Team Id : PNT2022TMID07232
Name : Rajaram K

```
import numpy as np import pandas as pd import
matplotlib.pyplot as plt import seaborn as sns from
pandas.api.types import is_numeric_dtype sns.set()
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder from
sklearn.preprocessing import StandardScaler
sns.set_style("darkgrid") from sklearn.linear_model
import LinearRegression from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor

from sklearn import  metrics
%matplotlib inline
```

LOADING ABALONE DATASET

```
abalone = pd.read_csv('abalone.csv', sep=',')
```

```
abalone.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

UNIVARIATE ANALYSIS

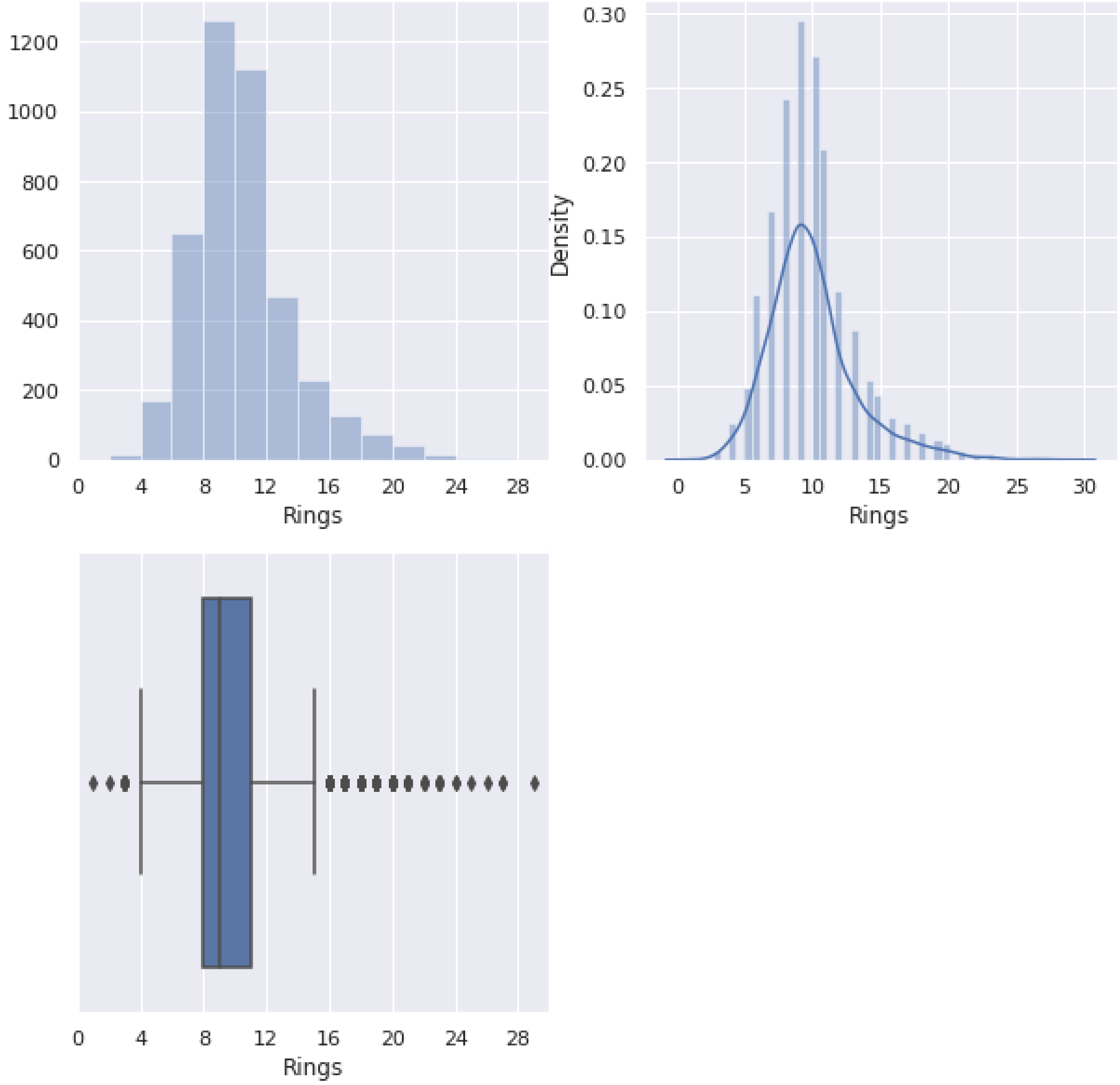
Here, we analyze the target variable (Rings), size, weight and sex.

1) Target Variable (Ring)

```
rows = 2 cols = 2

i = 0 plt.figure(figsize=(cols * 5, rows

* 5)) i += 1

plt.subplot(rows, cols, i)
plt.xticks(range(0, 31, 4)) plt.xlim(0,
30)
_ = sns.distplot(abalone['Rings'], kde=False, bins=range(0, 31, 2))

i += 1
plt.subplot(rows, cols, i) _ =
sns.distplot(abalone['Rings'])
```

```
i += 1
plt.subplot(rows, cols, i)
plt.xticks(range(0, 31, 4)) plt.xlim(0,
30)
_ = sns.boxplot(abalone['Rings'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
```



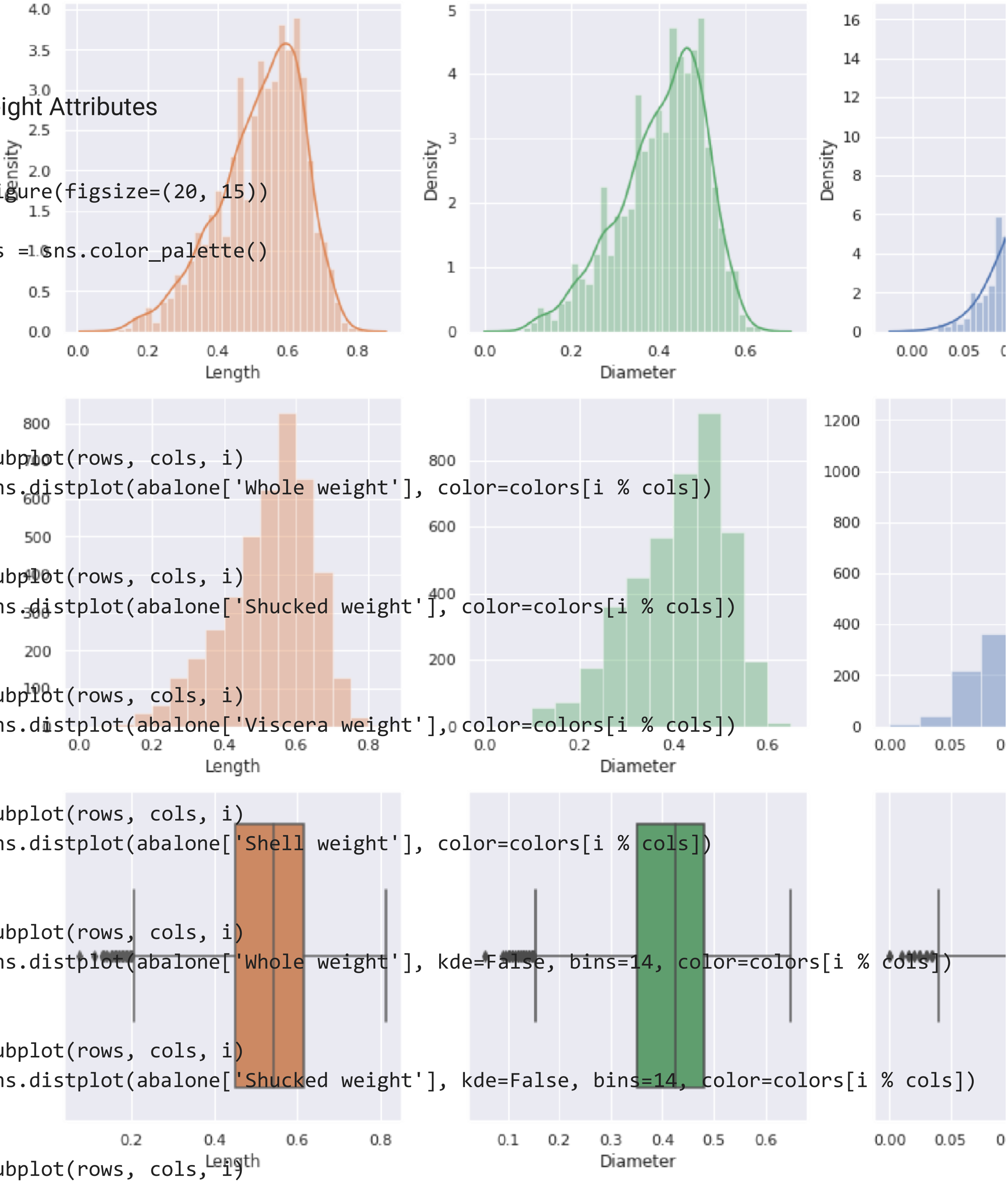
The analysis shows that the Ring attribute values ranges from 1 to 29 rings on an abalone specimen. However, the most frequent values of Rings are highly concentrated around the median of the distribution, so that, the 2nd and 3rd quartiles are de ned in a range of less than 1 std deviation. We observe that its possible to approximate the distribution of this attribute to a normal curve.

2) Size attributes

Here, we analyze the attributes that represents the dimensions of an abalone. These attributes are Length, Diameter and Height. For each of these attributes we will plot two histograms and their respective boxplot.

```
# removing outliers
abalone = abalone[abalone['Height'] < 0.4]

plt.figure(figsize=(15, 15)) colors =

sns.color_palette()

lines = 3
rows = 3 i
= 0

i += 1 plt.subplot(lines,
rows, i)
_ = sns.distplot(abalone['Length'], color=colors[i % 3])
     i += 1
plt.subplot(lines, rows, i)
_ = sns.distplot(abalone['Diameter'], color=colors[i % 3])

i += 1 plt.subplot(lines,
rows, i)
_ = sns.distplot(abalone['Height'], color=colors[i % 3])

i += 1 plt.subplot(lines,
rows, i)
_ = sns.distplot(abalone['Length'], kde=False, bins=np.arange(0.0, 0.9, 0.05), color=color
     i += 1
plt.subplot(lines, rows, i)
_ = sns.distplot(abalone['Diameter'], kde=False, bins=np.arange(0.0, 0.7, 0.05), color=col

i += 1
plt.subplot(lines, rows, i)
_ = sns.distplot(abalone['Height'], kde=False, bins=10, color=colors[i % 3])

i += 1 plt.subplot(lines,
rows, i)
_ = sns.boxplot(abalone['Length'], color=sns.color_palette()[i % 3])

i += 1 plt.subplot(lines,
rows, i) _ =
sns.boxplot(abalone['Diamet
er'], color=colors[i % 3])

i += 1
plt.subplot(lines, rows, i)
_ = sns.boxplot(abalone['Height'], color=colors[i % 3])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
```



3) Weight Attributes

```
plt.figure(figsize=(20, 15))

colors = sns.color_palette()


plt.subplot(rows, cols, i)
_ = sns.distplot(abalone['Whole weight'], color=colors[i % cols])


plt.subplot(rows, cols, i)
_ = sns.distplot(abalone['Shucked weight'], color=colors[i % cols])


plt.subplot(rows, cols, i)
_ = sns.distplot(abalone['Viscera weight'], color=colors[i % cols])


plt.subplot(rows, cols, i)
_ = sns.distplot(abalone['Shell weight'], color=colors[i % cols])


plt.subplot(rows, cols, i)
_ = sns.distplot(abalone['Whole weight'], kde=False, bins=14, color=colors[i % cols])


plt.subplot(rows, cols, i)
_ = sns.distplot(abalone['Shucked weight'], kde=False, bins=14, color=colors[i % cols])


plt.subplot(rows, cols, i)
```

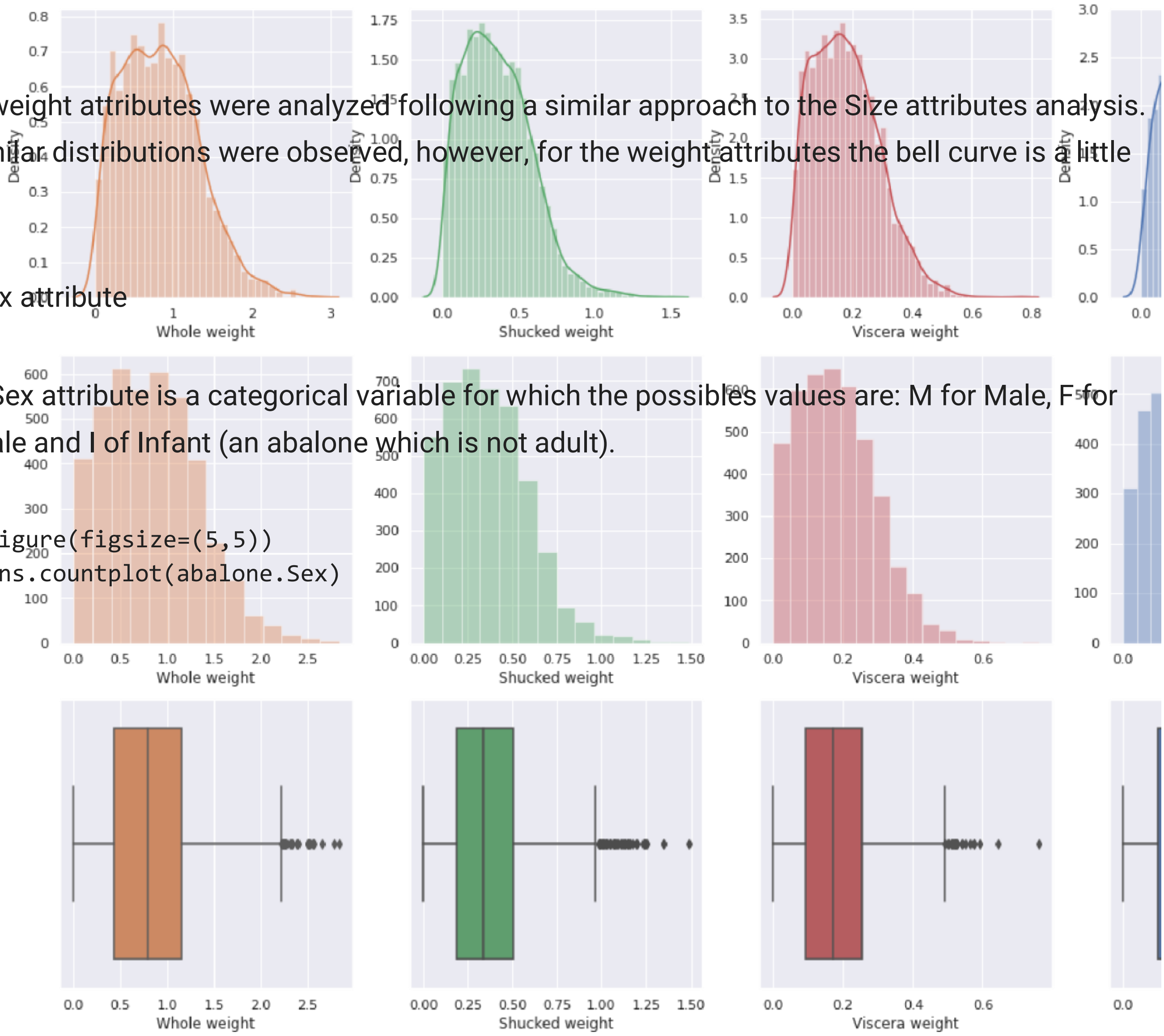
Analyzing the Height boxplot, we conclude that the high peak is formed due the presence of two

`FutureWarning` observations that lie far beyond the central positions of the distribution.

```
rows = 3
```

```
cols = 4


i = 0


i += 1


     i
+= 1



i += 1



i += 1



i += 1


     i
+= 1



i += 1

_ = sns.distplot(abalone['Viscera weight'], kde=False, bins=16, color=colors[i % cols])
i += 1 plt.subplot(rows,
cols, i)
_ = sns.distplot(abalone['Shell weight'], kde=False, bins=20, color=colors[i % cols])

i += 1
plt.subplot(rows, cols, i)
_ = sns.boxplot(abalone['Whole weight'], color=colors[i % cols])

i += 1 plt.subplot(rows,
cols, i)
_ = sns.boxplot(abalone['Shucked weight'], color=colors[i % cols])

i += 1
plt.subplot(rows, cols, i)
_ = sns.boxplot(abalone['Viscera weight'], color=colors[i % cols])

i += 1
plt.subplot(rows, cols, i)
_ = sns.boxplot(abalone['Shell weight'], color=colors[i % cols])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
FutureWarning
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
FutureWarning
```



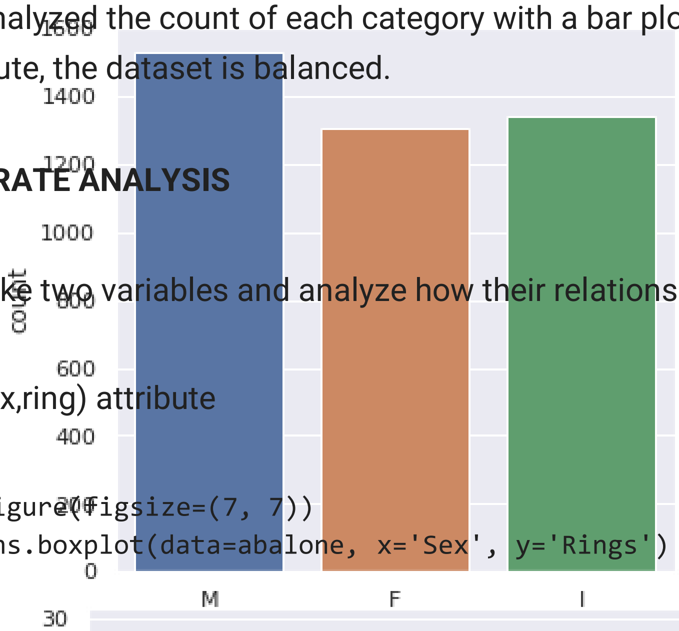
The weight attributes were analyzed following a similar approach to the Size attributes analysis. A similar distributions were observed, however, for the weight attributes the bell curve is a little larger.

4) Sex attribute

The Sex attribute is a categorical variable for which the possibles values are: M for Male, F for Female and I of Infant (an abalone which is not adult).

```
plt.figure(figsize=(5,5))
_ = sns.countplot(abalone.Sex)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
FutureWarning
```



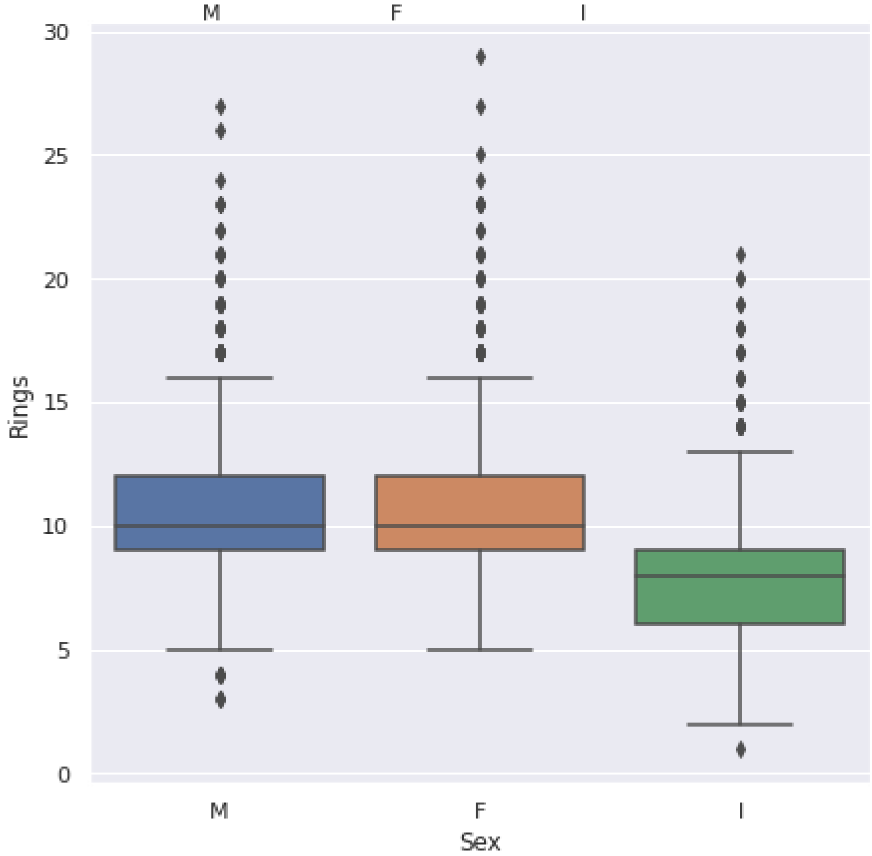
We analyzed the count of each category with a bar plot, and concluded that relative to this attribute, the dataset is balanced.

BIVARATE ANALYSIS

We take two variables and analyze how their relationship affects each other

1) (sex,ring) attribute

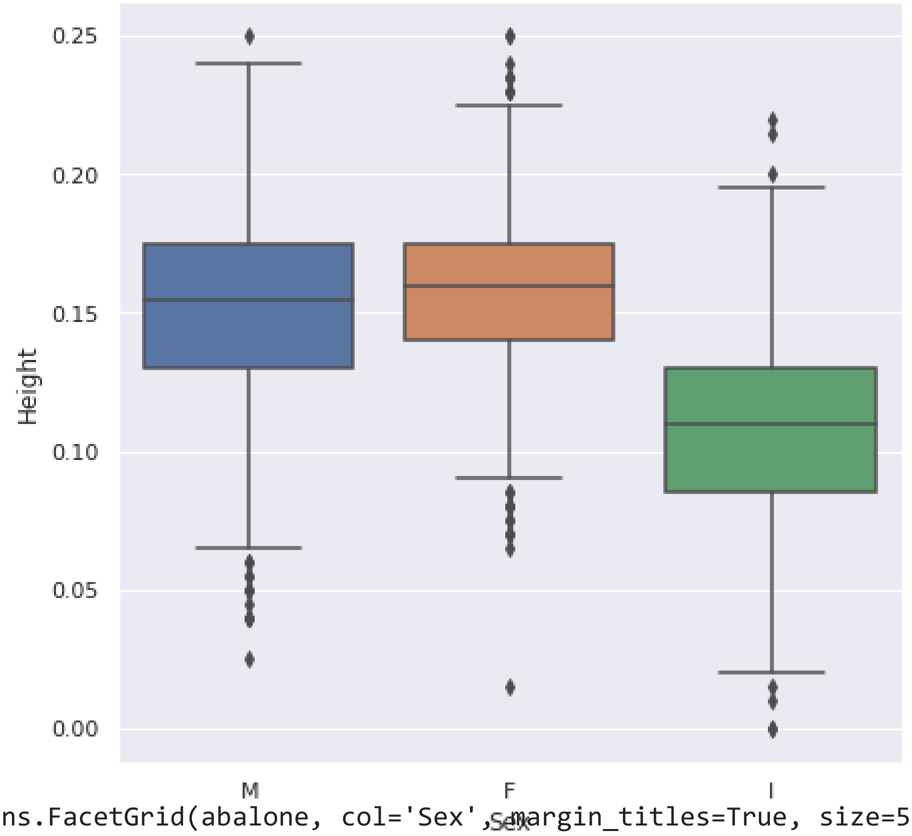
```
plt.figure(figsize=(7, 7))
_ = sns.boxplot(data=abalone, x='Sex', y='Rings')
```



We observe that the median of Rings for the I category is lower than the median for M and F categories.

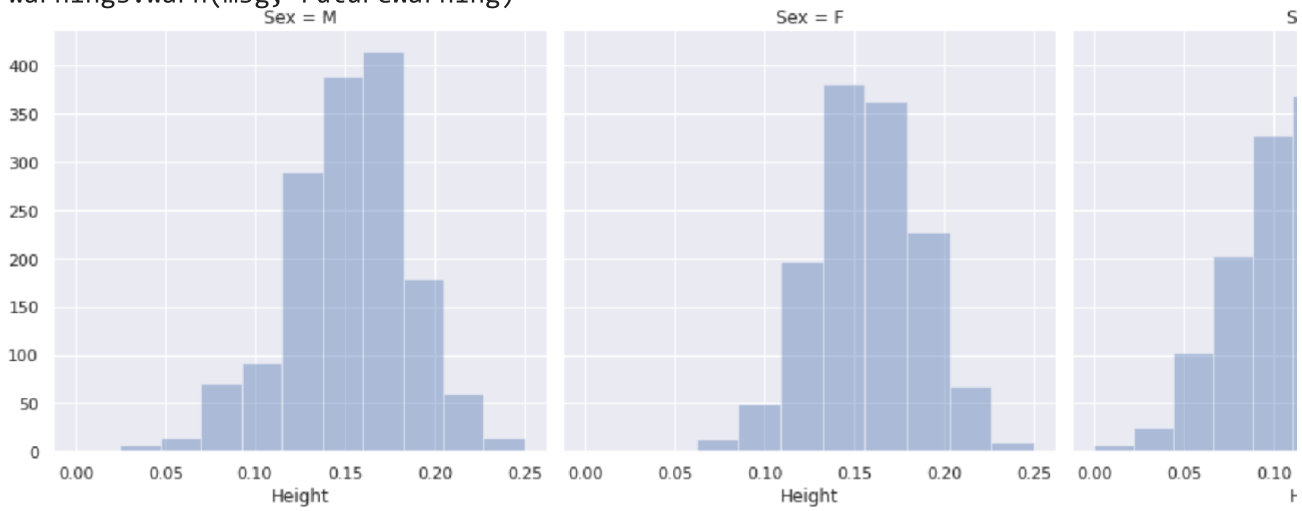
2) (Sex,height) attribute

```
plt.figure(figsize=(7, 7))
_ = sns.boxplot(data=abalone, x='Sex', y='Height')
```



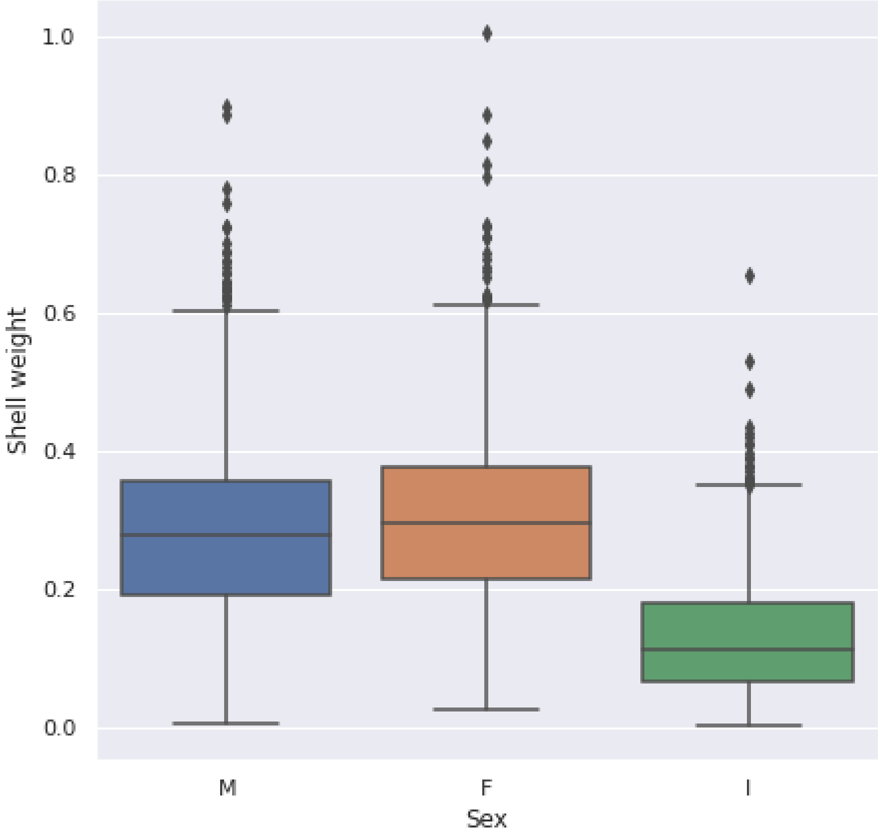

```
g = sns.FacetGrid(abalone, col='Sex', margin_titles=True, size=5)
_ = g.map(sns.distplot, 'Height', kde=False, bins=10)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:337: UserWarning: The `siz
warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
warnings.warn(msg, FutureWarning)
```



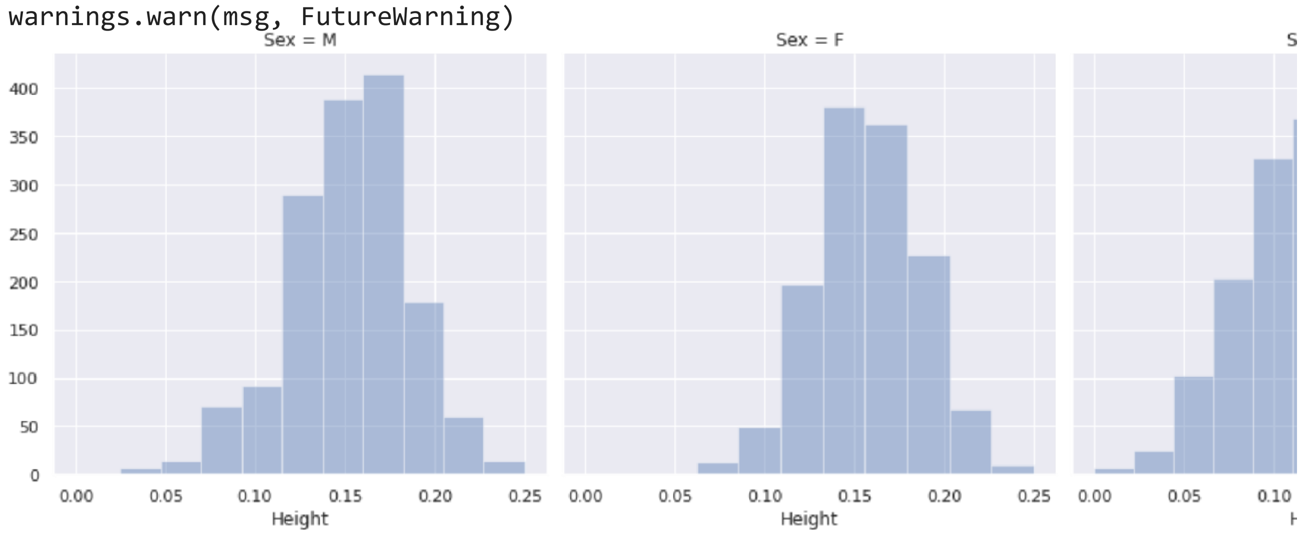
3) (Sex,shell weight) attribute

```
plt.figure(figsize=(7, 7))
_ = sns.boxplot(data=abalone, x='Sex', y='Shell weight')
```




```
g = sns.FacetGrid(abalone, col='Sex', margin_titles=True, size=5) _
= g.map(sns.distplot, 'Height', kde=False, bins=10)
```

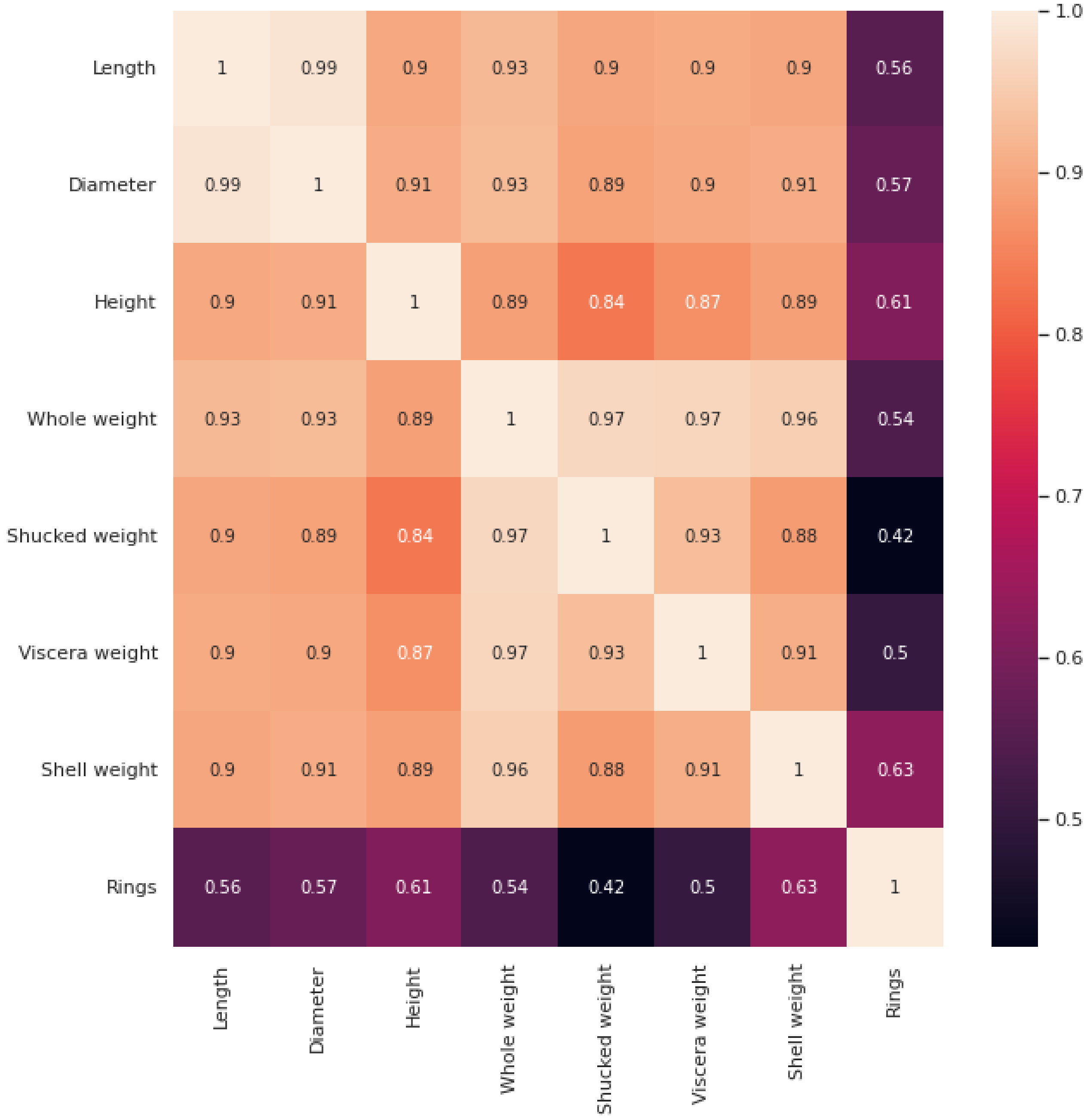
```
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:337: UserWarning: The `siz
warnings.warn(msg, UserWarning)
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:
warnings.warn(msg, FutureWarning)
```



MULTIVARIATE ANALYSIS

Correlation matrix in Heatmap:

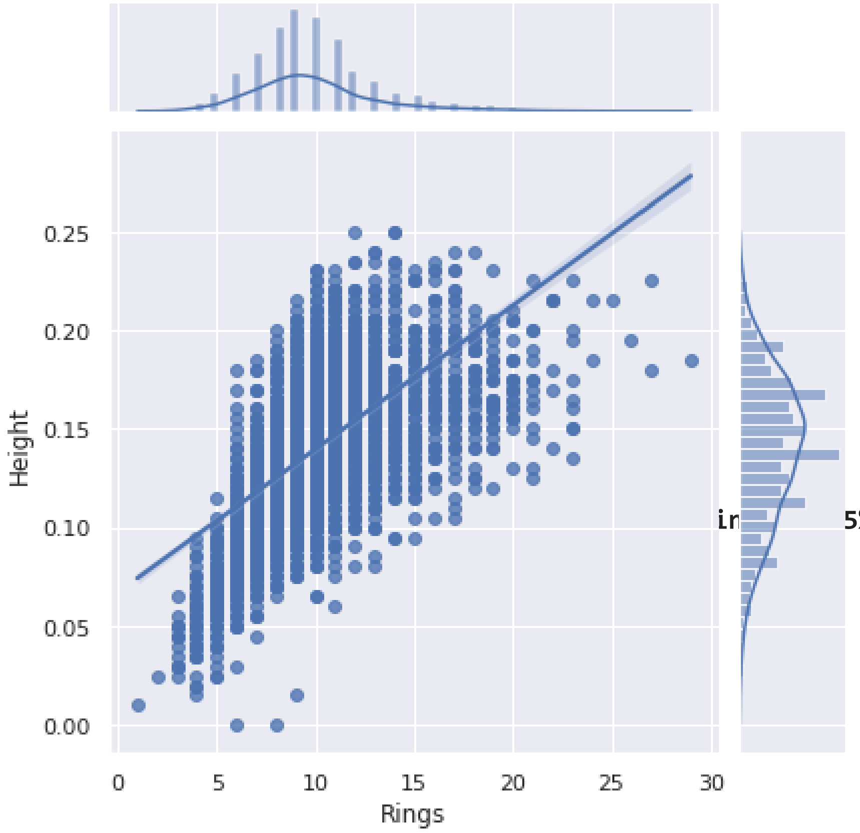
```
plt.figure(figsize=(10, 10)) corr
= abalone.corr()
_ = sns.heatmap(corr, annot=True)
```



Analyzing the correlation matrix, we notice that Height and Shell weight are the attributes that most correlates to Rings. Therefore, we concentrated the multivariate analysis on the correlation of these two attributes with Rings:

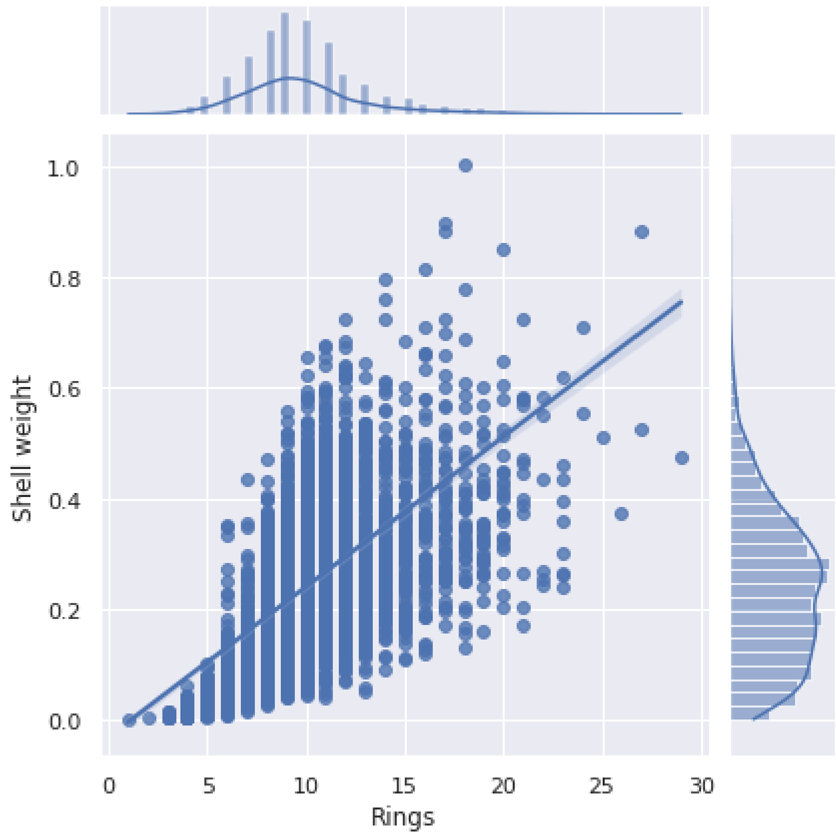
```
plt.figure(figsize=(20, 5))

_ = sns.jointplot(data=abalone, x='Rings', y='Height', kind='reg')
_ = sns.jointplot(data=abalone, x='Rings', y='Shell weight', kind='reg')
    <Figure size 1440x360 with 0 Axes>
```

For the lower values of Rings we have concentrated values of Height and Shell weight. As value of Rings increases, the scatterplot becames larger, and for the highest values of Rings it become disperse.



DESCRIPTIVE STATISTICS

```
abalone.describe().T
```



				ir	5%	50%	75%	max
Length	4175.0	0.523965	0.120084	0.0750	0.45000	0.5450	0.61500	0.8150
Diameter	4175.0	0.407856	0.099230	0.0550	0.35000	0.4250	0.48000	0.6500
Height	4175.0	0.139189	0.038489	0.0000	0.11500	0.1400	0.16500	0.2500
Whole weight	4175.0	0.828468	0.490027	0.0020	0.44150	0.7995	1.15300	2.8255
Shucked weight	4175.0	0.359195	0.221713	0.0010	0.18600	0.3360	0.50175	1.4880

Viscera weight	4175.0	0.180536	0.109534	0.0005	0.09325	0.1710	0.25275	0.7600
Shell weight	4175.0	0.238791	0.139162	0.0015	0.13000	0.2340	0.32875	1.0050
Rings	4175.0	9.934132	3.224802	1.0000	8.00000	9.0000	11.00000	29.0000

HANDLING WITH MISSING DATA

To check missing values, we can use isnull() or notnull()

To replace values in missing cell, we can use llna(),replace() and interpolate()

```
df = pd.DataFrame(abalone) df.isnull()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
4172	False	False	False	False	False	False	False
4173	False	False	False	False	False	False	False
4174	False	False	False	False	False	False	False
4175	False	False	False	False	False	False	False
4176	False	False	False	False	False	False	False

4175 rows × 9 columns isnull() -

returns true for NULL values

notnull() - returns false for NULL values(NaN)

```
df.fillna(0)
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight
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
...	...	...	...	...	...	...	...
4172	F	0.565	0.450	0.165	0.8870	0.3700	0.2390
4173	M	0.590	0.440	0.135	0.9660	0.4390	0.2145
4174	M	0.600	0.475	0.205	1.1760	0.5255	0.2875
4175	F	0.625	0.485	0.150	1.0945	0.5310	0.2610
4176	M	0.710	0.555	0.195	1.9485	0.9455	0.3765

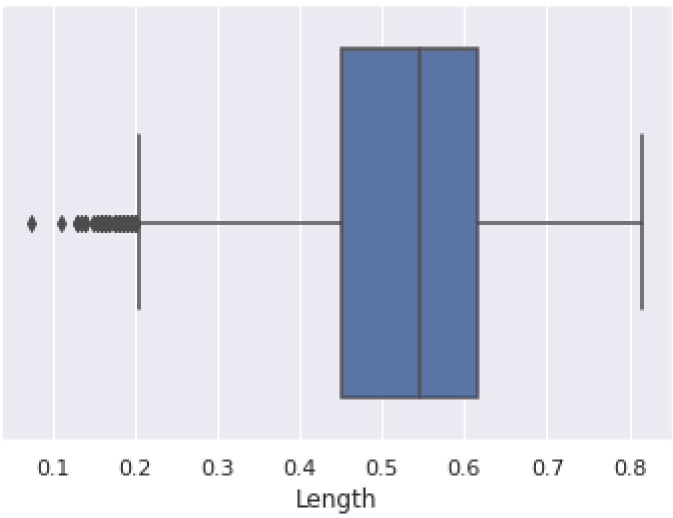
4175 rows × 9 columns

Replacing the missing values with 0 using llna **OUTLIERS**

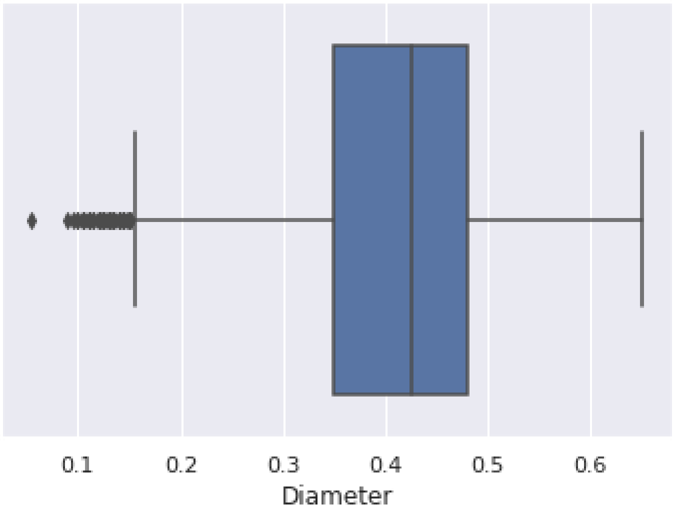
IN EACH ATTRIBUTES

```
sns.boxplot(df['Length'],data=df)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f8942052ed0>
```

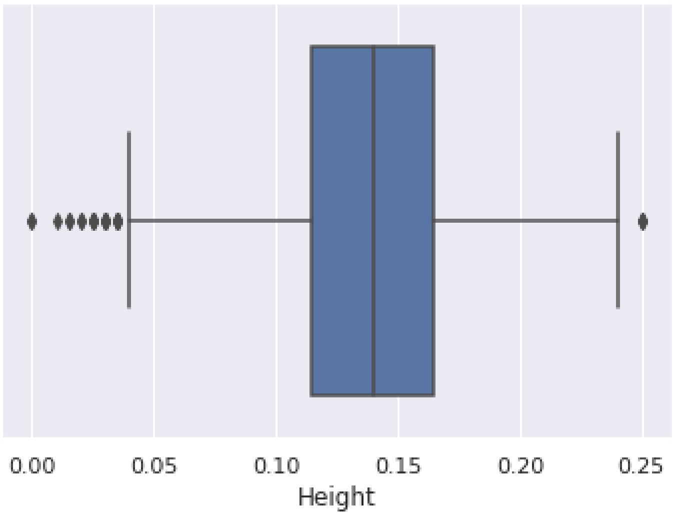



```
sns.boxplot(df['Diameter'],data=df) /usr/local/lib/python3.7/dist-
packages/seaborn/_decorators.py:43: FutureWarning: Pass   FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f89420eb490>
```




```
sns.boxplot(df['Height'],data=df)
```

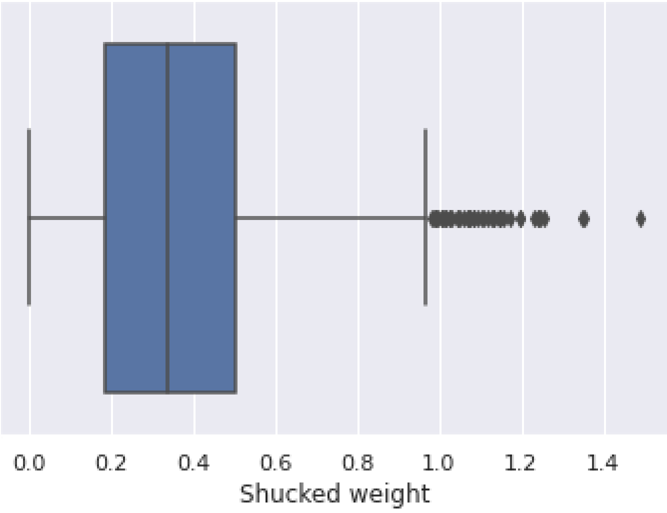
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f8942a5d090>
```




```
sns.boxplot(df['Whole weight'],data=df) /usr/local/lib/python3.7/dist-
packages/seaborn/_decorators.py:43: FutureWarning: Pass   FutureWarning
    <matplotlib.axes._subplots.AxesSubplot at 0x7f8941fc6650>
```

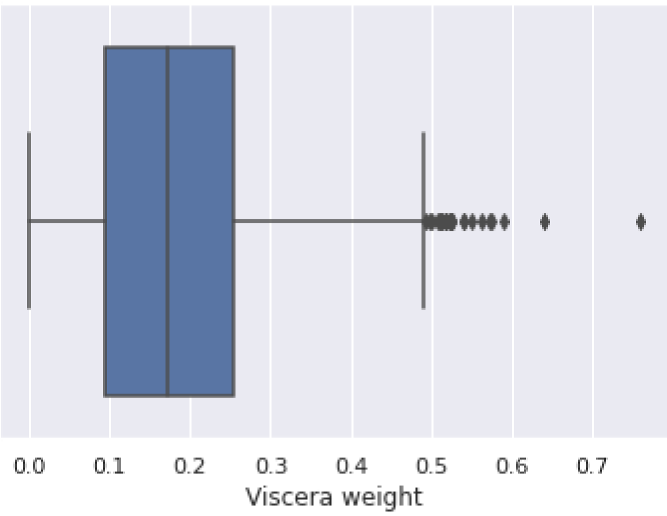
```
sns.boxplot(df['Shucked weight'],data=df)
```

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f89421a0290>



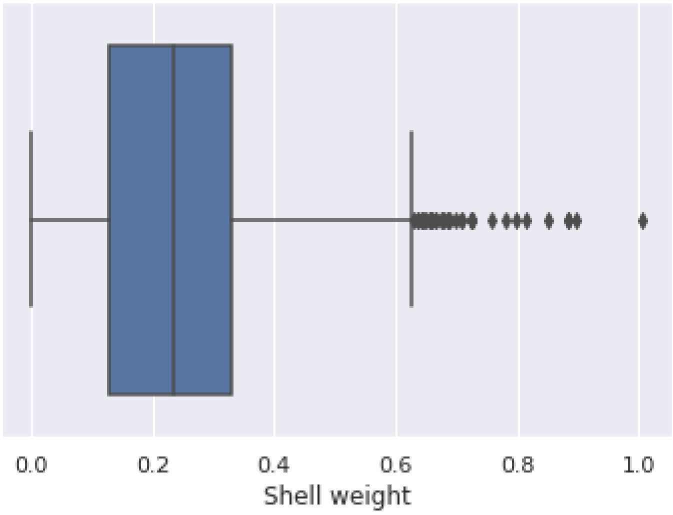

```
sns.boxplot(df['Viscera weight'],data=df)
```

/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f8941fadd10>



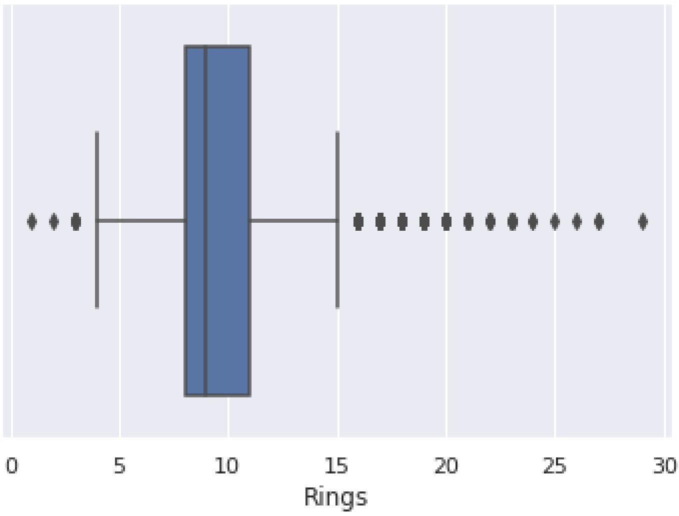

```
sns.boxplot(df['Shell weight'],data=df) /usr/local/lib/python3.7/dist-
packages/seaborn/_decorators.py:43: FutureWarning: Pass   FutureWarning
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f8942285a90>




```
sns.boxplot(df['Rings'],data=df)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f8942222b10>
```




```
Q1 = abalone.quantile(0.25)
Q3 = abalone.quantile(0.75)
IQR = Q3-Q1 print(IQR)
```

```
Length            0.16500
Diameter          0.13000
Height            0.05000
Whole weight      0.71150
Shucked weight    0.31575
Viscera weight    0.15950
Shell weight      0.19875 Rings
3.00000 dtype: float64
```

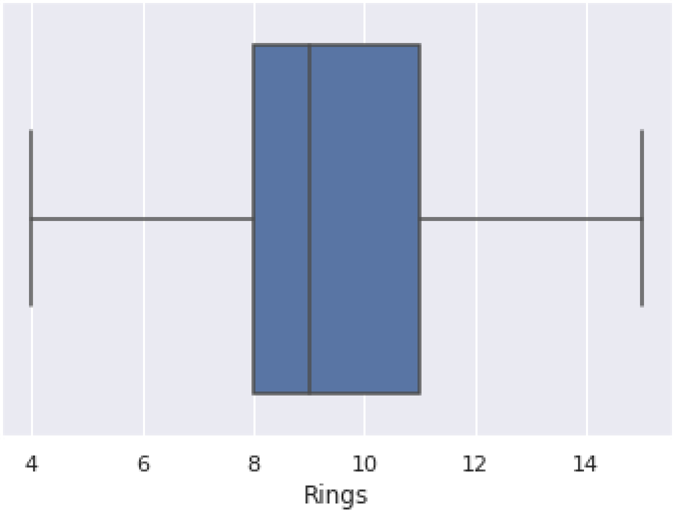
Removing outliers using IQR

```
abalone = abalone[~((abalone < (Q1 - 1.5 * IQR)) |(abalone > (Q3 + 1.5 * IQR))).any(axis=1
abalone.shape
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: Automa
  """Entry point for launching an IPython kernel. (3781,
9)
```

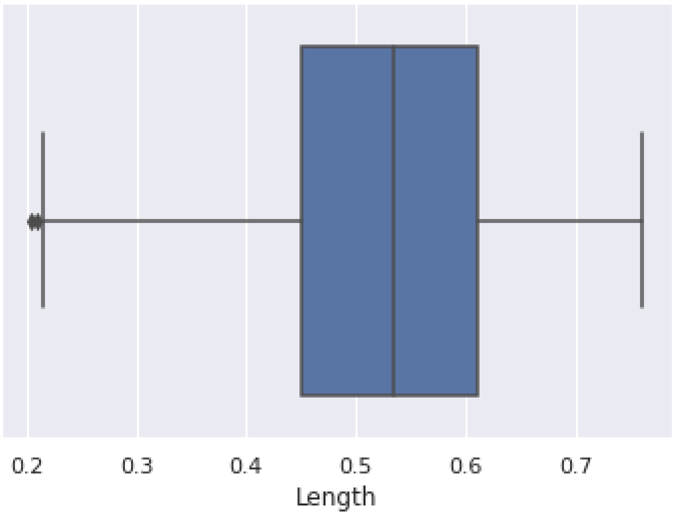
```
sns.boxplot(abalone['Rings'],data=abalone)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f8942082c90>
```




```
sns.boxplot(abalone['Length'],data=abalone)
```

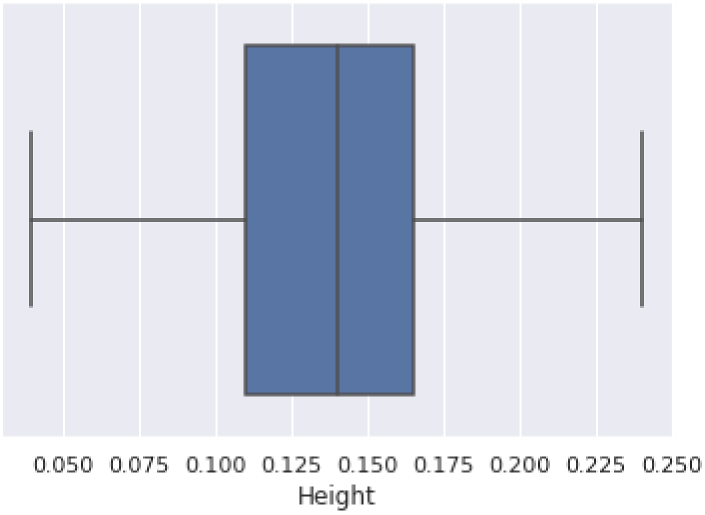
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f894046d9d0>
```



TEAM ID: PNT2022TMID53225

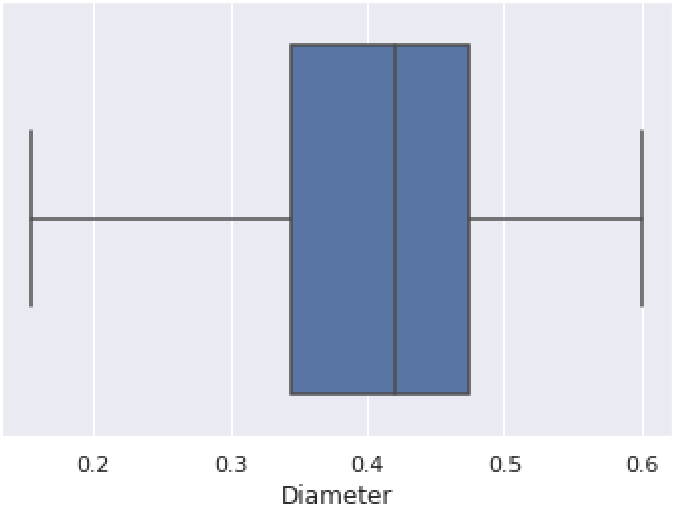
```
sns.boxplot(abalone['Height'],data=abalone)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f89429c1390>
```

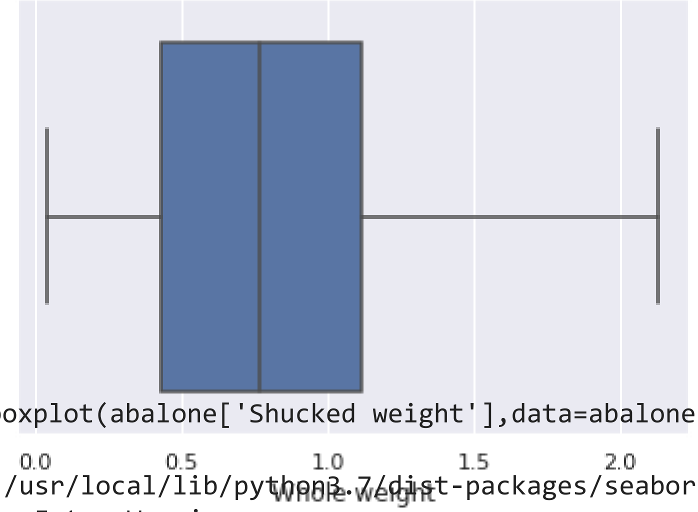



```
sns.boxplot(abalone['Diameter'],data=abalone)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f89421ae7d0>
```

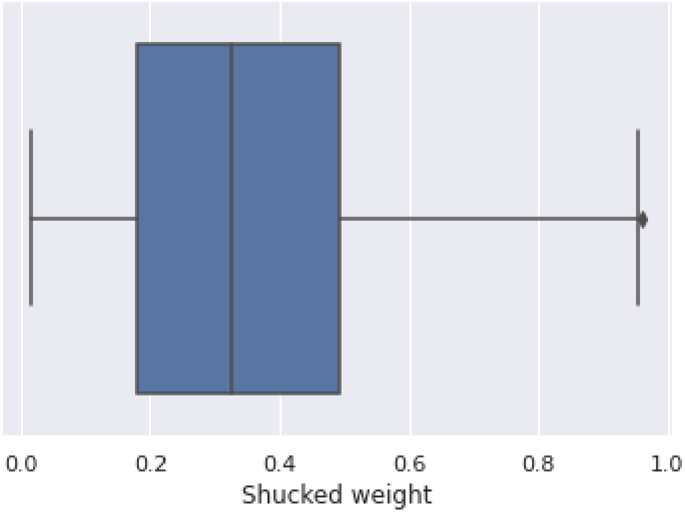



```
sns.boxplot(abalone['Whole weight'],data=abalone) /usr/local/lib/python3.7/dist-
packages/seaborn/_decorators.py:43: FutureWarning: Pass   FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f894040de90>
```




```
sns.boxplot(abalone['Shucked weight'],data=abalone)
```

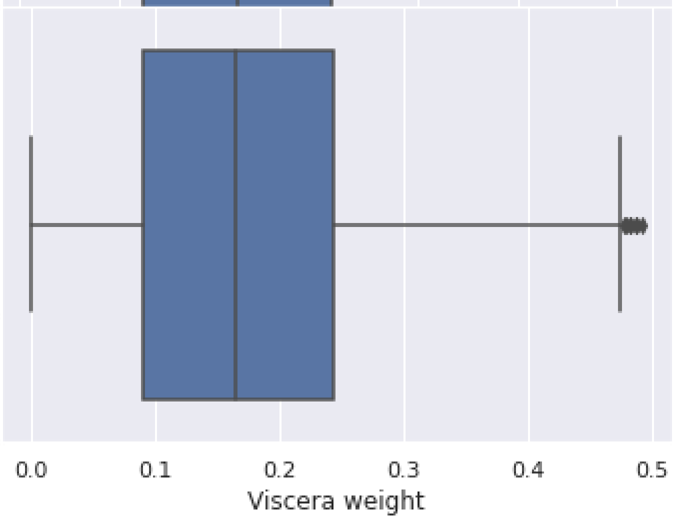
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f8940257650>
```




```
sns.boxplot(abalone['Shell weight'],data=abalone) /usr/local/lib/python3.7/dist-
packages/seaborn/_decorators.py:43: FutureWarning: Pass   FutureWarning
sns.boxplot(abalone['Viscera weight'],data=abalone)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8940424f50>
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f8940424f90>
```



After removing the outliers, the above dataset has received.

TEAM ID: PNT2022TMID53225

LABEL ENCODING OF CATEGORICAL DATA

```
le=LabelEncoder()
abalone['Sex']=le.fit_transform(abalone['Sex'])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: SettingWithCopyWarnin
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
```

```
abalone
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight
0	2	0.455	0.365	0.095	0.5140	0.2245	0.1010
1	2	0.350	0.265	0.090	0.2255	0.0995	0.0485
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415
3	2	0.440	0.365	0.125	0.5160	0.2155	0.1140
4	1	0.330	0.255	0.080	0.2050	0.0895	0.0395
...	...	...	...	...	...	...	...
4172	0	0.565	0.450	0.165	0.8870	0.3700	0.2390
4173	2	0.590	0.440	0.135	0.9660	0.4390	0.2145
4174	2	0.600	0.475	0.205	1.1760	0.5255	0.2875
4175	0	0.625	0.485	0.150	1.0945	0.5310	0.2610
4176	2	0.710	0.555	0.195	1.9485	0.9455	0.3765

3781 rows × 9 columns

Above we have encoded the categorical data "Sex" as 0 or 1 or 2 based on M or F or I

. Spliting the Data into dependent and Independent Variables

```
X = abalone.iloc[:, :-1].values
y = abalone.iloc[:, -1].values
```

9. Scaling independent variables

```
scaler = StandardScaler() scaler.fit(abalone)
```

```
StandardScaler()
```

10. Spliting training and test data

```
train_X,val_X,train_y,val_y = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

```
print("Shape of Training X :",train_X.shape) print("Shape
of Validation X :",val_X.shape)
```

```
Shape of Training X : (3024, 8)
Shape of Validation X : (757, 8)
```

```
print("Shape of Training y :",train_y.shape) print("Shape
of Validation y :",val_y.shape)
```

```
Shape of Training y : (3024,)
Shape of Validation y : (757,)
```

LINEAR REGRESSION

```
lr = LinearRegression() lr.fit(train_X,train_y)
```

```
LinearRegression()
```

```
%%time y_pred_val_lr = lr.predict(val_X) print('MAE on Validation set
:',metrics.mean_absolute_error(val_y, y_pred_val_lr)) print("\n")
print('MSE on Validation set :',metrics.mean_squared_error(val_y, y_pred_val_lr))
print("\n") print('RMSE on Validation set :',np.sqrt(metrics.mean_absolute_error(val_y,
y_pred_val_lr) print("\n") print('R2 Score on Validation set :',metrics.r2_score(val_y,
y_pred_val_lr)) print("\n")
```

```
MAE on Validation set : 1.2719689486359298


MSE on Validation set : 2.7606215450501024


RMSE on Validation set : 1.127816008325795


R2 Score on Validation set : 0.5119499107890585


CPU times: user 9.52 ms, sys: 1.03 ms, total: 10.6 ms Wall
time: 9.65 ms
```

SUPPORT VECTOR MACHINE

```
svm = SVR()
svm.fit(train_X,train_y)
```

```
SVR()
```

```
%%time y_pred_val_svm = svm.predict(val_X) print('MAE on Validation set
:',metrics.mean_absolute_error(val_y, y_pred_val_svm)) print("\n")
print('MSE on Validation set :',metrics.mean_squared_error(val_y, y_pred_val_svm))
print("\n") print('RMSE on Validation set :',np.sqrt(metrics.mean_absolute_error(val_y,
y_pred_val_svm print("\n") print('R2 Score on Validation set :',metrics.r2_score(val_y,
y_pred_val_svm)) print("\n")
```

```
MAE on Validation set : 1.2208952787270895
MSE on Validation set : 2.7012620714060267


RMSE on Validation set : 1.1049413010323623


R2 Score on Validation set : 0.5224440679687887


CPU times: user 152 ms, sys: 28 µs, total: 152 ms Wall
time: 153 ms
```

DECISION TREE REGRESSOR

```
dc = DecisionTreeRegressor(random_state = 0) dc.fit(train_X,train_y)
```

```
DecisionTreeRegressor(random_state=0)
```

```
%%time
y_pred_val_dc = dc.predict(val_X)
print('MAE on Validation set :',metrics.mean_absolute_error(val_y, y_pred_val_dc))
print("\n") print('MSE on Validation set :',metrics.mean_squared_error(val_y,
y_pred_val_dc)) print("\n")
print('RMSE on Validation set :',np.sqrt(metrics.mean_absolute_error(val_y, y_pred_val_dc)
print("\n")
print('R2 Score on Validation set :',metrics.r2_score(val_y, y_pred_val_dc)) print("\n")
```

```
MAE on Validation set : 1.6393659180977542


MSE on Validation set : 4.88110964332893


RMSE on Validation set : 1.2803772561623212


R2 Score on Validation set : 0.13706896870869845


CPU times: user 1.94 ms, sys: 0 ns, total: 1.94 ms Wall
time: 1.95 ms
```

OVERVIEW OF R2 SCORES OF ALL MODELS

```
print('Logistic Regression R2 Score on Validation set :',metrics.r2_score(val_y, y_pred_va
print('SVR R2 Score on Validation set :',metrics.r2_score(val_y, y_pred_val_svm))
print('Decision Tree Regressor R2 Score on Validation set :',metrics.r2_score(val_y, y_pre
```

```
Logistic Regression R2 Score on Validation set : 0.5119499107890585
SVR R2 Score on Validation set : 0.5224440679687887
Decision Tree Regressor R2 Score on Validation set : 0.13706896870869845
```