Import the libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [2]:
```
import os
os.chdir("C:/Users/vijay/OneDrive/Desktop/Data Sets")
```

In [3]:
```
abalone = pd.read_csv('abalone.csv')
```

In [4]:
```
abalone
```

Out[4]:

	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 [5]:
```
abalone.head()
```

Out[5]:

	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 [6]:
```
abalone.tail()
```

Out[6]:

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

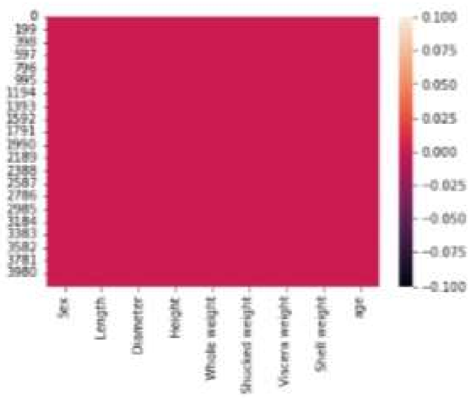
In [ ]:
```
#age can be calculated by using adding value 1.5 to Rings
```

In [7]:
```
abalone['age'] = abalone['Rings']+1.5
abalone = abalone.drop('Rings', axis = 1)
```

Univariate Analysis

In [8]:
```
sns.heatmap(abalone.isnull())
```

Out[8]: <AxesSubplot:>

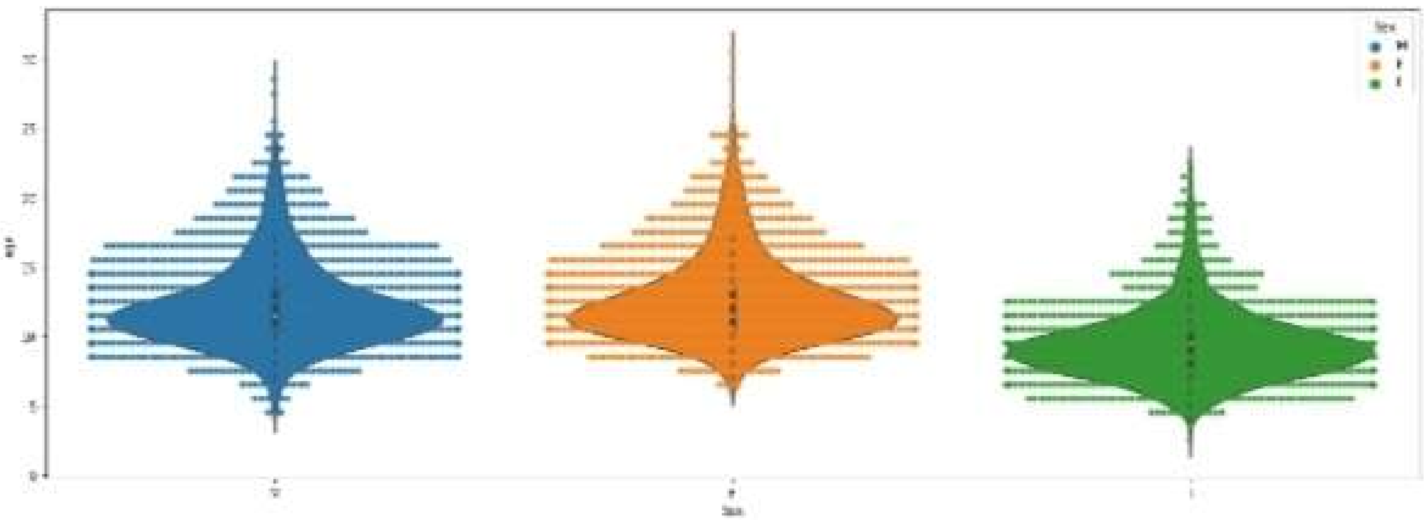


In [10]:

```
plt.figure(figsize = (20,7))
sns.swarmplot(x = 'Sex', y = 'age', data = abalone, hue = 'Sex')
sns.violinplot(x = 'Sex', y = 'age',data = abalone)
```

```
C:\Users\DELL\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 56.2% of the poin
ts cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\DELL\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 52.2% of the poin
ts cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
C:\Users\DELL\anaconda3\lib\site-packages\seaborn\categorical.py:1296: UserWarning: 58.5% of the poin
