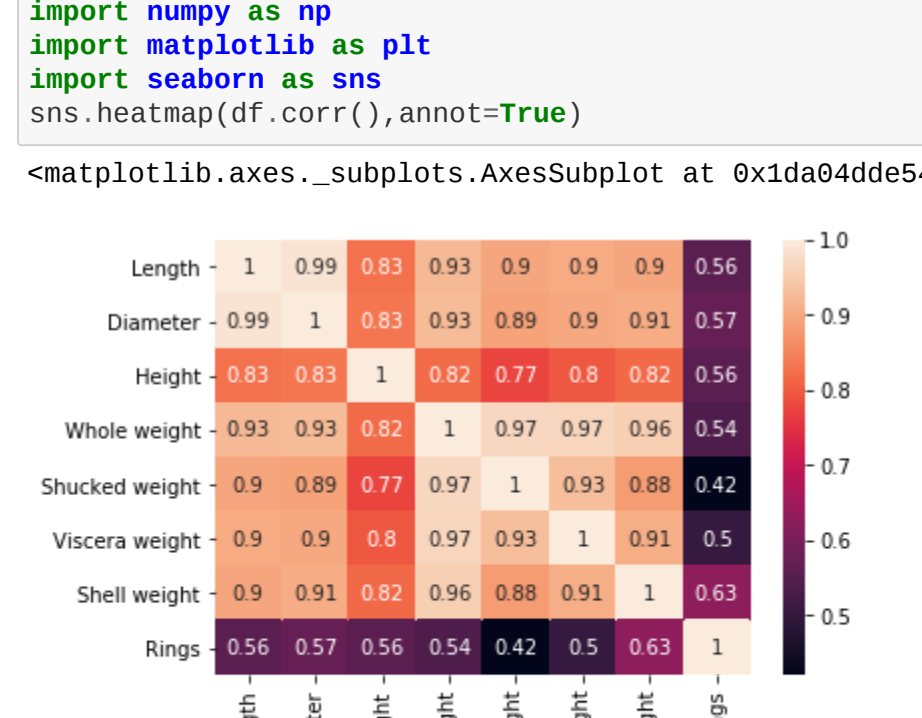
In [6]:
```python
df.info()
```

```
<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 [7]:
```python
dfcor=df.corr()
dfcor
```
Out[7]:

	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

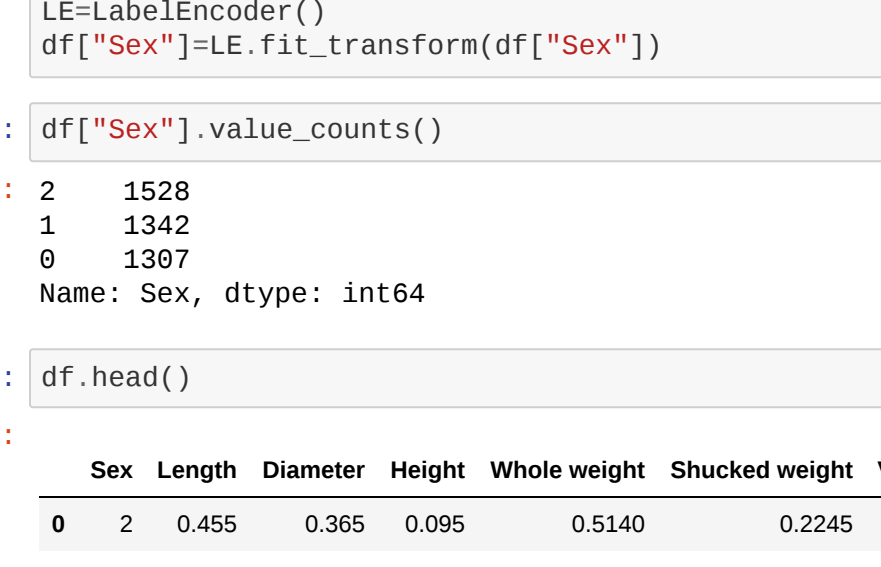
In [8]:
```python
import numpy as np
import matplotlib as plt
import seaborn as sns
sns.heatmap(df.corr(),annot=True)
```
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x1da04dde548>



In [9]:
```python
df['Rings'].unique()
```
Out[9]:
```
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 [10]:
```python
df.Rings.value_counts().sort_index()
```
Out[10]:
```
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
```

In [11]:
```python
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(x='Rings',data=df)
```
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x1da05671e88>



In [ ]:

In [3]:
```python
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 [4]:
```python
df["Sex"].value_counts()
```
Out[4]:
```
2    1528
1    1342
0    1307
Name: Sex, dtype: int64
```

In [5]:
```python
df.head()
```
Out[5]:

	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

In [6]:
```python
#calculate and round off correlation matrix
corMat = df(abalone.iloc[:,:8].corr()).values
corMat = np.around(corMat, decimals = 3)
#print correlation with 'Rings' feature
feature_importance = df(abalone.iloc[:,:8].corr()).iloc[:-1, -1].sort_values(ascending=False
)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-6-6b646a1c4c1f> in <module>
      1 #calculate and round off correlation matrix
----> 2 corMat = df(abalone.iloc[:,:8].corr()).values
      3 corMat = np.around(corMat, decimals = 3)
      4 #print correlation with 'Rings' feature
      5 feature_importance = df(abalone.iloc[:,:8].corr()).iloc[:-1, -1].sort_values(ascendin
g=False)

NameError: name 'abalone' is not defined
```

In [ ]:

In [7]:
```python
import numpy as np
from scipy.stats import zscore
z=np.abs(zscore(df))
z
```
Out[7]:
```
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]])
```

In [8]:
```python
threshold=3
print(np.where(z>3))
```

```
(array([   6,   72,   81,   83,  129,  163,  164,  165,  165,  165,  166,
        166,  167,  168,  170,  181,  232,  236,  236,  236,  237,  237,
        238,  238,  239,  270,  275,  277,  278,  294,  310,  313,  314,
        334,  351,  355,  358,  358,  362,  375,  416,  417,  428,  431,
        433,  469,  478,  480,  501,  526,  526,  530,  572,  628,  673,
        674,  675,  678,  694,  696,  696,  719,  719,  720,  720,  744,
        754,  756,  811,  891,  891,  891, 1048, 1051, 1051, 1052, 1193,
       1197, 1199, 1202, 1204, 1206, 1206, 1207, 1207, 1207, 1207, 1209,
       1209, 1257, 1417, 1417, 1418, 1422, 1422, 1426, 1427, 1427, 1427, 1428,
       1429, 1429, 1528, 1754, 1756, 1757, 1759, 1761, 1762, 1762,
       1763, 1763, 1763, 1821, 1982, 1986, 1987, 1987, 2051, 2090, 2108,
       2108, 2114, 2114, 2157, 2161, 2176, 2180, 2199, 2201, 2208, 2209,
       2274, 2275, 2305, 2334, 2334, 2334, 2335, 2344, 2358, 2368, 2381,
       2436, 2459, 2471, 2544, 2623, 2624, 2624, 2625, 2709, 2710,
       2710, 2810, 2810, 2811, 2811, 2811, 2862, 2863, 2863, 2863, 3007,
       3008, 3140, 3148, 3149, 3149, 3151, 3151, 3188, 3193, 3280, 3359,
       3427, 3427, 3427, 3599, 3599, 3628, 3628, 3713, 3715, 3715, 3715,
       3800, 3837, 3899, 3899, 3902, 3924, 3924, 3930, 3944, 3961, 3962, 3996,
       4145, 4148], dtype=int64), array([8, 8, 7, 8, 7, 7, 7, 4, 5, 7, 7, 8, 8, 7, 6, 8, 8,
1, 2, 3, 1, 2,
       1, 2, 1, 8, 8, 7, 8, 8, 8, 8, 8, 7, 8, 8, 4, 7, 8, 8, 8, 8, 8, 8,
       8, 8, 8, 8, 8, 1, 2, 8, 8, 8, 8, 8, 2, 1, 2, 1, 2, 8,
       8, 8, 8, 4, 5, 7, 6, 4, 5, 4, 5, 5, 5, 5, 6, 5, 6, 4, 5, 6, 7, 4,
       5, 3, 3, 5, 5, 6, 5, 4, 5, 6, 7, 1, 2, 5, 5, 5, 6, 6, 5, 4, 5, 6,
       4, 5, 6, 5, 5, 1, 1, 2, 3, 7, 7, 8, 1, 2, 7, 7, 8, 8, 8, 8, 7, 8,
       7, 8, 8, 4, 6, 8, 8, 8, 5, 4, 4, 5, 6, 5, 6, 8, 1, 8, 8, 8, 5, 4, 4,
       6, 5, 6, 4, 5, 6, 5, 5, 4, 5, 6, 5, 7, 8, 7, 7, 8, 7, 8, 8, 8,
       4, 5, 6, 4, 5, 6, 5, 4, 5, 6, 7, 4, 2, 1, 2, 1, 8, 8, 8, 5, 5, 3,
       7, 6], dtype=int64))
```

In [9]:
```python
df_new=df[(z<3).all(axis=1)]
```

In [10]:
```python
df_new
```
Out[10]:

	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

In [13]:
```python
x=df_new.iloc[:,0:-1]
x.head()
```
Out[13]:

	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 [14]:
```python
y=df_new.iloc[:,-1]
y.head()
```
Out[14]:
```
0    15
1     7
2     9
3    10
4     7
Name: Rings, dtype: int64
```

In [16]:
```python
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 [17]:
```python
x_train.shape
```
Out[17]: (2698, 8)

In [18]:
```python
y_train.shape
```
Out[18]: (2698,)

In [19]:
```python
x_test.shape
```
Out[19]: (1329, 8)

In [20]:
```python
y_test.shape
```
Out[20]: (1329,)

In [21]:
```python
from sklearn.linear_model import LinearRegression
lm=LinearRegression()
lm.fit(x_train,y_train)
```
Out[21]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

In [22]:
```python
pred=lm.predict(x_test)
print("Predicted result price:",pred)
print("actual price",y_test)
```

```
Predicted result price: [ 8.62053035  9.9941614  11.6187725  ...  9.75167392 11.95032521
 11.38476212]
actual price 2514     8
855      9
2495    12
2160    19
1060     7
        ..
2322    12
1812    10
4100     9
180     14
3870    15
Name: Rings, Length: 1329, dtype: int64
```

In [ ]:

In [40]:
```python
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(0, 1))

x_train_scaled = scaler.fit_transform(x_train)
x_train = pd.DataFrame(x_train_scaled)

x_test_scaled = scaler.fit_transform(x_test)
x_test = pd.DataFrame(x_test_scaled)
```

In [42]:
```python
#import required packages
from sklearn import neighbors
from sklearn.metrics import mean_squared_error
from math import sqrt
import matplotlib.pyplot as plt
%matplotlib inline
```

In [46]:
```python
from sklearn.model_selection import GridSearchCV
params = {'n_neighbors':[2,3,4,5,6,7,8,9]}

knn = neighbors.KNeighborsRegressor()

model = GridSearchCV(knn, params, cv=5)
model.fit(x_train,y_train)
model.best_params_
```
Out[46]: {'n_neighbors': 9}

In [38]:
```python
y_pred =  knn.predict(x_test)
```

In [26]:
```python
from sklearn.svm import SVC
svm = SVC(random_state=42, gamma="auto")
svm.fit(x_train, y_train)
print("Test Accuracy: {}%".format(round(svm.score(x_test, y_test)*100, 2)))
```

Test Accuracy: 22.5%

In [28]:
```python
from sklearn.metrics import mean_squared_error
>>> from math import sqrt
>>> train_preds = knn.predict(x_train)
>>> mse = mean_squared_error(y_train, train_preds)
>>> rmse = sqrt(mse)
>>> rmse
```
Out[28]: 2.2902766074151266

In [29]:
```python
test_preds = knn.predict(x_test)
>>> mse = mean_squared_error(y_test, test_preds)
>>> rmse = sqrt(mse)
>>> rmse
```
Out[29]: 2.461440786393407

In [39]:
```python
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
ac = accuracy_score(y_test, y_pred)
```

In [ ]: