In [2]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

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

In [4]:
```
df.describe()
```

Out[4]:

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

Out[5]:

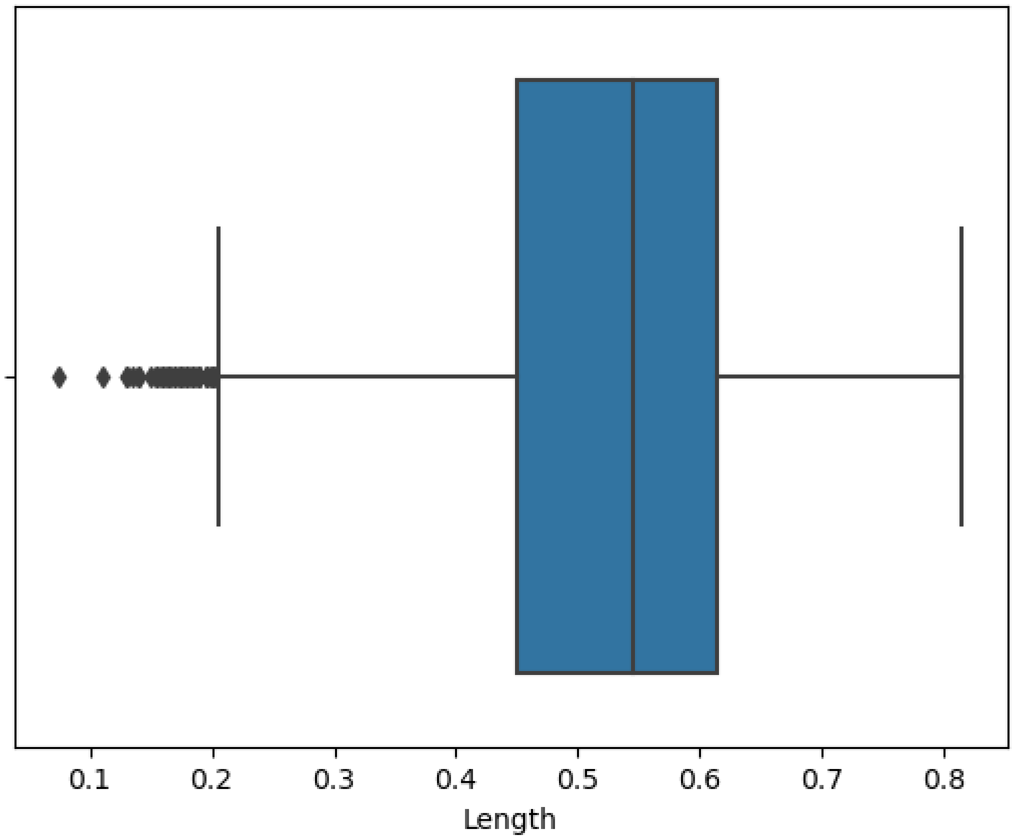
	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330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

In [6]:
```
sns.boxplot(df.Length)
```

```
C:\Users\91904\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[6]:

```
<AxesSubplot:xlabel='Length'>
```

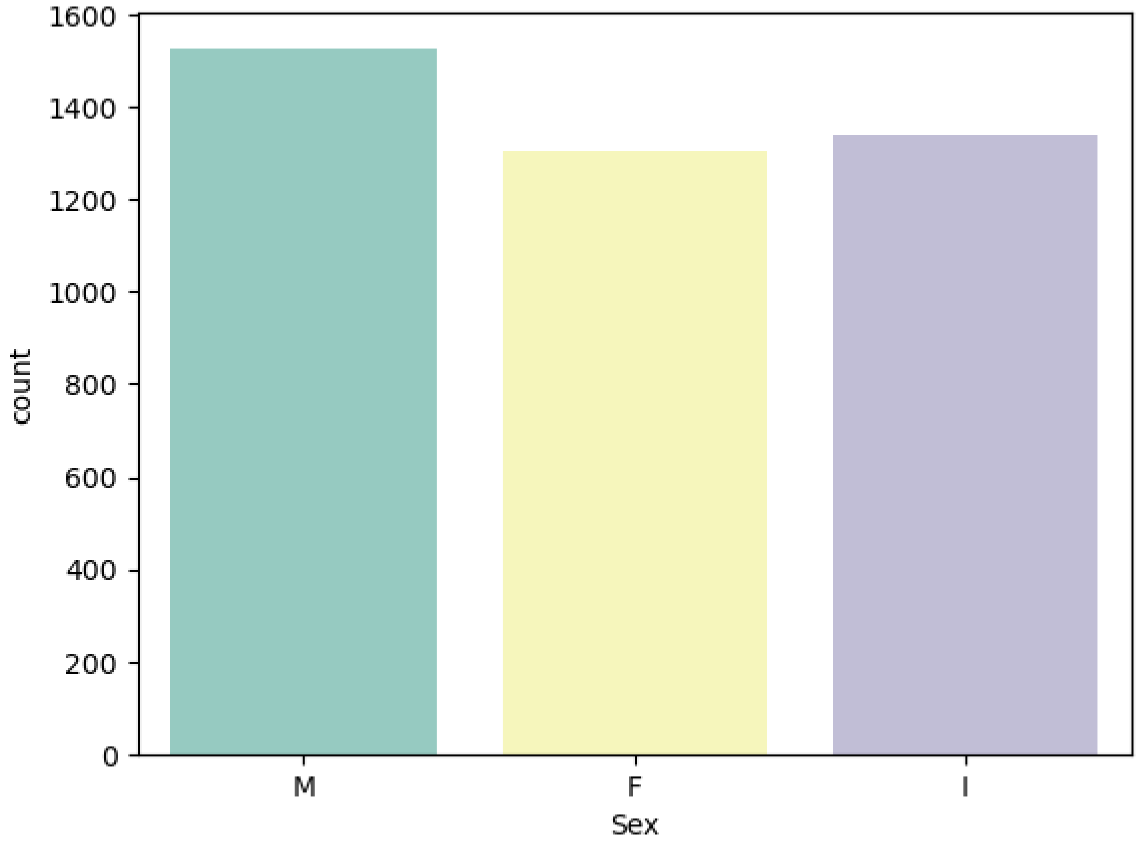


In [7]:

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

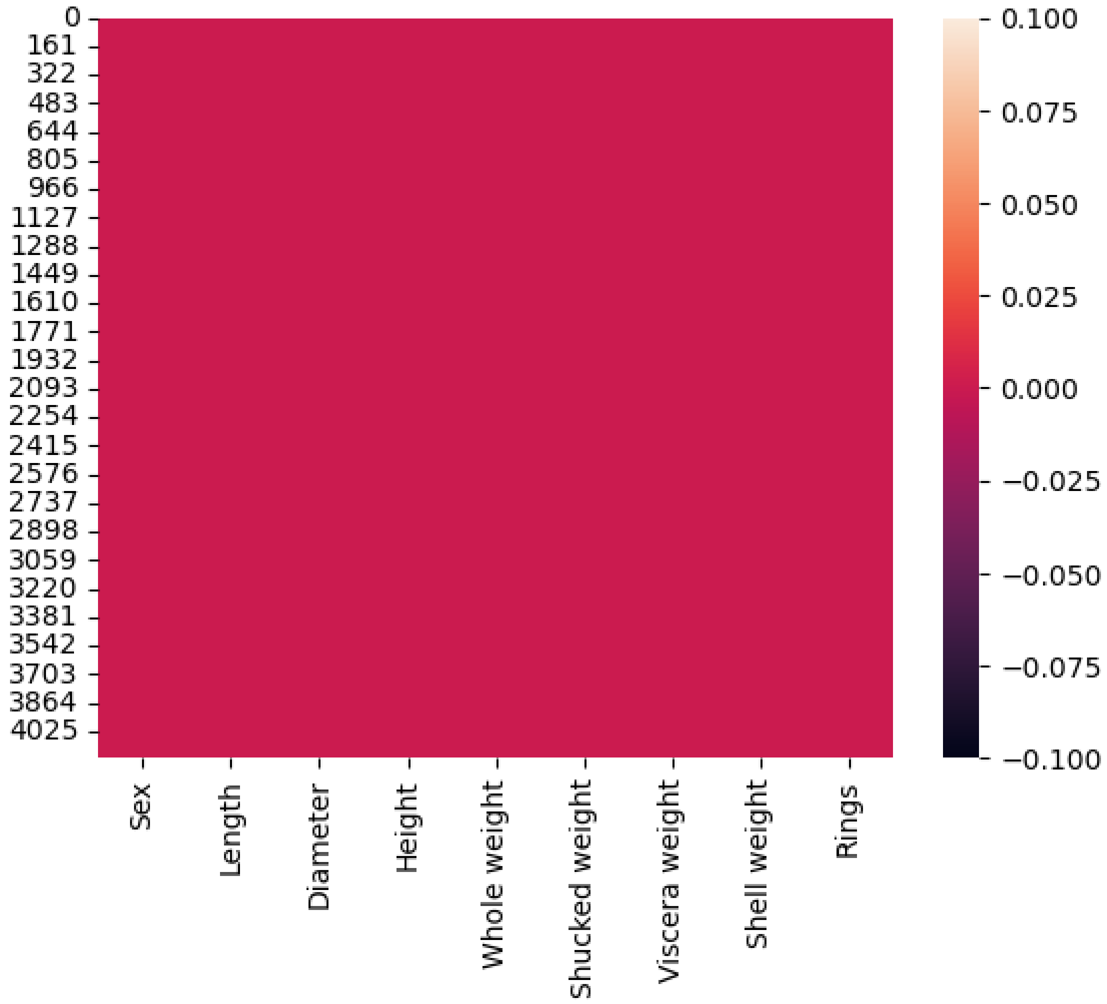
Out[7]:

```
<AxesSubplot:xlabel='Sex', ylabel='count'>
```



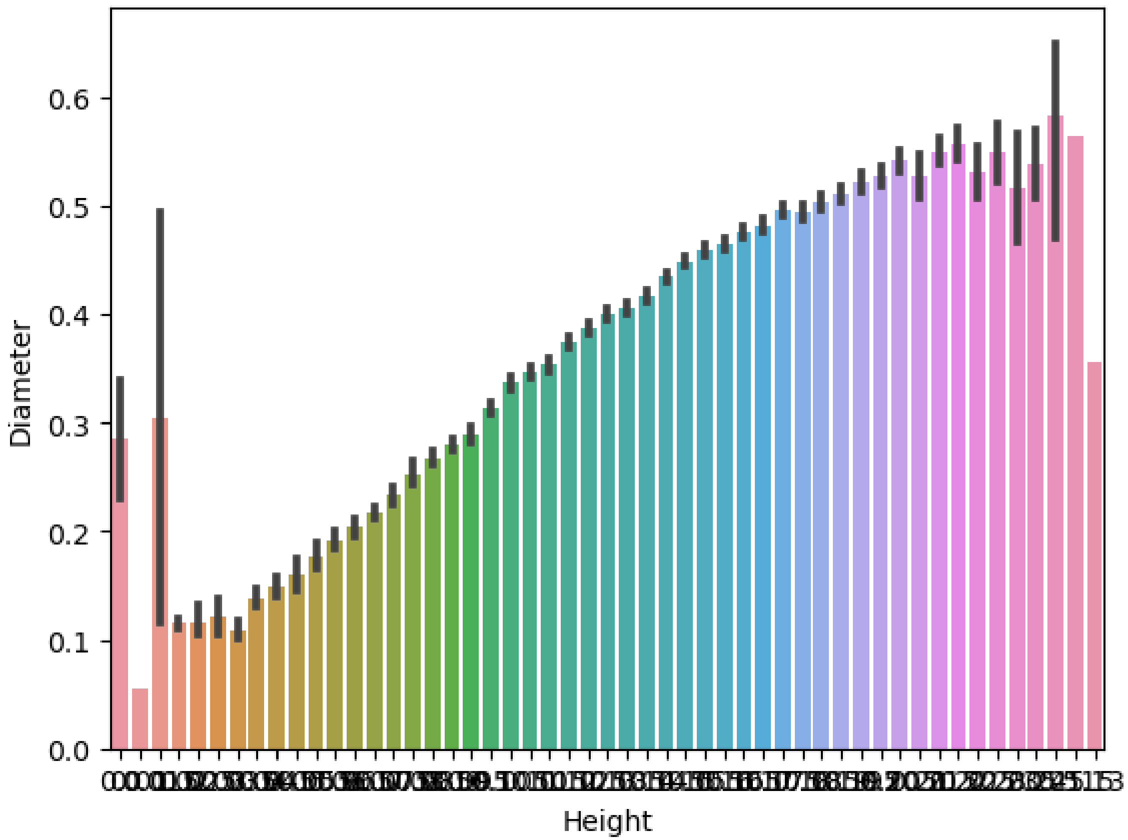
In [8]:
```
sns.heatmap(df.isnull())
```

Out[8]: <AxesSubplot:>



In [9]:
```
sns.barplot(x=df.Height,y=df.Diameter)
```

Out[9]:
```
<AxesSubplot:xlabel='Height', ylabel='Diameter'>
```

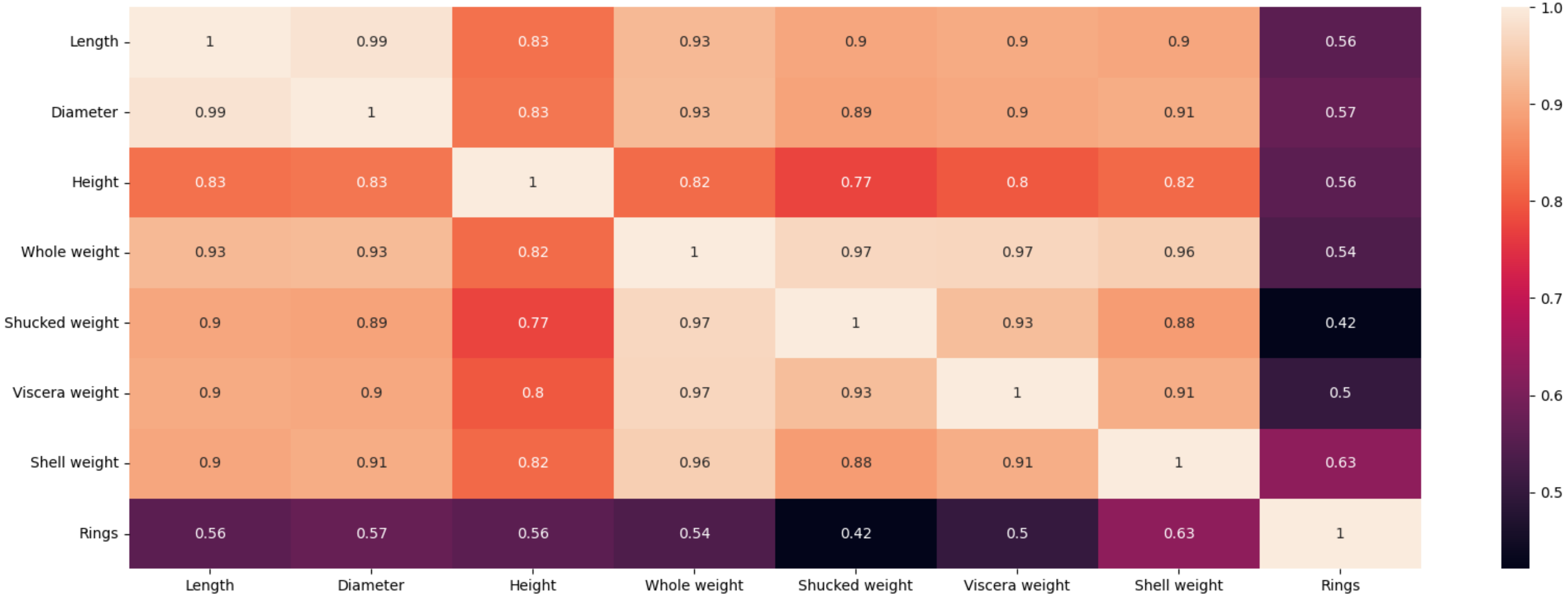


In [10]:
```
nf = df.select_dtypes(include = [np.number]).columns
cf = df.select_dtypes(include = [np.object]).columns
```

```
C:\Users\91904\AppData\Local\Temp\ipykernel_15636\561459143.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 = df.select_dtypes(include = [np.object]).columns
```

In [11]:
```
plt.figure(figsize = (20,7))
sns.heatmap(df[nf].corr(),annot = True)
```

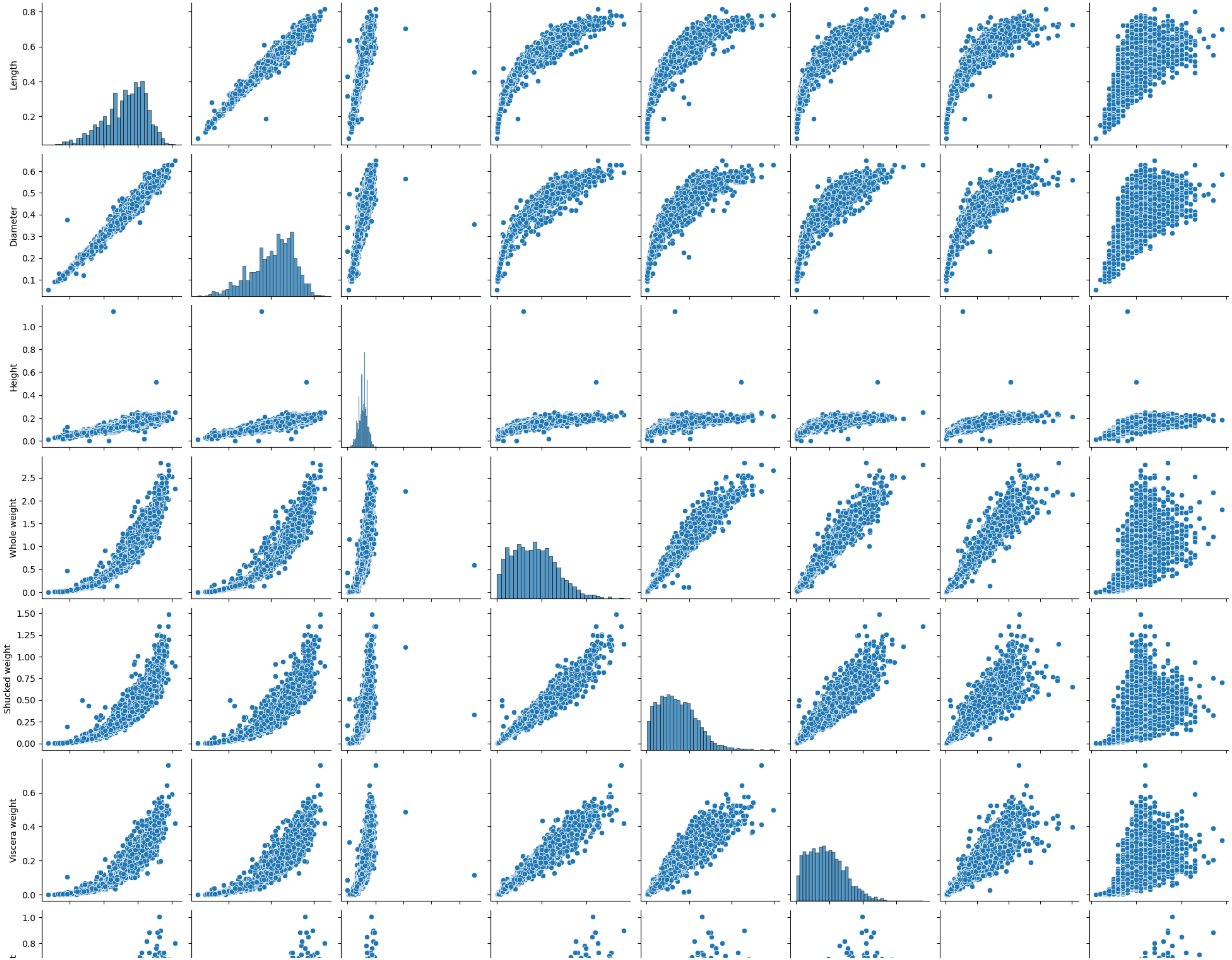
Out[11]: <AxesSubplot:>

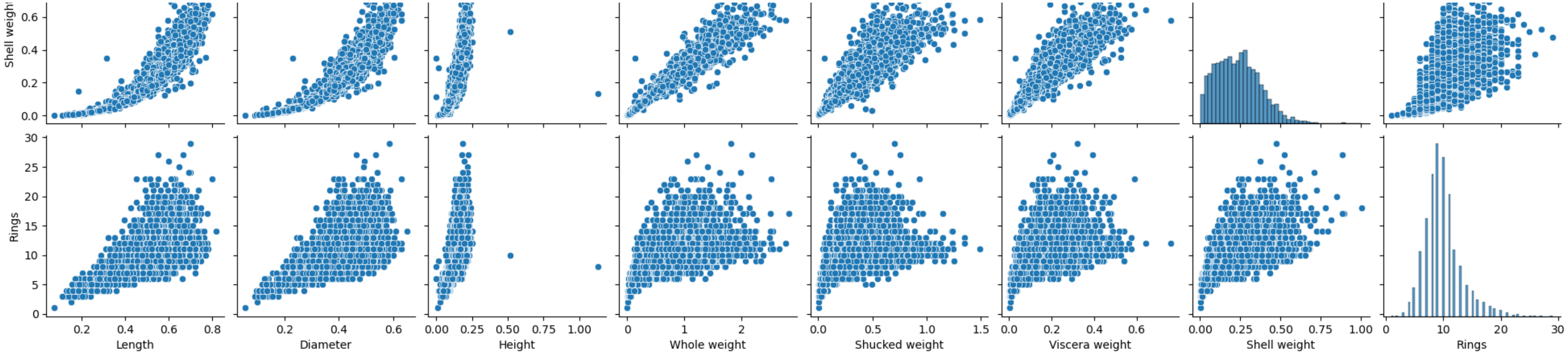


	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
Length	1	0.99	0.83	0.93	0.9	0.9	0.9	0.56
Diameter	0.99	1	0.83	0.93	0.89	0.9	0.91	0.57
Height	0.83	0.83	1	0.82	0.77	0.8	0.82	0.56
Whole weight	0.93	0.93	0.82	1	0.97	0.97	0.96	0.54
Shucked weight	0.9	0.89	0.77	0.97	1	0.93	0.88	0.42
Viscera weight	0.9	0.9	0.8	0.97	0.93	1	0.91	0.5
Shell weight	0.9	0.91	0.82	0.96	0.88	0.91	1	0.63
Rings	0.56	0.57	0.56	0.54	0.42	0.5	0.63	1

In [12]:
```
sns.pairplot(df)
```

Out[12]:
```
<seaborn.axisgrid.PairGrid at 0x1233a722280>
```


Length
Diameter
Height
Whole weight
Shucked weight
Viscera weight



In [13]:
```
df['Height'].describe()
```

Out[13]:
```
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 [14]:
```
df['Height'].mean()
```

Out[14]:
```
0.1395163993296614
```

In [15]:
```
df.max()
```

Out[15]:
```
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 [16]:
```
df['Sex'].value_counts()
```

Out[16]:

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

In [17]:

```
df[df.Height == 0]
```

Out[17]:

	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 [18]:

```
df['Shucked weight'].kurtosis()
```

Out[18]:

```
0.5951236783694207
```

In [19]:

```
df['Diameter'].median()
```

Out[19]:

```
0.425
```

In [20]:

```
df['Shucked weight'].skew()
```

Out[20]:

```
0.7190979217612694
```

In [21]:

```
df.isna().any()
```

Out[21]:

```
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 [22]:

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

Out[22]:

	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
Rings	0	0.0

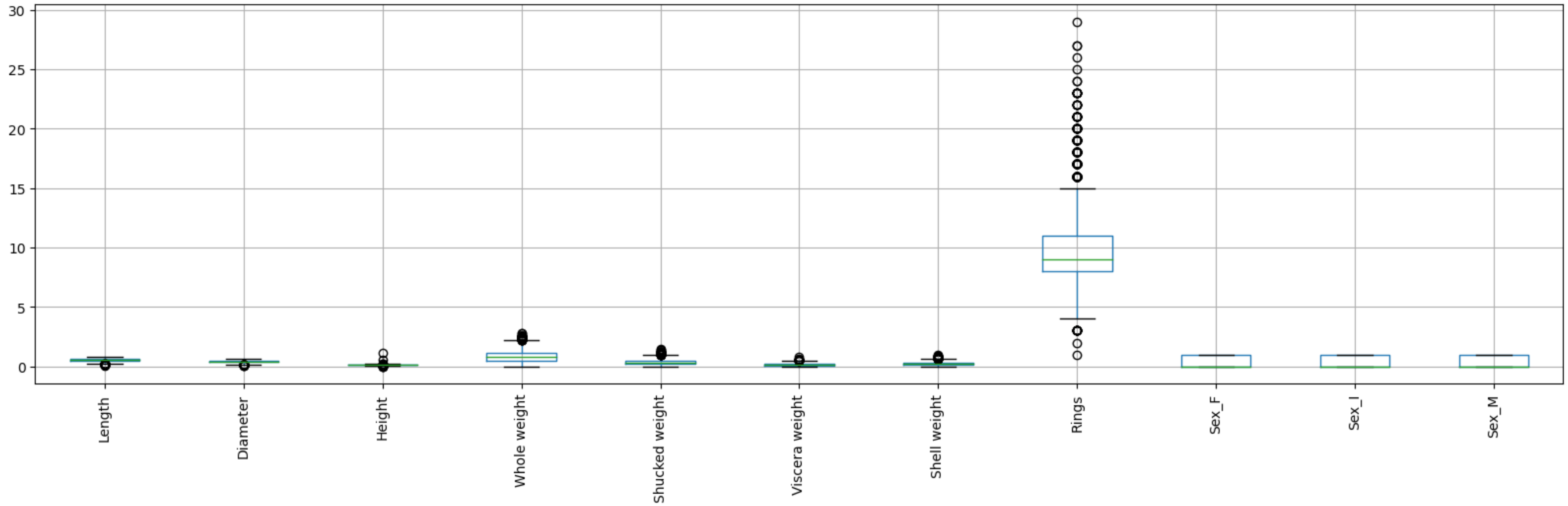
In [23]:
```
q1=df.Rings.quantile(0.25)
q2=df.Rings.quantile(0.75)
iqr=q2-q1
```

In [24]:
```
print(iqr)
```

```
3.0
```

In [25]:
```
df = pd.get_dummies(df)
dummy_df = df
df.boxplot( rot = 90, figsize=(20,5))
```

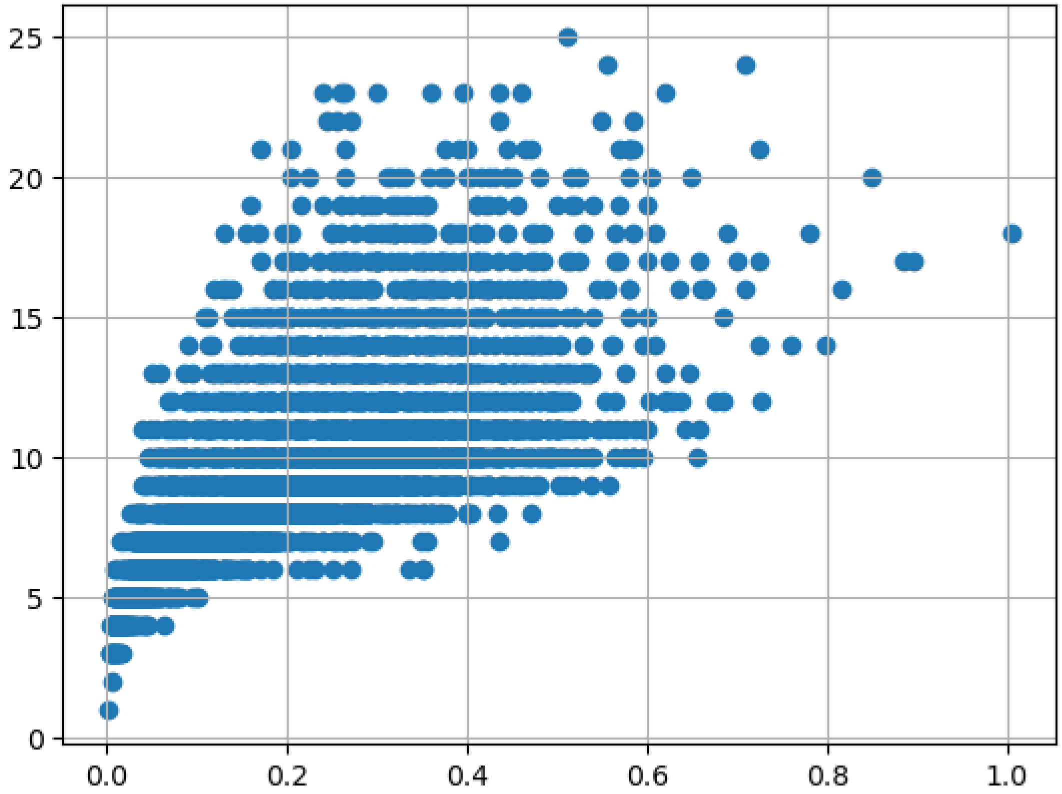
Out[25]:
```
<AxesSubplot:>
```



In [26]:
```
df['age'] = df['Rings']
df = df.drop('Rings', axis = 1)
```

In [27]:
```
df.drop(df[(df['Viscera weight']> 0.5) & (df['age'] < 20)].index, inplace=True)
df.drop(df[(df['Viscera weight']<0.5) & (df['age'] > 25)].index, inplace=True)
```

In [28]:
```
var = 'Shell weight'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)
```



In [29]:
```
numerical_features = df.select_dtypes(include = [np.number]).columns
categorical_features = df.select_dtypes(include = [np.object]).columns
```

```
C:\Users\91904\AppData\Local\Temp\ipykernel_15636\3796453440.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 = df.select_dtypes(include = [np.object]).columns
```

In [30]:
```
numerical_features
categorical_features
```

Out[30]:
```
Index([], dtype='object')
```

In [31]:
```
abalone_numeric = df[['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight','Viscera weight', 'Shell weight', 'age',
```

In [32]:
```
abalone_numeric.head()
```

Out[32]:

	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 [33]:
```
x = df.iloc[:, 0:1].values
```

In [34]:
```
y = df.iloc[:, 1]
```

In [35]:
```
y
```

Out[35]:
```
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: 4150, dtype: float64
```

In [36]:
```
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: 4150, dtype: float64
```

In [37]:

```
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:

 [[0.51351351]
 [0.37162162]
 [0.61486486]
 ...
 [0.70945946]
 [0.74324324]
 [0.85810811]]
```

In [38]:

```
X = df.drop('age', axis = 1)
y = df['age']
```

In [39]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.feature_selection import SelectKBest
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)
X_train
```

Out[39]:

```
array([[0.47 , 0.37 , 0.18 , ..., 0.   , 0.   , 1.   ],
       [0.535, 0.435, 0.15 , ..., 0.   , 0.   , 1.   ],
       [0.49 , 0.4  , 0.135, ..., 0.   , 1.   , 0.   ],
       ...,
       [0.545, 0.4  , 0.13 , ..., 0.   , 1.   , 0.   ],
       [0.59 , 0.475, 0.155, ..., 1.   , 0.   , 0.   ],
       [0.55 , 0.425, 0.15 , ..., 0.   , 1.   , 0.   ]])
```

In [40]:

```
y_train
```

Out[40]:

```
792      9
1304     9
1577     8
2772    10
507     15
        ..
3494     9
1295     9
1608     9
3283    11
581     14
Name: age, Length: 3112, dtype: int64
```

In [41]:

```
from sklearn import linear_model as lm
from sklearn.linear_model import LinearRegression
model=lm.LinearRegression()
results=model.fit(X_train,y_train)
```

In [42]:

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

```
Accuracy of the model: 0.5284655210389322
```

In [43]:

```
lm = LinearRegression()
lm.fit(X_train, y_train)
y_train_pred = lm.predict(X_train)
y_train_pred
```

Out[43]:
```
array([10.552379  ,  9.87098986,  8.41911783, ...,  8.34601774,
       11.92987898,  9.24299279])
```

In [44]:
```
X_train
```

Out[44]:
```
array([[0.47 , 0.37 , 0.18 , ..., 0.   , 0.   , 1.   ],
       [0.535, 0.435, 0.15 , ..., 0.   , 0.   , 1.   ],
       [0.49 , 0.4  , 0.135, ..., 0.   , 1.   , 0.   ],
       ...,
       [0.545, 0.4  , 0.13 , ..., 0.   , 1.   , 0.   ],
       [0.59 , 0.475, 0.155, ..., 1.   , 0.   , 0.   ],
       [0.55 , 0.425, 0.15 , ..., 0.   , 1.   , 0.   ]])
```

In [45]:
```
y_train
```

Out[45]:
```
792      9
1304     9
1577     8
2772    10
507     15
        ..
3494     9
1295     9
1608     9
3283    11
581     14
Name: age, Length: 3112, dtype: int64
```

In [46]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error
s = mean_squared_error(y_train, y_train_pred)
print('Mean Squared error :%2f'%s)
```

```
Mean Squared error :4.800536
```

In [47]:
```
y_train_pred = lm.predict(X_train)
y_test_pred = lm.predict(X_test)
```

In [48]:
```
y_test_pred
```

Out[48]:
```
array([13.22005956, 12.63289321, 10.55556642, ...,  8.8203319 ,
        9.93986695,  4.99254467])
```

In [49]:
```
X_test
```

Out[49]:
```
array([[0.595, 0.5  , 0.18 , ..., 1.   , 0.   , 0.   ],
       [0.53 , 0.455, 0.165, ..., 1.   , 0.   , 0.   ],
       [0.655, 0.515, 0.145, ..., 0.   , 1.   , 0.   ],
       ...,
       [0.54 , 0.43 , 0.14 , ..., 0.   , 1.   , 0.   ],
       [0.625, 0.485, 0.16 , ..., 0.   , 1.   , 0.   ],
       [0.185, 0.135, 0.045, ..., 0.   , 1.   , 0.   ]])
```

In [50]:
```
y_test
```

Out[50]:
```
3895    13
770     11
2405    15
2298     7
2768    11
        ..
1185     9
2212    13
1603     9
3687    11
3994     4
Name: age, Length: 1038, dtype: int64
```

In [51]:
```
p = mean_squared_error(y_test, y_test_pred)
print('Mean Squared error of testing set :%2f'%p)
```

```
Mean Squared error of testing set :4.431685
```

In [52]:
```
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:0.53
```

In [53]:
```
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:0.56
```

In [ ]: