ASSIGNMENT DATE	2/11/2022
STUDENT NAME	Durga S
STUDENT ROLL NUMBER	811519104028
MAXIMUM MARKS	2

Description: - Predicting the age of abalone from physical measurements. The age of abalone is

determined by cutting the shell through the cone, staining it, and counting the number of rings through

a microscope -- a boring and time-consuming task. Other measurements, which are easier to obtain,

are used to predict age. Further information, such as weather patterns and location (hence food

availability) may be required to solve the problem.

Building a Regression Model

1. Download the dataset: Dataset

2. Load the dataset into the tool.

3. Perform Below Visualizations.

· Univariate Analysis

· Bi-Variate Analysis

· Multi-Variate Analysis

4. Perform descriptive statistics on the dataset.

5. Check for Missing values and deal with them.

6. Find the outliers and replace them outliers

7. Check for Categorical columns and perform encoding.

8. Split the data into dependent and independent variables.

9. Scale the independent variables

10. Split the data into training and testing
11. Build the Model
12. Train the Model
13. Test the Model
14. Measure the performance using Metrics.

In [28]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [29]:
```
data=pd.read_csv("C:/Users/admin/Downloads/abalone.csv")
```

In [30]:
```
data.head()
```

Out[30]:

	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 [31]:
```
data.describe()
```

Out[31]:

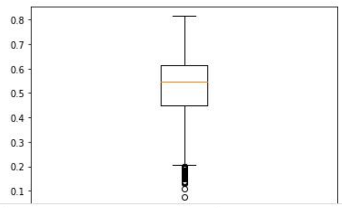
	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0.238831	9.933684
std	0.120093	0.099240	0.041827	0.490389	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 [32]:
```
data.isnull().sum()
```

Out[32]:
```
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]:
```
plt.boxplot(data.Length)
```

Out[5]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x15d9e1a3cd0>,
  <matplotlib.lines.Line2D at 0x15d9e1a3fa0>],
 'caps': [<matplotlib.lines.Line2D at 0x15d9e1be370>,
  <matplotlib.lines.Line2D at 0x15d9e1be700>],
 'boxes': [<matplotlib.lines.Line2D at 0x15d9e1a3880>],
 'medians': [<matplotlib.lines.Line2D at 0x15d9e1bea90>],
 'fliers': [<matplotlib.lines.Line2D at 0x15d9e1bee20>],
 'means': []}
```



In [8]:
```
q3=data.Length.quantile(0.75)
q1=data.Length.quantile(0.25)
iqr=q3-q1
print(iqr)
```

```
0.16499999999999998
```

In [9]:
```
ue=q3+1.5*(iqr)
print(ue)
le=q1-1.5*(iqr)
print(le)
```

```
0.8624999999999999
0.20250000000000004
```

0.20250000000000004

In [10]:
```
data[(data.Length<ue)&(data.Length>le)]
```

Out[10]:

	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

4128 rows × 9 columns

In [11]:
```
data.Length[data.Length>ue]=ue
data.Length[data.Length<le]=le
```

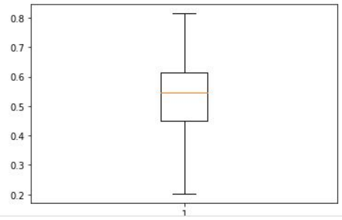
```
C:\Users\admin\AppData\Local\Temp/ipykernel_4524/3962208659.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy
  data.Length[data.Length>ue]=ue
C:\Users\admin\AppData\Local\Temp/ipykernel_4524/3962208659.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ve
rsus-a-copy
  data.Length[data.Length<le]=le
```

In [12]:
```
plt.boxplot(data.Length)
```

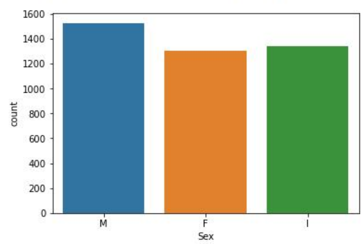
Out[12]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x15d9eb593a0>,
  <matplotlib.lines.Line2D at 0x15d9eb59730>],
 'caps': [<matplotlib.lines.Line2D at 0x15d9eb59ac0>,
  <matplotlib.lines.Line2D at 0x15d9eb59e50>],
 'boxes': [<matplotlib.lines.Line2D at 0x15d9eb4cfd0>],
 'medians': [<matplotlib.lines.Line2D at 0x15d9eb66220>],
 'fliers': [<matplotlib.lines.Line2D at 0x15d9eb665b0>],
 'means': []}
```



In [6]:
```
sns.countplot(data.Sex)
```

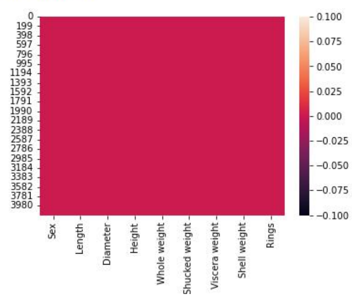
```
C:\Users\admin\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword a
rg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit ke
yword will result in an error or misinterpretation.
  warnings.warn(
```

Out[6]:
```
<AxesSubplot:xlabel='Sex', ylabel='count'>
```



In [13]:
```
sns.heatmap(data.isnull())
```

Out[13]:
```
<AxesSubplot:>
```

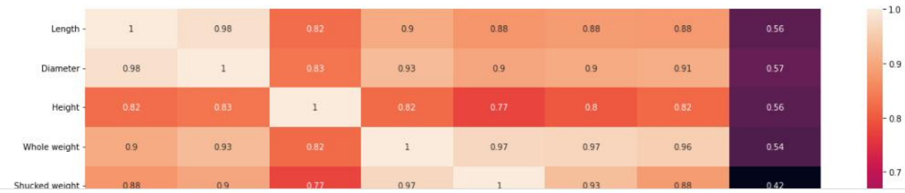


In [24]:
```
nf = data.select_dtypes(include = [np.number]).columns
cf = data.select_dtypes(include = [np.object]).columns
```

```
C:\Users\admin\AppData\Local\Temp/ipykernel_4524/3365015700.py:2: DeprecationWarning: `np.object` is a deprecated alias for the builtin `object`. To silence this warning, use `object` by itself. Doing this will not modify any behavior and is safe.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  cf = data.select_dtypes(include = [np.object]).columns
```

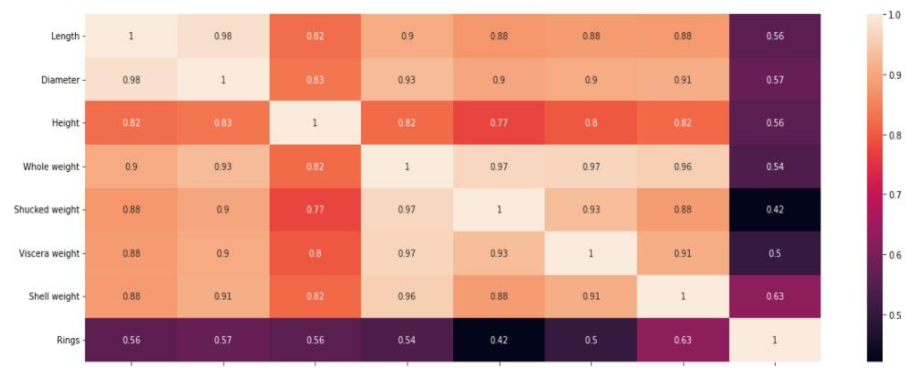
In [25]:
```
plt.figure(figsize = (20,7))
sns.heatmap(data[nf].corr(),annot = True)
```

Out[25]: <AxesSubplot:>



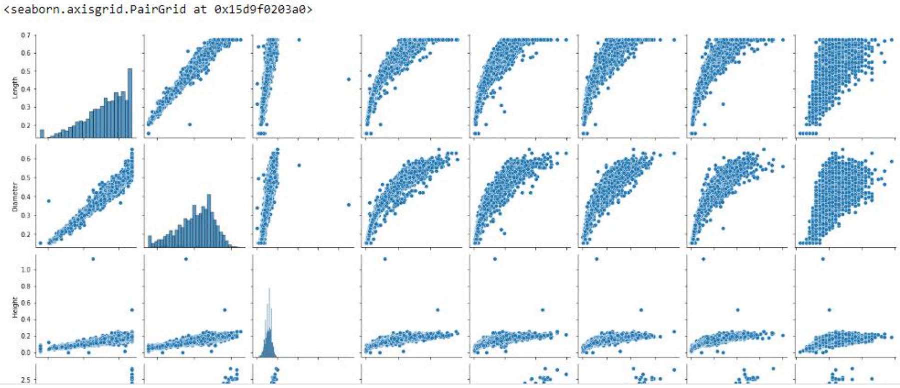
In [25]:
```
plt.figure(figsize = (20,7))
sns.heatmap(data[nf].corr(),annot = True)
```

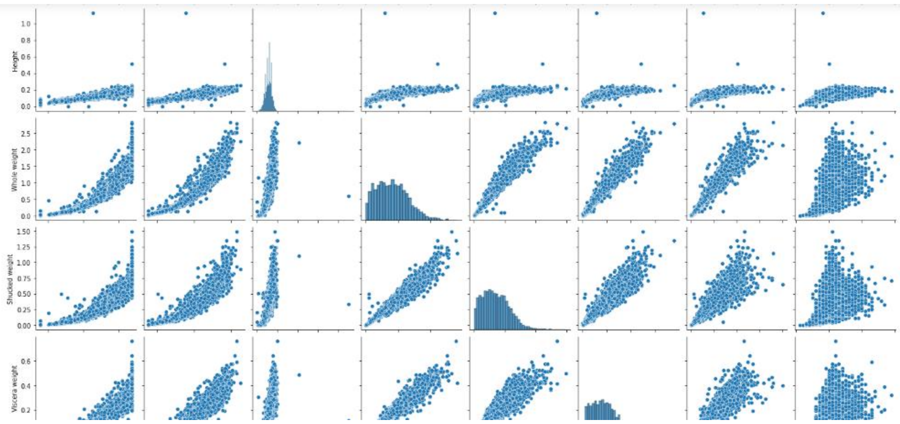
Out[25]: <AxesSubplot:>

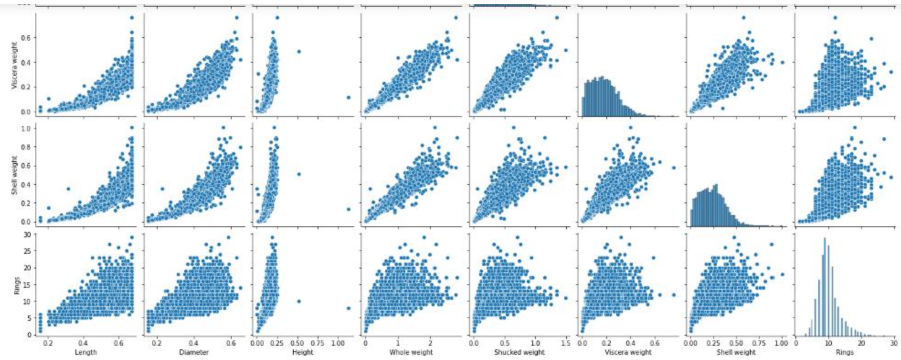


In [26]:
```
sns.pairplot(data)
```

Out[26]: <seaborn.axisgrid.PairGrid at 0x15d9f0203a0>







In [27]:
```python
data.isnull().sum()
```

Out[27]:
```
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 [33]:
```python
data['Height'].describe()
```

Out[33]:
```
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
```

In [34]:
```
data.max()
```

Out[34]:
```
Sex                    M
Length             0.815
Diameter            0.65
Height              1.13
Whole weight      2.8255
Shucked weight     1.488
Viscera weight      0.76
Shell weight       1.005
Rings                 29
dtype: object
```

In [35]:
```
data['Sex'].value_counts()
```

Out[35]:
```
M    1528
I    1342
F    1307
Name: Sex, dtype: int64
```

In [36]:
```
data[data.Height == 0]
```

Out[36]:

	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

In [37]:
```
data['Shucked weight'].kurtosis()
```

Out[37]: 0.5951236783694207

In [38]:
```
data['Diameter'].median()
```

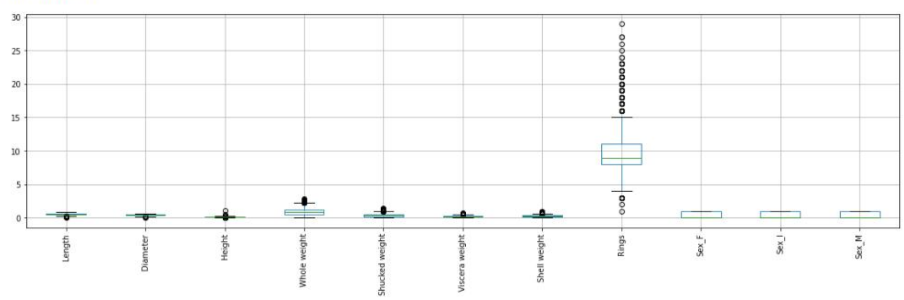
Out[38]: 0.425

In [39]:
```
data['Shucked weight'].skew()
```

Out[39]: 0.7190979217612694

In [43]:
```
data = pd.get_dummies(data)
dummy = data
data.boxplot( rot = 90, figsize=(20,5))
```

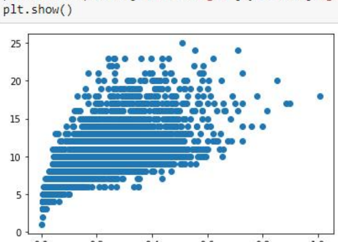
Out[43]: <AxesSubplot:>



In [50]:
```
data['age'] = data['Rings']
data = data.drop('Rings', axis = 1)
```

In [51]:
```
data.drop(data[(data['Viscera weight']> 0.5) & (data['age'] < 20)].index, inplace=True)
data.drop(data[(data['Viscera weight']<0.5) & (data['age'] > 25)].index, inplace=True)
```

In [52]:
```
plt.scatter(x=data['Shell weight'],y = data['age'])
plt.show()
```



In [54]:
```
numerical_features = data.select_dtypes(include = [np.number]).columns
categorical_features = data.select_dtypes(include = [np.object]).columns
```

```
C:\Users\admin\AppData\Local\Temp/ipykernel_4524/3364968343.py:2: DeprecationWarning: `np.object` is a deprecated alias for the builtin `object`. To silence this warning, use `object` by itself. Doing this will not modify any behavior and is safe.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  categorical_features = data.select_dtypes(include = [np.object]).columns
```

In [55]:
```
numerical_features
categorical_features
```

Out[55]:
```
Index([], dtype='object')
```

In [57]:
```
abalone_numeric = data[['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight','Viscera weight', 'Shell weight', 'age',
```

In [58]:
```
abalone_numeric.head()
```

Out[58]:

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

In [59]:
```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
model=scaler.fit(data)
scaled_data=model.transform(data)
print(scaled_data)
```

```
[[0.51351351 0.5210084  0.0840708  ... 0.         1.         0.58333333]
 [0.37162162 0.35294118 0.07964602 ... 0.         1.         0.25      ]
 [0.61486486 0.61344538 0.11946903 ... 0.         0.         0.33333333]
 ...
 [0.70945946 0.70588235 0.18141593 ... 0.         1.         0.33333333]
 [0.74324324 0.72268908 0.13274336 ... 0.         0.         0.375     ]
 [0.85810811 0.84033613 0.17256637 ... 0.         1.         0.45833333]]
```


In [59]:
```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
model=scaler.fit(data)
scaled_data=model.transform(data)
print(scaled_data)
```

```
[[0.51351351 0.5210084  0.0840708  ... 0.         1.         0.58333333]
 [0.37162162 0.35294118 0.07964602 ... 0.         1.         0.25      ]
 [0.61486486 0.61344538 0.11946903 ... 0.         0.         0.33333333]
 ...
 [0.70945946 0.70588235 0.18141593 ... 0.         1.         0.33333333]
 [0.74324324 0.72268908 0.13274336 ... 0.         0.         0.375     ]
 [0.85810811 0.84033613 0.17256637 ... 0.         1.         0.45833333]]
```


In [60]:
```
x = data.drop('age', axis = 1)
y = data['age']
```

```
In [62]: from sklearn.model_selection import train_test_split
         train_x,test_x,train_y,test_y=train_test_split(x,y,test_size=0.2,random_state=25)

In [63]: from sklearn.linear_model import  LinearRegression
         model=LinearRegression()

In [64]: model.fit(train_x,train_y)

Out[64]: LinearRegression()

In [65]: model.predict(test_x)

Out[65]: array([ 9.94301644, 12.46888676,  6.53380794,  4.46644662,  9.63517262,
                 8.74456903,  9.09747127, 13.29464626,  9.38111887,  9.21741905,
                 5.665421  ,  7.99255495,  9.87593558,  9.63938237,  9.796547  ,
                10.63946681,  9.74755474, 11.71758458,  8.4292209 ,  9.85388942,
                 7.60136481, 11.03319821, 12.66507999, 11.94267943,  9.21941446,
                10.74556065, 10.08296121,  8.64506228, 12.26465815,  8.08701057,
                 8.76986532,  9.80191058, 11.21690527, 12.31962844,  6.63454747,
                 7.76173894, 13.15952026, 11.20386834, 11.19037116, 11.01660578,
                 8.94430032,  8.48179814,  8.27284612, 10.03875755,  9.81437691,
                11.44122216,  8.10659319, 13.41844322,  9.13833438,  7.16226984,
                10.66778019, 12.06996671,  9.64265261, 11.44479356,  9.10262088,
                 9.07829606,  7.7929642 , 10.90121456,  8.95586527,  5.66173825,
                 7.70229579,  9.67405012,  7.85663108, 12.02022154, 10.42759406,
```

```
In [66]: predicted_y=model.predict(test_x)

In [67]: model.score(test_x,test_y)

Out[67]: 0.4907507202674961

In [68]: from sklearn.metrics import mean_squared_error

In [69]: mean_squared_error(test_y,predicted_y)

Out[69]: 4.72907775645871
```