ts cannot be placed; you may want to decrease the size of the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```

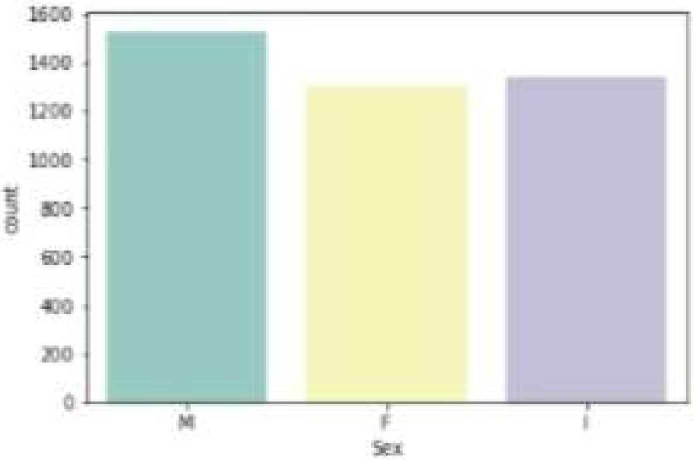
Out[10]: <AxesSubplot:xlabel='Sex', ylabel='age'>



In [9]:

```
sns.countplot(x = 'Sex', data = abalone, palette = 'Set3')
```

Out[9]: <AxesSubplot:xlabel='Sex', ylabel='count'>



Bivariate Analysis

In [10]:

```
numerical_features = abalone.select_dtypes(include = [np.number]).columns
categorical_features = abalone.select_dtypes(include = [np.object]).columns
```

```
C:\Users\vijay\AppData\Local\Temp\ipykernel_24684\1905949472.py:2: DeprecationWarning: `np.object` i
a deprecated alias for the builtin `object`. To silence this warning, use `object` by itself. Doing
his 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-no
es.html#deprecations
  categorical_features = abalone.select_dtypes(include = [np.object]).columns
```

In [11]:

```
numerical_features
```

Out[11]:

```
Index(['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
       'Viscera weight', 'Shell weight', 'age'],
      dtype='object')
```

In [12]:

```
categorical_features
```

Out[12]:

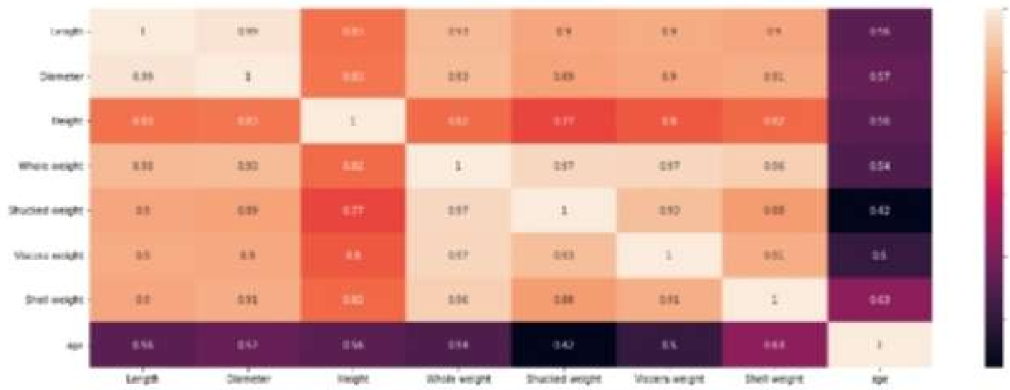
```
Index(['Sex'], dtype='object')
```

In [13]:

```
plt.figure(figsize = (20,7))
sns.heatmap(abalone[numerical_features].corr(),annot = True)
```

Out[13]:

```
<AxesSubplot:>
```



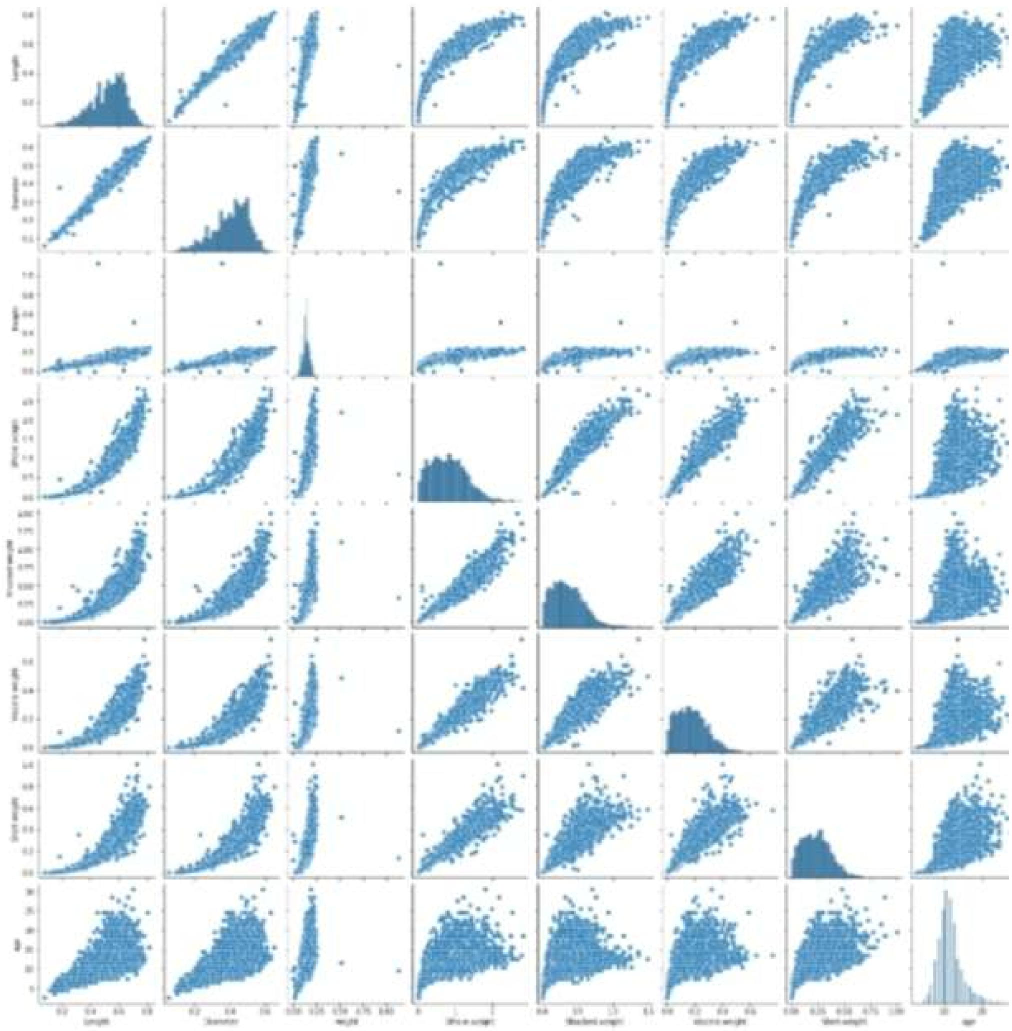
Multivariate Analysis

In [14]:

```
sns.pairplot(abalone)
```

Out[14]:

```
<seaborn.axisgrid.PairGrid at 0x1ed6ef50820>
```



Descriptive Statistics

In [15]:
```
#continuous variables
```

In [16]:
```
abalone['Length'].describe()
```

Out[16]:
```
count    4177.000000
mean        0.523992
std         0.120093
min         0.075000
25%         0.450000
50%         0.545000
75%         0.615000
max         0.815000
Name: Length, dtype: float64
```

In [17]:
```
abalone['Shucked weight'].describe()
```

Out[17]:
```
count    4177.000000
mean        0.359367
std         0.221963
min         0.001000
25%         0.186000
50%         0.336000
75%         0.502000
max         1.488000
Name: Shucked weight, dtype: float64
```

In [18]:
```
abalone['Shell weight'].describe()
```

Out[18]:
```
count    4177.000000
mean        0.238831
std         0.139203
min         0.001500
25%         0.130000
50%         0.234000
75%         0.329000
max         1.005000
Name: Shell weight, dtype: float64
```

In [19]:
```
abalone['Height'].describe()
```

Out[19]:
```
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 [23]:
```
# Categorical variable
```

In [24]:
```
abalone['Sex'].describe()
```

Out[24]:
```
count     4177
unique       3
top          M
freq      1528
Name: Sex, dtype: object
```

In [25]:
```
abalone['Sex'].value_counts()
```

Out[25]:
```
M    1528
I    1342
F    1307
Name: Sex, dtype: int64
```

In [26]:
```
#Distribution measures
```

In [27]:
```
abalone['Length'].kurtosis()
```

Out[27]:
```
0.06462097389494126
```

In [28]:
```
abalone['Length'].skew()
```

Out[28]:
```
-0.639873268981801
```

In [29]:
```
abalone['Shucked weight'].kurtosis()
```

Out[29]:
```
0.5951236783694207
```

In [30]:
```
abalone['Shucked weight'].skew()
```

Out[30]:
```
0.7190979217612694
```

Missing values

n [31]:
```
missing_values = abalone.isnull().sum()
```

n [32]:
```
missing_values
```

ut[32]:
```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
age               0
dtype: int64
```

n [33]:
```
missing_values = abalone.isnull().sum().sort_values(ascending = False)
percentage_missing_values = (missing_values/len(abalone))*100
pd.concat([missing_values, percentage_missing_values], axis = 1, keys= ['Missing values', '% Missing'
```

ut[33]:

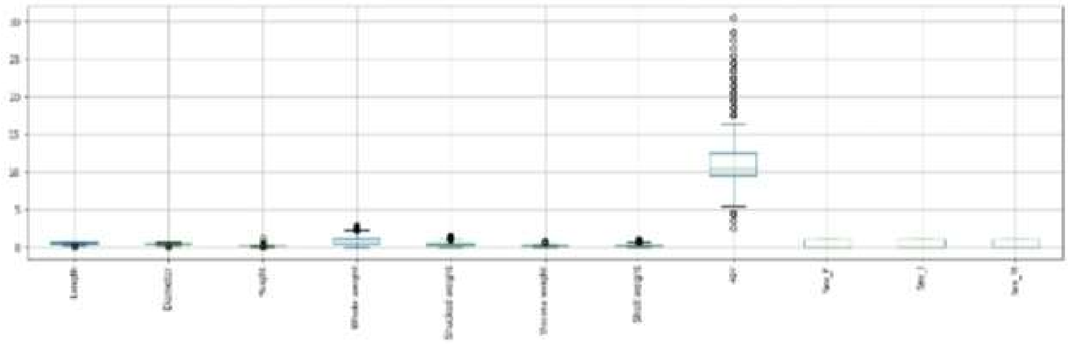
	Missing values	% Missing
Sex	0	0.0
Length	0	0.0
Diameter	0	0.0
Height	0	0.0
Whole weight	0	0.0
Shucked weight	0	0.0
Viscera weight	0	0.0
Shell weight	0	0.0
age	0	0.0

Outliers

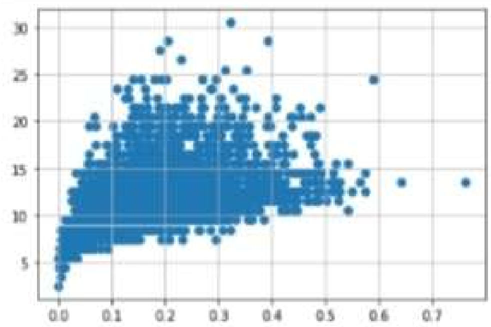
n [34]:
```
abalone = pd.get_dummies(abalone)
dummy_df = abalone
```

n [35]:
```
abalone.boxplot( rot = 90, figsize=(20,5))
```

ut[35]:
```
<AxesSubplot:>
```



n [36]:
```
var = 'Viscera weight'
plt.scatter(x = abalone[var], y = abalone['age'])
plt.grid(True)
```



n [37]:
```
abalone.drop(abalone[(abalone['Viscera weight']> 0.5) & (abalone['age'] < 20)].index, inplace=True)
abalone.drop(abalone[(abalone['Viscera weight']<0.5) & (abalone['age'] > 25)].index, inplace=True)
```

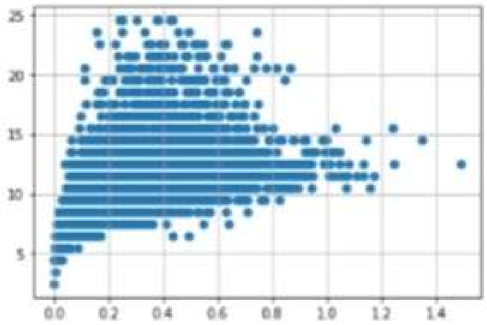
n [38]:
```
var = 'Shell weight'
plt.scatter(x = abalone[var], y = abalone['age'])
plt.grid(True)
```

In [39]:

```
abalone.drop(abalone[(abalone['Shell weight']> 0.6) & (abalone['age'] < 25)].index, inplace=True)
abalone.drop(abalone[(abalone['Shell weight']<0.8) & (abalone['age'] > 25)].index, inplace=True)
```

In [40]:

```
var = 'Shucked weight'
plt.scatter(x = abalone[var], y = abalone['age'])
plt.grid(True)
```

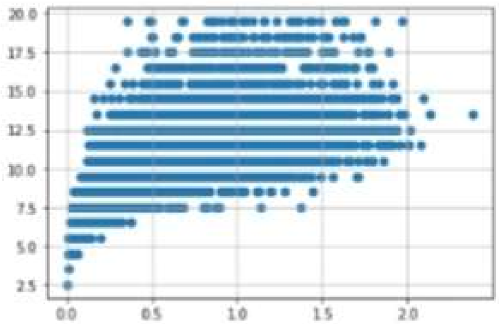


In [41]:

```
abalone.drop(abalone[(abalone['Shucked weight'] >= 1) & (abalone['age'] < 20)].index, inplace = True)
abalone.drop(abalone[(abalone['Viscera weight']<1) & (abalone['age'] > 20)].index, inplace = True)
```

In [42]:

```
var = 'Whole weight'
plt.scatter(x = abalone[var], y = abalone['age'])
plt.grid(True)
```

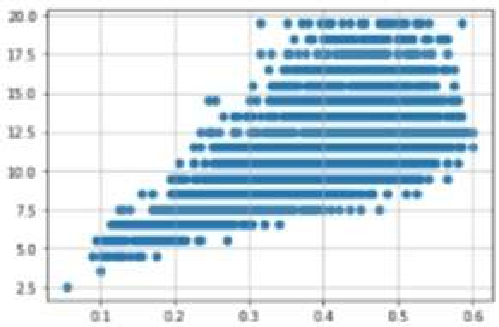


In [43]:

```
abalone.drop(abalone[(abalone['Whole weight'] >= 2.5) & (abalone['age'] < 25)].index, inplace = True)
abalone.drop(abalone[(abalone['Whole weight']<2.5) & (abalone['age'] > 25)].index, inplace = True)
```

In [44]:

```
var = 'Diameter'
plt.scatter(x = abalone[var], y = abalone['age'])
plt.grid(True)
```

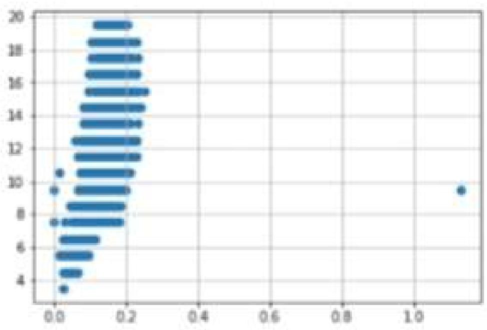


In [45]:

```
abalone.drop(abalone[(abalone['Diameter'] <0.1) & (abalone['age'] < 5)].index, inplace = True)
abalone.drop(abalone[(abalone['Diameter']<0.6) & (abalone['age'] > 25)].index, inplace = True)
abalone.drop(abalone[(abalone['Diameter']>=0.6) & (abalone['age'] < 25)].index, inplace = True)
```

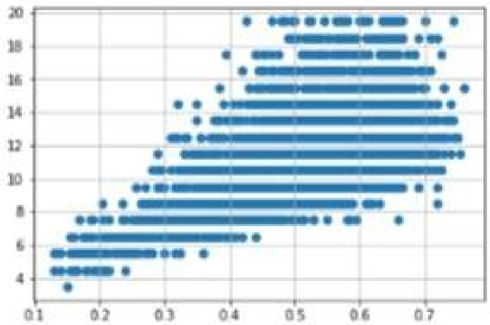
In [47]:

```
var = 'Height'
plt.scatter(x = abalone[var], y = abalone['age'])
plt.grid(True)
```



In [48]:
```
abalone.drop(abalone[(abalone['Height'] > 0.4) & (abalone['age'] < 15)].index, inplace = True)
abalone.drop(abalone[(abalone['Height']<0.4) & (abalone['age'] > 25)].index, inplace = True)
```

In [49]:
```
var = 'Length'
plt.scatter(x = abalone[var], y = abalone['age'])
plt.grid(True)
```



In [50]:
```
abalone.drop(abalone[(abalone['Length'] <0.1) & (abalone['age'] < 5)].index, inplace = True)
abalone.drop(abalone[(abalone['Length']<0.8) & (abalone['age'] > 25)].index, inplace = True)
abalone.drop(abalone[(abalone['Length']>=0.8) & (abalone['age'] < 25)].index, inplace = True)
```

In [51]:
```
abalone
```

Out[51]:

	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	0.1500	16.5	0	0	1
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	8.5	0	0	1
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	10.5	1	0	0
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	11.5	0	0	1
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	8.5	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...
4172	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	12.5	1	0	0
4173	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	11.5	0	0	1
4174	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	10.5	0	0	1
4175	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	11.5	1	0	0
4176	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	13.5	0	0	1

3995 rows × 11 columns

Categorical columns

In [52]:
```
numerical_features = abalone.select_dtypes(include = [np.number]).columns
categorical_features = abalone.select_dtypes(include = [np.object]).columns
```

```
C:\Users\vijay\AppData\Local\Temp\ipykernel_24684\1905949472.py:2: DeprecationWarning: `np.object` is
a deprecated alias for the builtin `object`. To silence this warning, use `object` by itself. Doing t
his 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-not
es.html#deprecations
  categorical_features = abalone.select_dtypes(include = [np.object]).columns
```

In [53]:
```
numerical_features
```

Out[53]:
```
Index(['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
       'Viscera weight', 'Shell weight', 'age', 'Sex_F', 'Sex_I', 'Sex_M'],
      dtype='object')
```

In [54]:
```
categorical_features
```

Out[54]:
```
Index([], dtype='object')
```

In [55]:
```
abalone_numeric = abalone[['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight','Viscera
```

In [56]:
```
abalone_numeric.head()
```

Out[56]:

	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	16.5	0	0	1
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	8.5	0	0	1
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	10.5	1	0	0
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	11.5	0	0	1
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	8.5	0	1	0

Dependent and Independent Variables

In [57]:
```
x = abalone.iloc[:, 0:1].values
```

In [58]:
```
y = abalone.iloc[:, 1]
```

In [59]:
```
x
```

Out[59]:
```
array([[0.455],
       [0.35 ],
       [0.53 ],
       ...,
       [0.6  ],
       [0.625],
       [0.71 ]])
```

In [60]:
```
y
```

Out[60]:
```
0       0.365
1       0.265
2       0.420
3       0.365
4       0.255
        ...
4172    0.450
4173    0.440
4174    0.475
4175    0.485
4176    0.555
Name: Diameter, Length: 3995, dtype: float64
```

Scaling the Independent Variables

In [61]:
```
print ("\n ORIGINAL VALUES: \n\n", x,y)
```

```
 ORIGINAL VALUES:

 [[0.455]
 [0.35 ]
 [0.53 ]
 ...
 [0.6  ]
 [0.625]
 [0.71 ]] 0       0.365
1       0.265
2       0.420
3       0.365
4       0.255
        ...
4172    0.450
4173    0.440
4174    0.475
4175    0.485
4176    0.555
Name: Diameter, Length: 3995, dtype: float64
```

In [62]:
```
from sklearn import preprocessing
min_max_scaler = preprocessing.MinMaxScaler(feature_range =(0, 1))
new_y= min_max_scaler.fit_transform(x,y)
print ("\n VALUES AFTER MIN MAX SCALING: \n\n", new_y)
```

```
 VALUES AFTER MIN MAX SCALING:

 [[0.51587302]
 [0.34920635]
 [0.63492063]
 ...
 [0.74603175]
 [0.78571429]
 [0.92063492]]
```

Split the data into Training and Testing

In [63]:
```
X = abalone.drop('age', axis = 1)
y = abalone['age']
```

In [64]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.feature_selection import SelectKBest
```

In [65]:
```
standardScale = StandardScaler()
standardScale.fit_transform(X)

selectkBest = SelectKBest()
X_new = selectkBest.fit_transform(X, y)

X_train, X_test, y_train, y_test = train_test_split(X_new, y, test_size = 0.25)
```

Build the model

Linear Regression

In [66]:
```
from sklearn import linear_model as lm
from sklearn.linear_model import LinearRegression
model=lm.LinearRegression()
results=model.fit(X_train,y_train)
```

In [67]:
```
accuracy = model.score(X_train, y_train)
print('Accuracy of the model:', accuracy)
```

```
Accuracy of the model: 0.5411538696091844
```

In [68]:
```
lm = LinearRegression()
lm.fit(X_train, y_train)
```

Out[68]:
```
LinearRegression()
```

In [69]:
```
y_train_pred = lm.predict(X_train)
y_test_pred = lm.predict(X_test)
```

Training the model

In [70]:
```
X_train
```

Out[70]:
```
array([[0.48 , 0.365, 0.12 , ..., 0.   , 0.   , 1.   ],
       [0.495, 0.41 , 0.125, ..., 1.   , 0.   , 0.   ],
       [0.455, 0.375, 0.125, ..., 0.   , 1.   , 0.   ],
       ...,
       [0.605, 0.495, 0.17 , ..., 1.   , 0.   , 0.   ],
       [0.23 , 0.18 , 0.05 , ..., 0.   , 1.   , 0.   ],
       [0.565, 0.435, 0.145, ..., 0.   , 1.   , 0.   ]])
```

In [71]:
```
y_train
```

Out[71]:
```
952      8.5
841     10.5
3251     9.5
1773    11.5
1122    10.5
        ...
1135     9.5
1296    10.5
2923    14.5
2547     6.5
1891    10.5
Name: age, Length: 2996, dtype: float64
```

In [72]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error
s = mean_squared_error(y_train, y_train_pred)
print('Mean Squared error of training set :%2f'%s)
```

```
Mean Squared error of training set :3.531472
```

Testing the model

In [73]:
```
X_test
```

Out[73]:
```
array([[0.615, 0.5  , 0.205, ..., 1.   , 0.   , 0.   ],
       [0.35 , 0.26 , 0.09 , ..., 0.   , 1.   , 0.   ],
       [0.495, 0.375, 0.15 , ..., 1.   , 0.   , 0.   ],
       ...,
       [0.56 , 0.45 , 0.16 , ..., 0.   , 0.   , 1.   ],
       [0.575, 0.46 , 0.145, ..., 1.   , 0.   , 0.   ],
       [0.43 , 0.34 , 0.11 , ..., 1.   , 0.   , 0.   ]])
```

In [74]:
```
y_test
```

Out[74]:
```
3300    17.5
3527     8.5
3292    12.5
1516    11.5
3021    10.5
        ...
1287     9.5
2700    14.5
198     16.5
1136     8.5
2376     9.5
Name: age, Length: 999, dtype: float64
```

In [75]:
```
p = mean_squared_error(y_test, y_test_pred)
print('Mean Squared error of testing set :%2f'%p)
```

```
Mean Squared error of testing set :3.657977
```

Testing the model

In [73]:
```
X_test
```

Out[73]:
```
array([[0.615, 0.5  , 0.205, ..., 1.   , 0.   , 0.   ],
       [0.35 , 0.26 , 0.09 , ..., 0.   , 1.   , 0.   ],
       [0.495, 0.375, 0.15 , ..., 1.   , 0.   , 0.   ],
       ...,
       [0.56 , 0.45 , 0.16 , ..., 0.   , 0.   , 1.   ],
       [0.575, 0.46 , 0.145, ..., 1.   , 0.   , 0.   ],
       [0.43 , 0.34 , 0.11 , ..., 1.   , 0.   , 0.   ]])
```

In [74]:
```
y_test
```

Out[74]:
```
3300    17.5
3527     8.5
3292    12.5
1516    11.5
3021    10.5
        ...
1287     9.5
2700    14.5
198     16.5
1136     8.5
2376     9.5
Name: age, Length: 999, dtype: float64
```

In [75]:
```
p = mean_squared_error(y_test, y_test_pred)
print('Mean Squared error of testing set :%2f'%p)
```

```
Mean Squared error of testing set :3.657977
```

In [76]:
```
from sklearn.metrics import r2_score
s = r2_score(y_train, y_train_pred)
print('R2 Score of training set:%.2f'%s)
```

```
R2 Score of training set:0.54
```

In [77]:
```
from sklearn.metrics import r2_score
p = r2_score(y_test, y_test_pred)
print('R2 Score of testing set:%.2f'%p)
```

```
R2 Score of testing set:0.51
```

In [ ]: