1.Download the dataset

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import scale
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

2. Load the Dataset

```
from google.colab import files
uploaded = files.upload()
```

Choose Files abalone.csv

- **abalone.csv**(text/csv) - 191962 bytes, last modified: 10/6/2022 - 100% done

```
Saving abalone.csv to abalone.csv
```

```
data= pd.read_csv("/content/abalone.csv",encoding='latin1',low_memory=False)
```

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

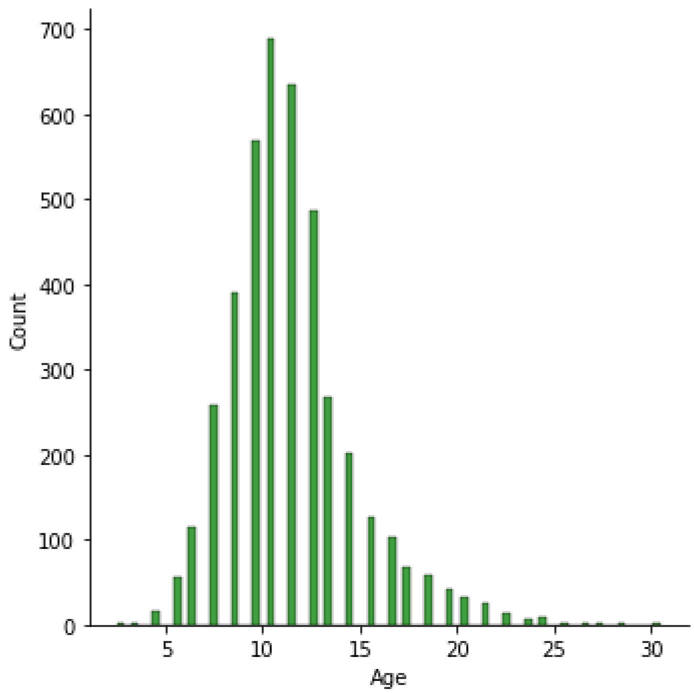
	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_w
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	

3.Performing Analysis

1. Univariate Analysis

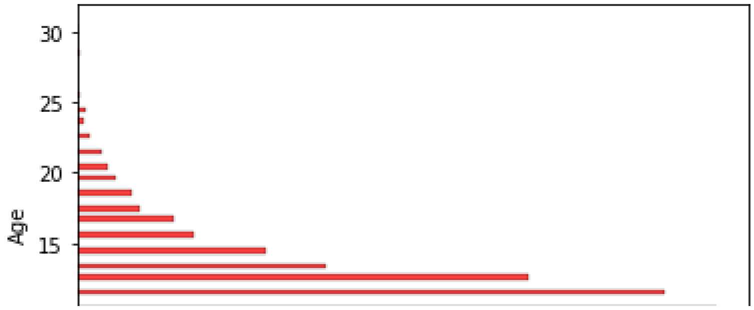
```
#Histogram
sns.displot(data["Age"], color='Green')
```

<seaborn.axisgrid.FacetGrid at 0x7f7fd68e1950>



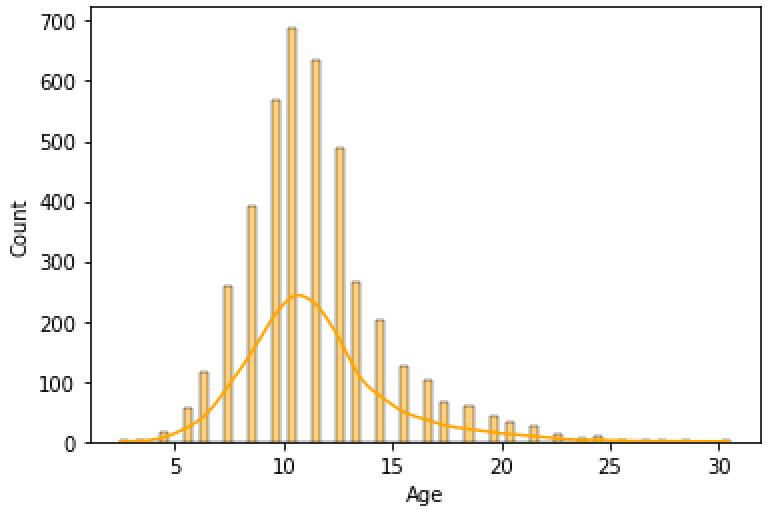

```
sns.histplot(y=data.Age,color='red')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7fd39e5a10>
```



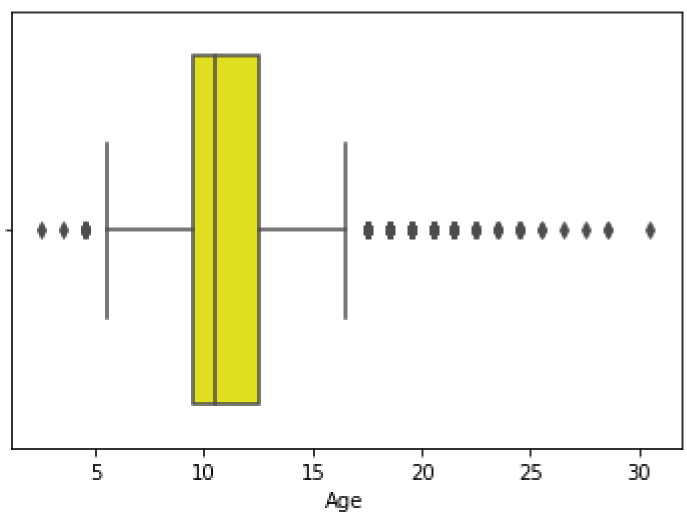

```
sns.histplot(x=data.Age,kde=True,color='orange')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7fd3e9cd90>
```



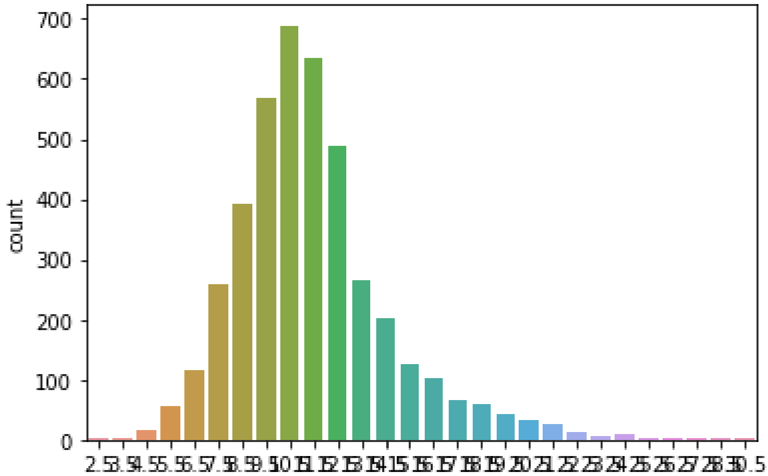

```
sns.boxplot(x=data.Age,color='yellow')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7fd386bb10>
```




```
# Count-plot
sns.countplot(x=data.Age)
```

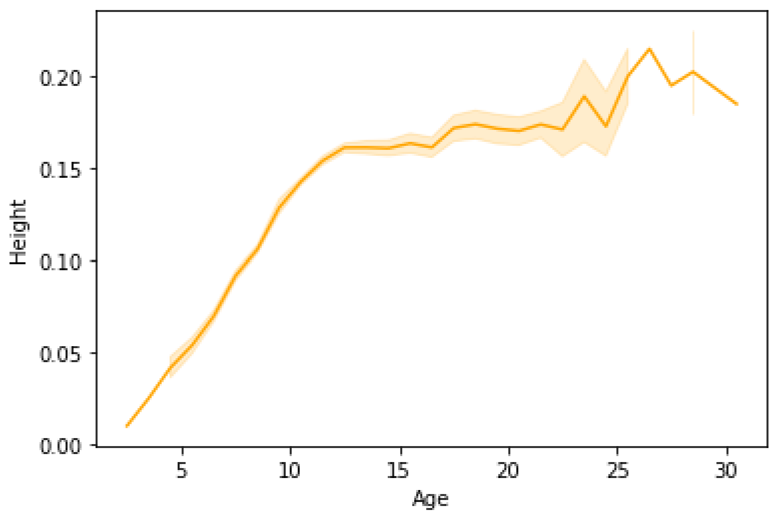
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7fd38a4090>
```



2. Bi - variate Analysis

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7fd3792f10>
```

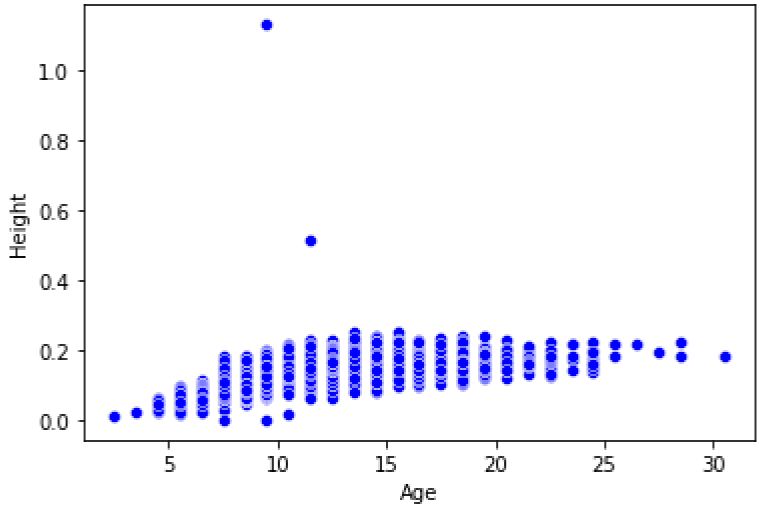



```
#Bar-plot
sns.barplot(x=data.Height,y=data.Age)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7fd37846d0>
```

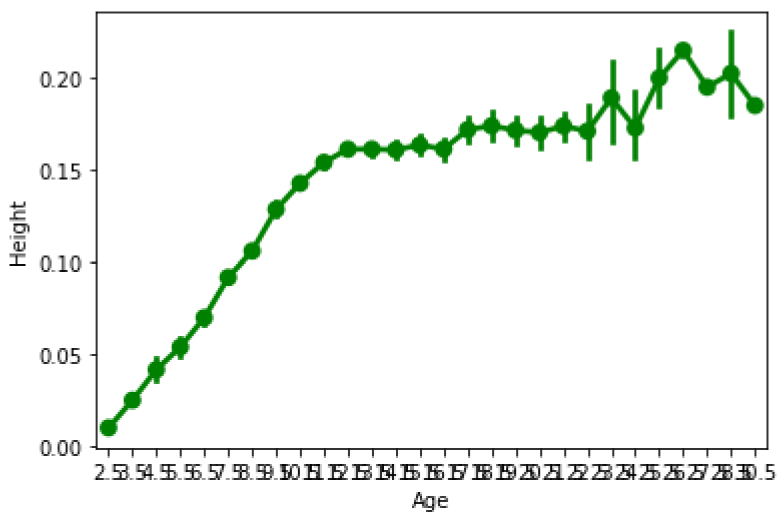

```
#Scatterplot
sns.scatterplot(x=data.Age,y=data.Height,color='blue')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7fd3784510>
```



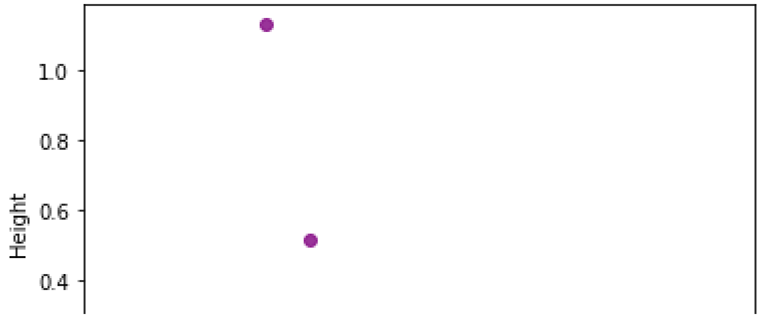

```
#point-plot
sns.pointplot(x=data.Age, y=data.Height, color="green")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7fd32b2890>
```




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

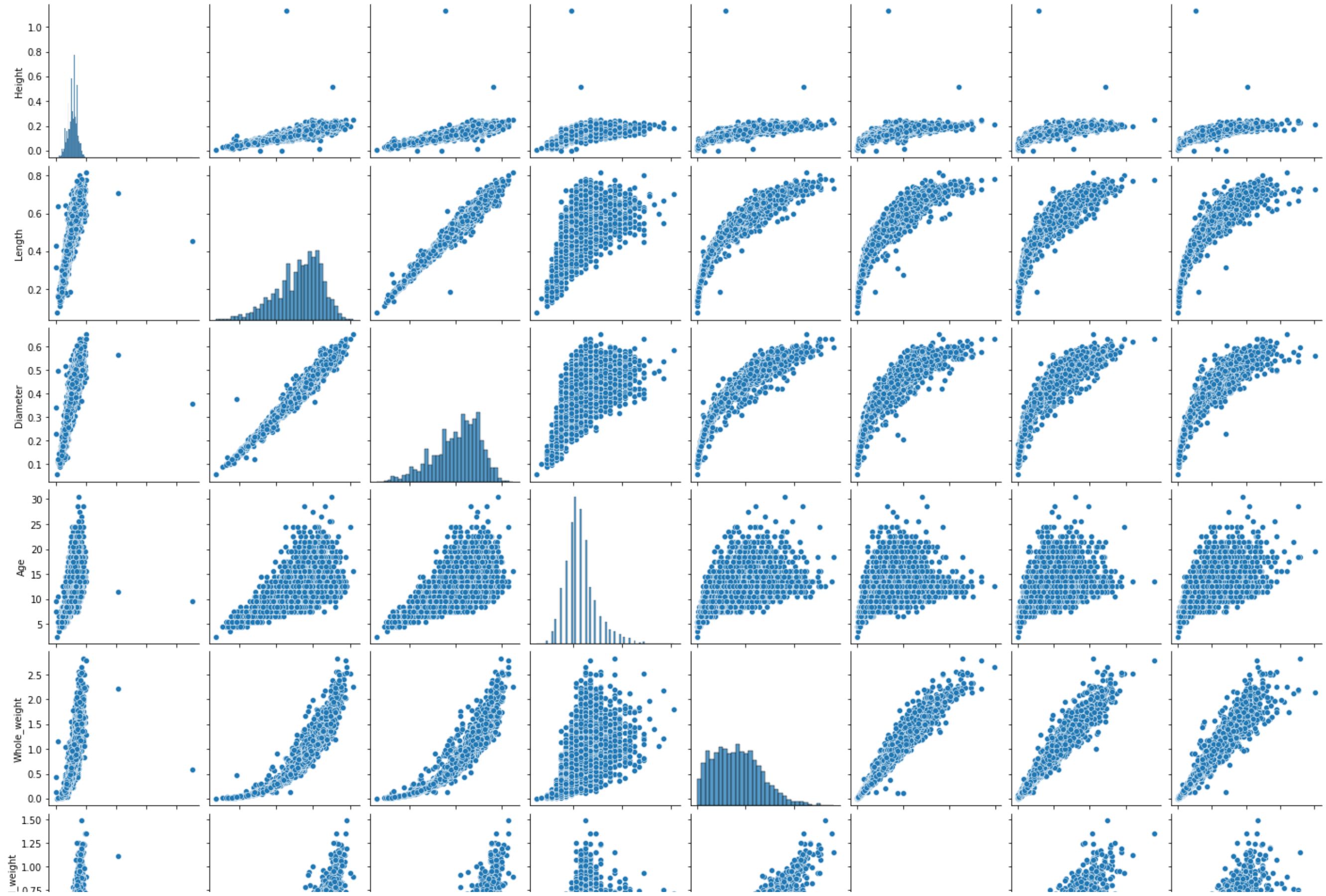
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7fd31caa90>
```



3. Multi-Variate Analysis

```
#pair-plot
sns.pairplot(data=data[["Height","Length","Diameter","Age","Whole_weight","Shucked_weight","\
```

```
<seaborn.axisgrid.PairGrid at 0x7f7fd125e990>
```



4. Perform descriptive statistics on the dataset

```
#mode
data[['Length','Diameter','Whole_weight','Shucked_weight','Viscera_weight','Shell_weight']].m
```

```
Length            0.523992
Diameter          0.407881
Whole_weight      0.828742
Shucked_weight    0.359367
Viscera_weight    0.180594
Shell_weight      0.238831
dtype: float64
```

```
#median
data[['Length','Diameter','Whole_weight','Shucked_weight','Viscera_weight','Shell_weight']].m
```

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

```
data[['Length','Diameter','Whole_weight','Shucked_weight','Viscera_weight','Shell_weight']].m
```

	Length	Diameter	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight
0	0.550	0.45	0.2225	0.175	0.1715	0.275
1	0.625	NaN	NaN	NaN	NaN	NaN

```
#qunatile
data[['Length','Diameter','Whole_weight','Shucked_weight','Viscera_weight','Shell_weight']].c
```

```
Length            0.5450
Diameter          0.4250
Whole_weight      0.7995
Shucked_weight    0.3360
Viscera_weight    0.1710
Shell_weight      0.2340
Name: 0.5, dtype: float64
```

```
#Standard - deviatiion
data[['Length','Diameter','Whole_weight','Shucked_weight','Viscera_weight','Shell_weight']].s
```

```
Length            0.120093
Diameter          0.099240
Whole_weight      0.490389
Shucked_weight    0.221963
Viscera_weight    0.109614
Shell_weight      0.139203
dtype: float64
```

```
#min
data[['Length','Diameter','Whole_weight','Shucked_weight','Viscera_weight','Shell_weight']].m
```

```
Length            0.0750
Diameter          0.0550
Whole_weight      0.0020
Shucked_weight    0.0010
Viscera_weight    0.0005
Shell_weight      0.0015
dtype: float64
```

```
#max
data[['Length','Diameter','Whole_weight','Shucked_weight','Viscera_weight','Shell_weight']].m
```

```
Length            0.8150
Diameter          0.6500
Whole_weight      2.8255
Shucked_weight    1.4880
Viscera_weight    0.7600
Shell_weight      1.0050
dtype: float64
```

```
#Skew
data[['Length','Diameter','Whole_weight','Shucked_weight','Viscera_weight','Shell_weight']].s
```

```
Length           -0.639873
Diameter         -0.609198
Whole_weight      0.530959
Shucked_weight    0.719098
Viscera_weight    0.591852
Shell_weight      0.620927
dtype: float64
```

```
data.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   Age             4177 non-null   float64
dtypes: float64(8), object(1)
memory usage: 293.8+ KB
```

```
data.shape
```

```
(4177, 9)
```

```
data.describe()
```

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weig

5. Check for Missing values and deal with them.

std	0.120093	0.099240	0.041827	0.490389	0.221963	0.1096

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

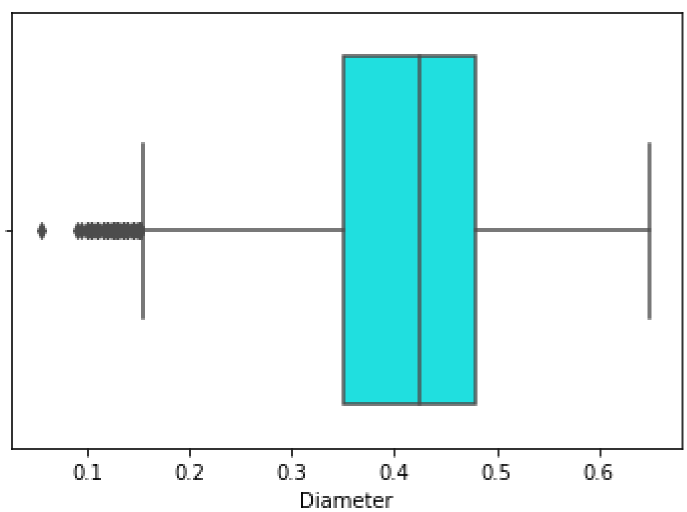
```
Sex               0
Length            0
Diameter          0
Height            0
Whole_weight      0
Shucked_weight    0
Viscera_weight    0
Shell_weight      0
Age               0
dtype: int64
```

Hence ,There is no missing values in the dataset

6. Find the outliers and replace them outliers

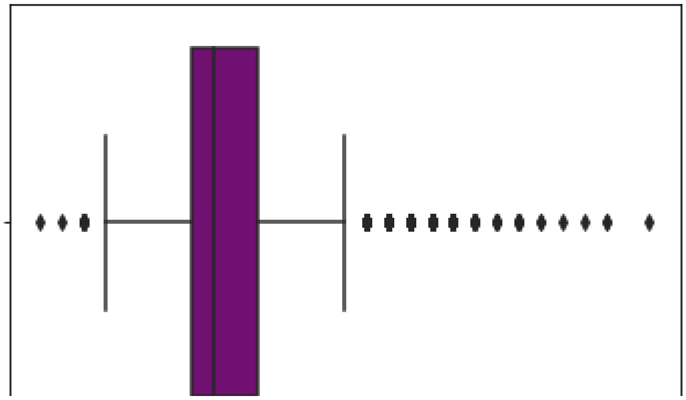
```
sns.boxplot(x=data["Diameter"],color="cyan")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7fcd94b990>
```




```
sns.boxplot(x=data["Age"],color="purple")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7fcd883390>
```



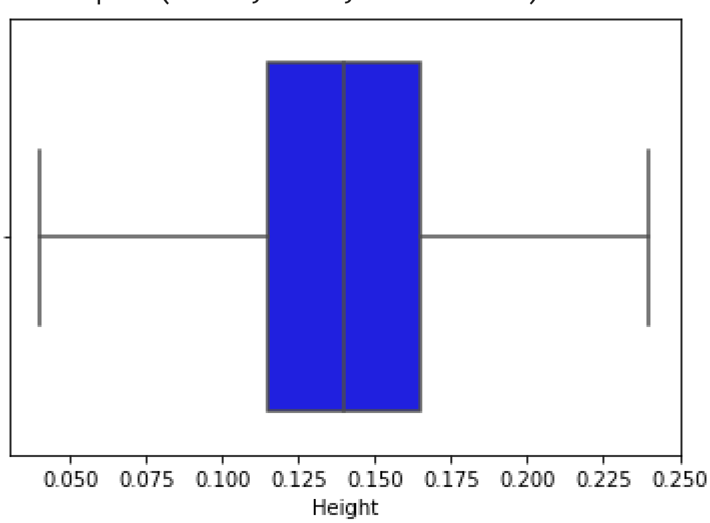
```
outliers=data.quantile(q=(0.25,0.75))
outliers
```

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_wei
0.25	0.450	0.35	0.115	0.4415	0.186	0.0935	0.
0.75	0.615	0.48	0.165	1.1530	0.502	0.2530	0.

```
for i in data:
    if data[i].dtype=='int64' or data[i].dtypes=='float64':
        q1=data[i].quantile(0.25)
        q3=data[i].quantile(0.75)
        iqr=q3-q1
        upper=q3+1.5*iqr
        lower=q1-1.5*iqr
        data[i]=np.where(data[i] >upper, upper, data[i])
        data[i]=np.where(data[i] <lower, lower, data[i])
```

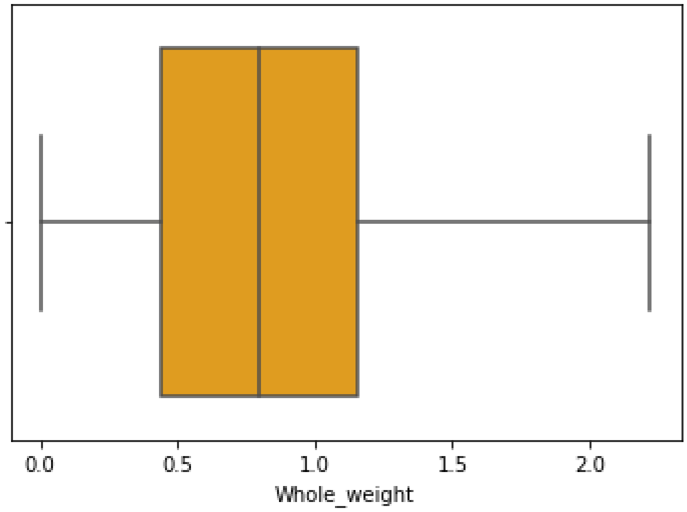
```
print(sns.boxplot(x=data["Height"],color="blue"))
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```



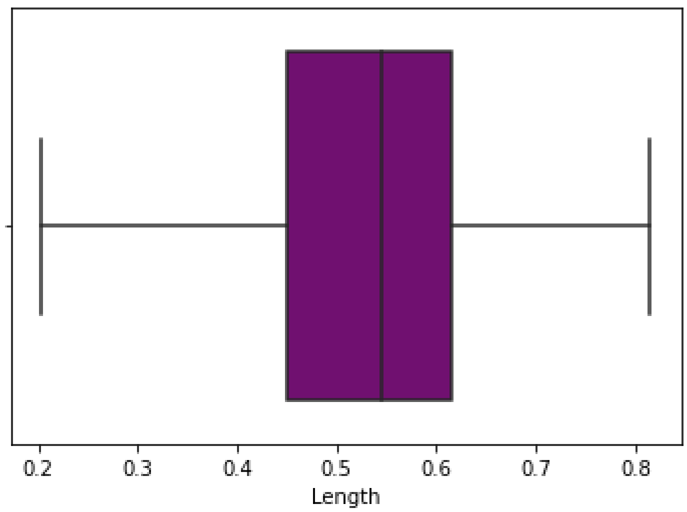
```
sns.boxplot(x=data["Whole_weight"],color="orange")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7fcd67dd90>
```




```
sns.boxplot(x=data["Length"],color="purple")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7fcd6838d0>
```



7. Checking for Categorical columns and performing encoding

```
label = LabelEncoder()
data.Length = label.fit_transform(data.Length)
data.head()
```

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_w
0	2	51	0.365	0.095	0.5140	0.2245	0.1010	
1	2	30	0.265	0.090	0.2255	0.0995	0.0485	
2	0	66	0.420	0.135	0.6770	0.2565	0.1415	

8. Splitting the data into dependent and independent variables

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

```
0    2
1    2
2    0
3    2
4    1
Name: Sex, dtype: int64
```

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

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight
0	51	0.365	0.095	0.5140	0.2245	0.1010	0.150
1	30	0.265	0.090	0.2255	0.0995	0.0485	0.070
2	66	0.420	0.135	0.6770	0.2565	0.1415	0.210
3	48	0.365	0.125	0.5160	0.2155	0.1140	0.155
4	26	0.255	0.080	0.2050	0.0895	0.0395	0.055

9. Scale the independent variables

```
X_Scaled = pd.DataFrame(scale(x), columns=x.columns)
X_Scaled.head()
```

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_w
0	-0.582509	-0.440884	-1.158093	-0.644740	-0.614985	-0.730304	-0.6
1	-1.464659	-1.459762	-1.288751	-1.238208	-1.191637	-1.213890	-1.2
2	0.047599	0.119499	-0.112828	-0.309436	-0.467362	-0.357253	-0.2

10. Splitting the data into training and testing

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

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

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

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

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

```
X_Train.head()
```

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shel
3141	-2.724873	-2.580528	-1.550067	-1.634195	-1.583761	-1.596153	
3521	-2.598852	-2.580528	-2.203358	-1.617739	-1.581454	-1.577730	
883	1.139785	1.240265	0.801778	1.158289	1.073452	0.292134	
3627	1.601863	1.189321	1.585727	2.185800	2.731903	2.355432	
2106	0.593692	0.476107	0.409804	0.439340	0.268446	0.278317	

```
X_Test.head()
```

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shel
668	0.215627	0.170443	0.409804	0.185291	-0.370485	0.577680	
1580	-0.204444	-0.084276	-0.504803	-0.434918	-0.446603	-0.343436	
3784	0.803727	0.730826	0.409804	0.880584	0.780513	1.784341	
463	-2.556844	-2.478640	-2.203358	-1.589968	-1.551468	-1.550097	
2615	1.013763	0.934602	0.932437	1.405139	1.456349	1.798157	

```
Y_Train.head()
```

```
3141    1
3521    1
883     2
3627    2
2106    2
Name: Sex, dtype: int64
```

```
Y_Test.head()
```

```
668     2
1580    1
3784    2
463     1
2615    2
Name: Sex, dtype: int64
```

11. Building the Model

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

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

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

```
y_predict = model.predict(X_Test)
```

```
y_predict_train = model.predict(X_Train)
```

12. Train the Model

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

```
Training accuracy:  0.98263992816522
```

13.Test the Model

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

```
Testing accuracy:  0.5466507177033493
```

14. Measuring the performance using Metrics

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

col_0	0	1	2
Sex			
0	118	38	93
1	39	216	36
2	125	48	123

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

```
              precision    recall  f1-score   support

           0       0.42      0.47      0.44       249
           1       0.72      0.74      0.73       291
           2       0.49      0.42      0.45       296

    accuracy                           0.55       836
   macro avg       0.54      0.54      0.54       836
weighted avg       0.55      0.55      0.54       836
```