

**Assignment -3**  
**Problem Statement-Abalone Age Prediction**

Assignment Date	06 October 2022
Student Name	Miss.Kiruthika M
Student Roll Number	620119106043
Maximum Marks	2 Marks

**Question-1:**

**1.download the data set**

**2.load the dataset**

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn
```

```
data=pd.read_csv("abalone.csv")data.head()
```

	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

```
Age=1.5+data.Ringsdata["Age"]=Agedata=data.rename(columns = {'Whole weight':'Whole_weight','Shucked weight': 'Shucked_weight','Viscera weight': 'Viscera_weight',
                    'Shell weight': 'Shell_weight'})data=data.drop(columns=["Rings"],axis=1)data.head()
```

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Age
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	16.5
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	8.5
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	10.5
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	11.5
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	8.5

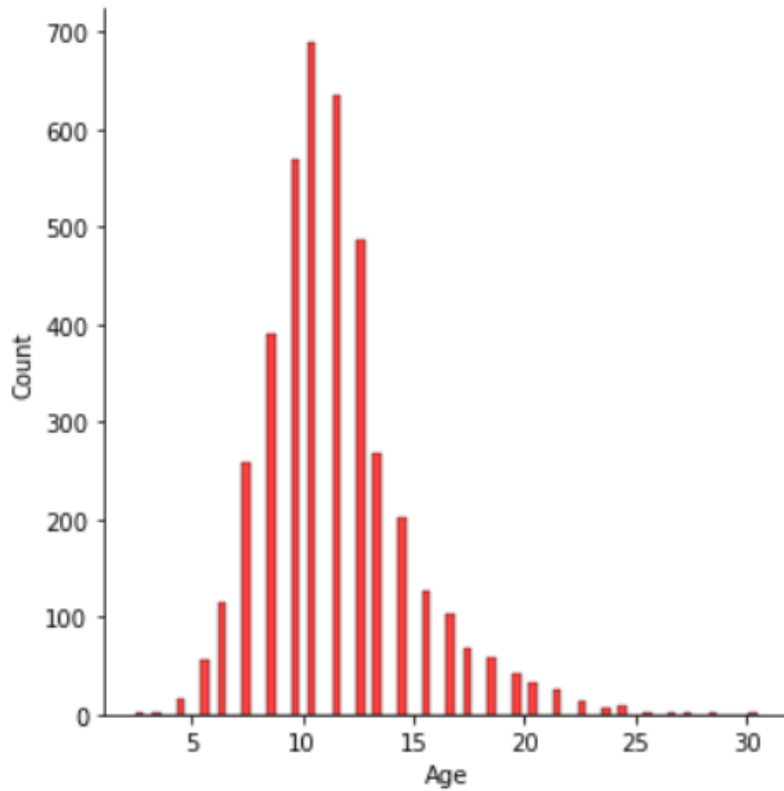
**3.perform below visualizations**  
**univariate analysis**

**Ans:**

```
sns.displot(data["Age"], color='red')
```

Out[4]:

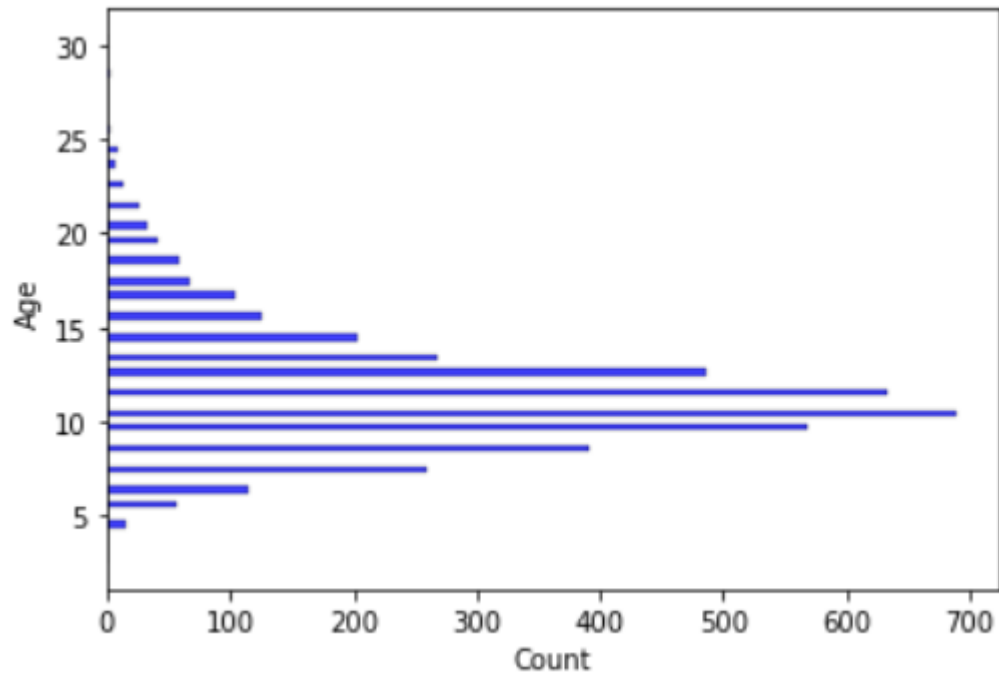
<seaborn.axisgrid.FacetGrid at 0x187953d26a0>



```
sns.histplot(y=data.Age,color='blue')
```

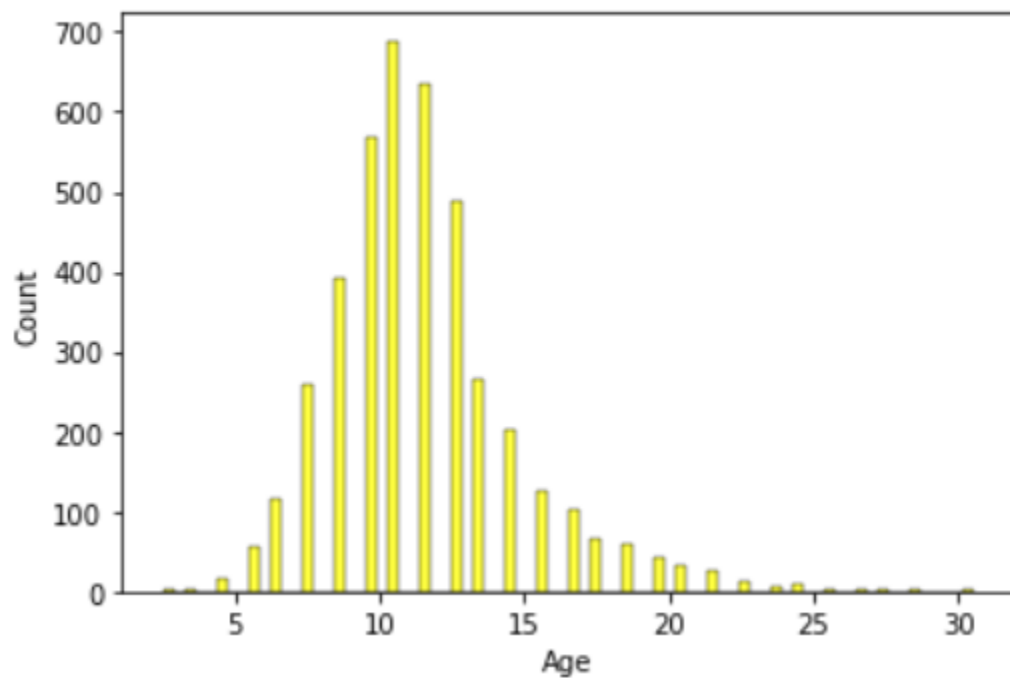
Out[5]:

<AxesSubplot:xlabel='Count', ylabel='Age'>



```
sns.histplot(x=data.Age,color='yellow')
```

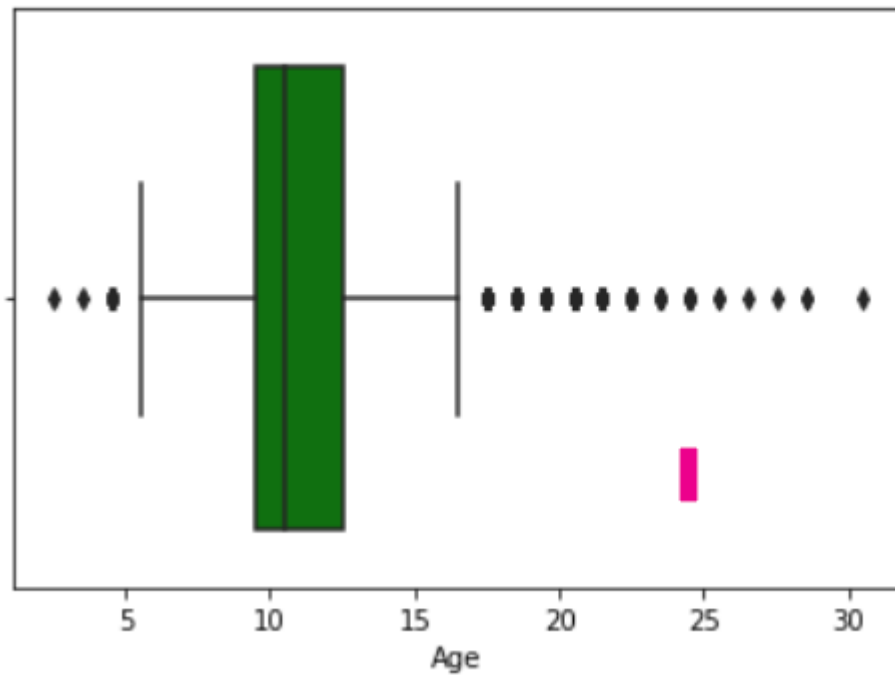
```
<AxesSubplot:xlabel='Age', ylabel='Count'>
```



In [6]:

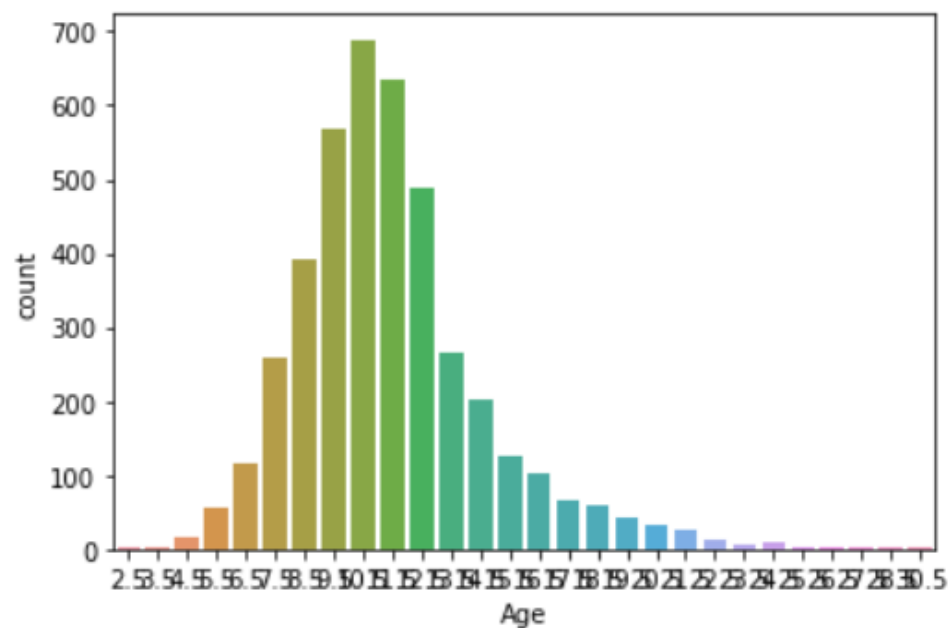
```
sns.boxplot(x=data.Age,color='green')
```

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



```
sns.countplot(x=data.Age)
```

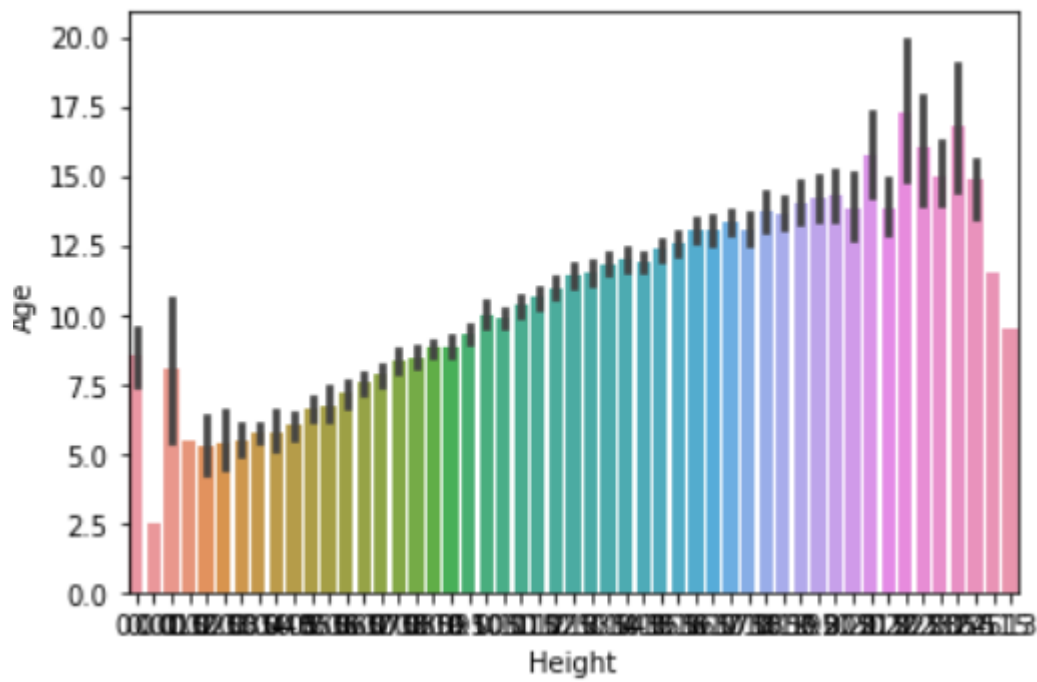
```
<AxesSubplot:xlabel='Age', ylabel='count'>
```



## bi - variate analysis

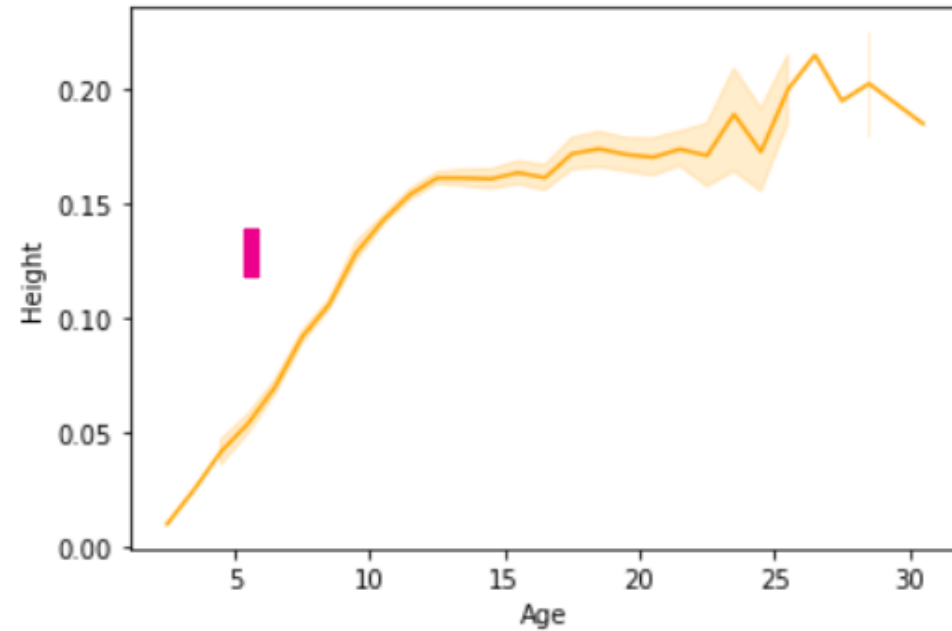
```
sns.barplot(x=data.Height,y=data.Age)
```

```
<AxesSubplot:xlabel='Height', ylabel='Age'>
```



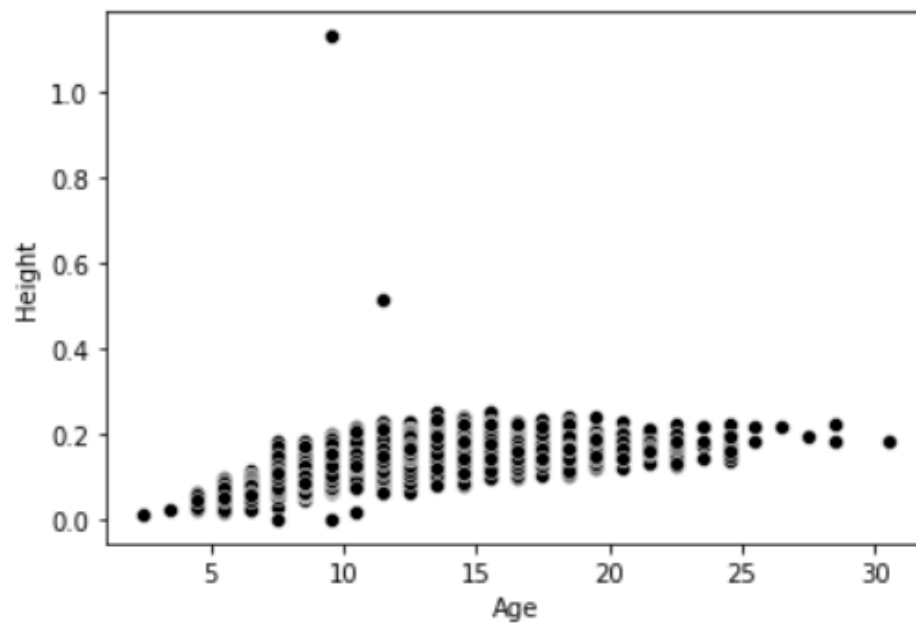
```
sns.lineplot(x=data.Age,y=data.Height, color='orange')
```

```
<AxesSubplot:xlabel='Age', ylabel='Height'>
```



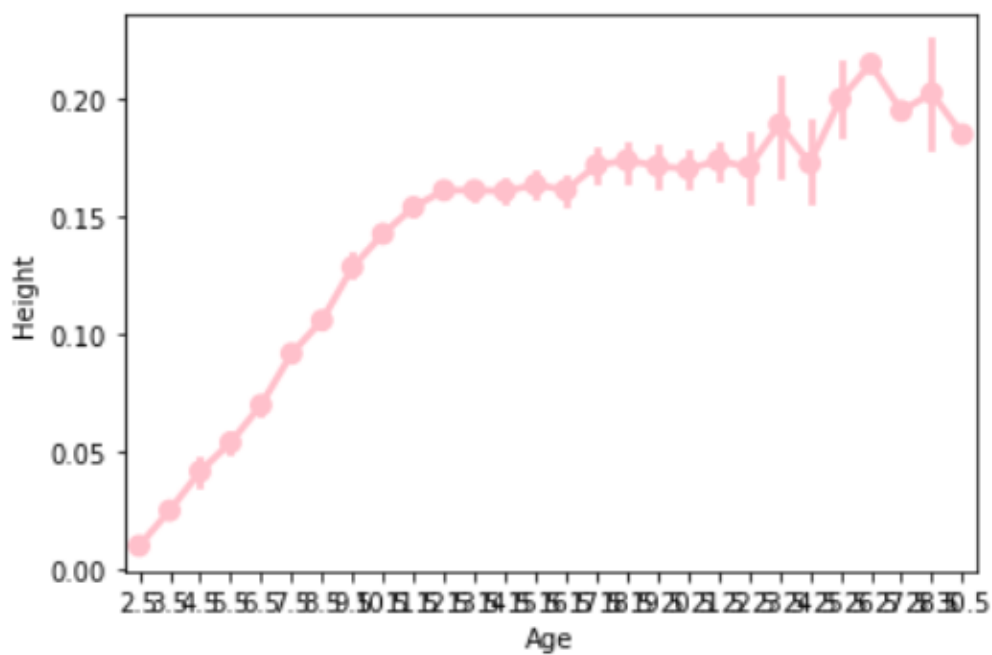
```
sns.scatterplot(x=data.Age,y=data.Height,color='black')
```

```
<AxesSubplot:xlabel='Age', ylabel='Height'>
```



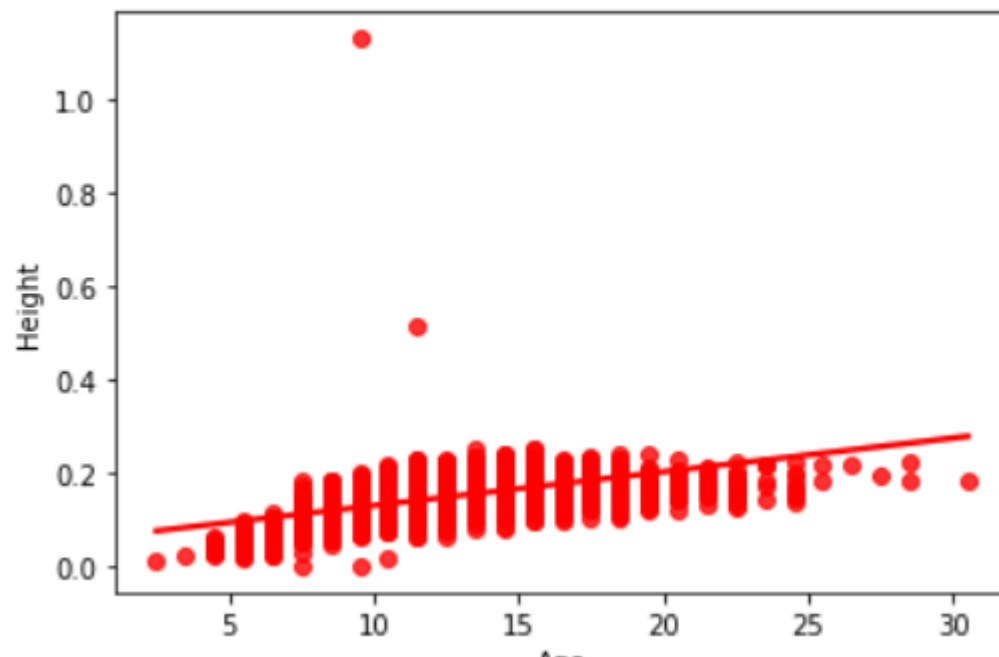
```
sns.pointplot(x=data.Age, y=data.Height, color="pink")
```

```
<AxesSubplot:xlabel='Age', ylabel='Height'>
```



```
sns.regplot(x=data.Age, y=data.Height, color='red')
```

```
<AxesSubplot:xlabel='Age', ylabel='Height'>
```

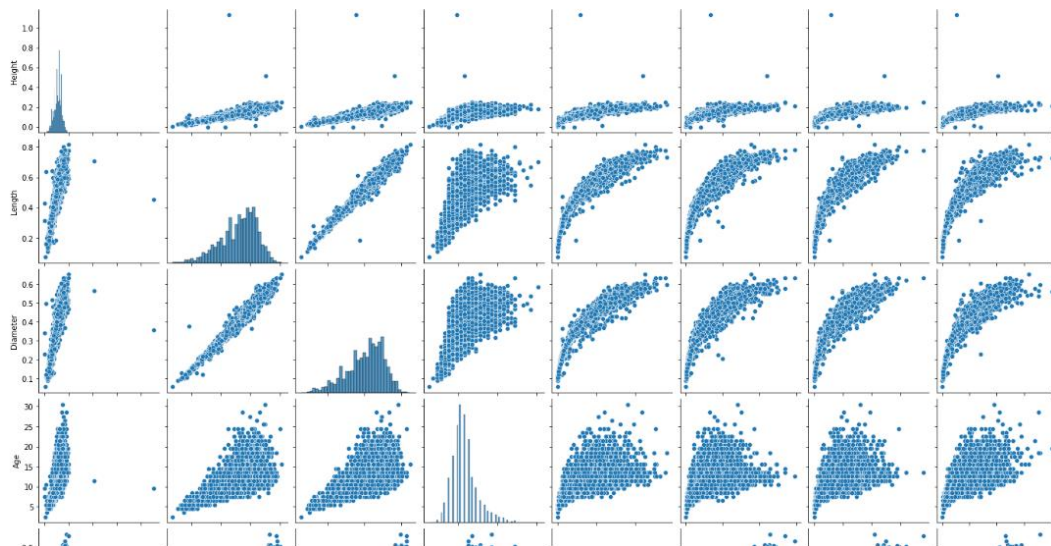


## multi - variate analysis

In [9]:

```
sns.pairplot(data=data[["Height","Length","Diameter","Age","Whole_weight","Shucked_weight","Viscera_weight","Shell_weig
```

```
15]: <seaborn.axisgrid.PairGrid at 0x18795288400>
```



## 4. descriptive statistics

```
data.describe(include='all')
```

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Age
count	4177	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000
unique	3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	M	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	1528	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mean	NaN	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0.238831	11.433684
std	NaN	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.139203	3.224169
min	NaN	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.001500	2.500000
25%	NaN	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.130000	9.500000
50%	NaN	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.234000	10.500000
75%	NaN	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.329000	12.500000
max	NaN	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.005000	30.500000

## 5. Check for Missing values and deal with them

```
data.isnull().sum()
```

```
Sex          0
Length       0
Diameter     0
Height       0
Whole_weight 0
Shucked_weight 0
Viscera_weight 0
Shell_weight 0
Age          0
dtype: int64
```

## 6. Find the outliers and replace them

### outliers

```
In [18]:
```

```
outliers=data.quantile(q=(0.25,0.75))outliers
```



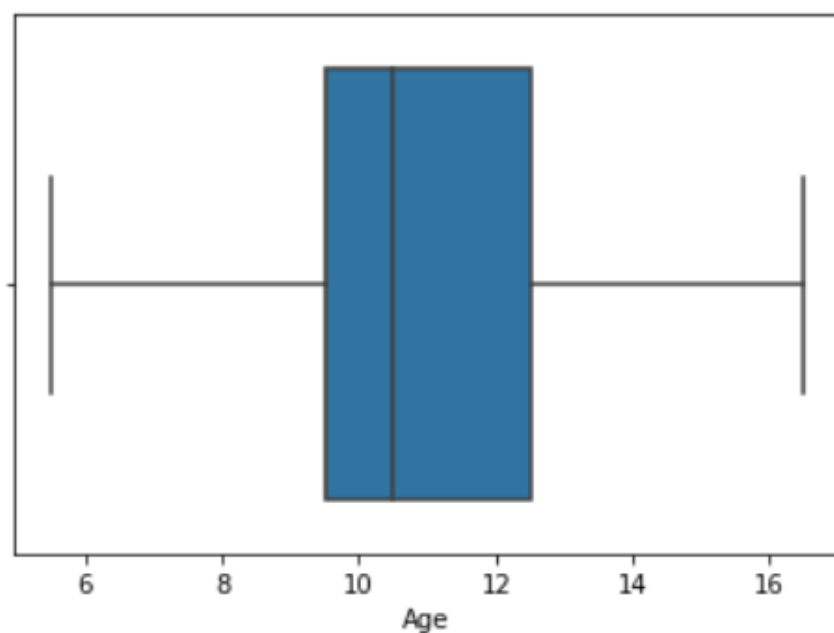
	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Age
<b>0.25</b>	0.450	0.35	0.115	0.4415	0.186	0.0935	0.130	9.5
<b>0.75</b>	0.615	0.48	0.165	1.1530	0.502	0.2530	0.329	12.5

```
a = data.Age.quantile(0.25)b = data.Age.quantile(0.75)c = b - a
lower_limit = a - 1.5 * c
data.median(numeric_only=True)
```

```
Length          0.5450
Diameter        0.4250
Height          0.1400
Whole_weight    0.7995
Shucked_weight  0.3360
Viscera_weight  0.1710
Shell_weight    0.2340
Age            10.5000
dtype: float64
```

```
data['Age'] = np.where(data['Age'] < lower_limit, 7, data['Age'])
sns.boxplot(x=data.Age, showfliers=False)
```

<AxesSubplot: xlabel= 'Age'>



## 7. Check for Categorical columns and perform encoding

In [21]:

```
data.head()
```

Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Age
M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	16.5
M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	8.5
F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	10.5
M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	11.5
I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	8.5

```
from sklearn.preprocessing import LabelEncoder
lab = LabelEncoder()data.Sex = lab.fit_transform(data.Sex)
data.head()
```

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Age
0	2	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	16.5
1	2	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	8.5
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	10.5
3	2	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	11.5
4	1	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	8.5

## 8. Split the data into dependent and independent variables

```
y = data["Sex"] y.head()
```

In [24]:

```
x=data.drop(columns=["Sex"],axis=1)x.head()
```

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Age
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	16.5
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	8.5
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	10.5
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	11.5
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	8.5

## 9. Scale the independent variables

In [25]:

```
from sklearn.preprocessing import scale
X_Scaled = pd.DataFrame(scale(x), columns=x.columns)
X_Scaled.head()
```

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Age
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	16.5
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	8.5
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	10.5
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	11.5
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	8.5

## 10. Split the data into training and testing

In [26]:

```
from sklearn.model_selection import train_test_split
```

```
X_Train, X_Test, Y_Train, Y_Test = train_test_split(X_Scaled, y, test_size=0.2,
random_state=0)
```

In [27]:

```
X_Train.shape,X_Test.shape
```

Output

```
((3341, 8), (836, 8))
```

```
Y_Train.shape,Y_Test.shape
```

output

```
((3341,), (836,))
```

In [29]:

```
X_Train.head()
```

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Age
3141	-2.864726	-2.750043	-1.423087	-1.622870	-1.553902	-1.583867	-1.644065	-1.543234
3521	-2.573250	-2.598876	-2.020857	-1.606554	-1.551650	-1.565619	-1.626104	-1.387181
883	1.132658	1.230689	0.728888	1.145672	1.041436	0.286552	1.538726	1.577830
3627	1.590691	1.180300	1.446213	2.164373	2.661269	2.330326	1.377072	0.017298
2106	0.591345	0.474853	0.370226	0.432887	0.255175	0.272866	0.906479	1.265723

X\_Test.head()

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Age
3141	-2.864726	-2.750043	-1.423087	-1.622870	-1.553902	-1.583867	-1.644065	-1.543234
3521	-2.573250	-2.598876	-2.020857	-1.606554	-1.551650	-1.565619	-1.626104	-1.387181
883	1.132658	1.230689	0.728888	1.145672	1.041436	0.286552	1.538726	1.577830
3627	1.590691	1.180300	1.446213	2.164373	2.661269	2.330326	1.377072	0.017298
2106	0.591345	0.474853	0.370226	0.432887	0.255175	0.272866	0.906479	1.265723

Y\_Train.head()

```
3141    1
3521    1
883     2
3627    2
2106    2
Name: Sex, dtype: int32
```

```
Y_Test.head()
```

## 11. Build the Model

```
In [33]:
```

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=10,criterion='entropy')
```

```
In [34]:
```

```
model.fit(X_Train,Y_Train)
```

Output

```
RandomForestClassifier(criterion='entropy', n_estimators=10)
```

## 12. Train the Model

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

```
In [38]:
```

```
print('Training accuracy: ',accuracy_score(Y_Train,y_predict_train))
```

Output

```
Training accuracy: 0.9817419934151451
```

## 13. Test the Model

```
In [39]:
```

```
print('Testing accuracy: ',accuracy_score(Y_Test,y_predict))
```

Output

```
Testing accuracy: 0.5322966507177034
```

## 14. Measure the performance using Metrics

```
In [40]:
```

```
pd.crosstab(Y_Test,y_predict)
```

```
col_0    0    1    2
```

```
Sex
```

```
0  116   29  104
```

```
1   37  216   38
```

```
2  125   58  113
```

```
print(classification_report(Y_Test,y_predict))
```

	precision	recall	f1-score	support
0	0.42	0.47	0.44	249
1	0.71	0.74	0.73	291
2	0.44	0.38	0.41	296
accuracy			0.53	836
macro avg	0.52	0.53	0.53	836
weighted avg	0.53	0.53	0.53	836