```
import numpy as np
import pandas as pd
import seaborn as sns
```

load dataset

```
df=pd.read_csv("/content/abalone.csv")
```

```
df.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

perform statistics values

```
df.describe()
```

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	41
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	

```
df.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   Rings           4177 non-null   int64
dtypes: float64(7), int64(1), object(1)
memory usage: 293.8+ KB
```

check missing values

```
df.isna().sum()
```

```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
Rings             0
dtype: int64
```

```
df.isna().sum().sum()
```

```
0
```

check catogorical columns

```
df._get_numeric_data()
```

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500	15
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9

```
df.shape
```

```
(4177, 9)
```

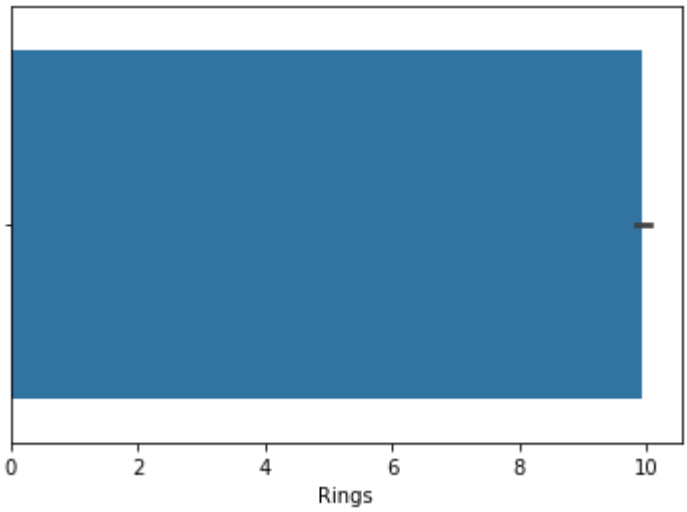
...	...	...	...	...	...	...	...	...

Univariant analysis

4174	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9

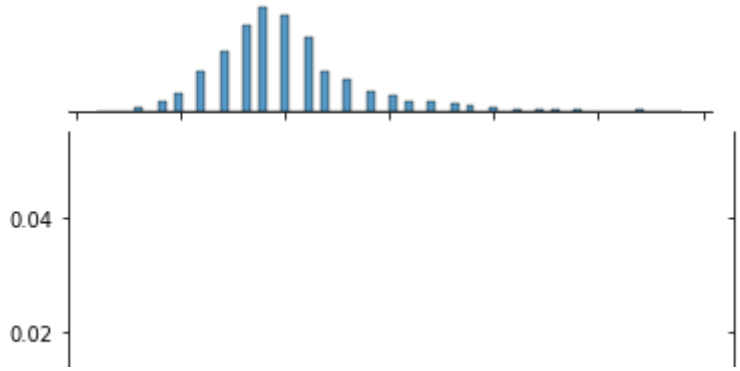
```
sns.barplot(df.Rings)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fa5de716750>
```




```
sns.jointplot(df.Rings)
```

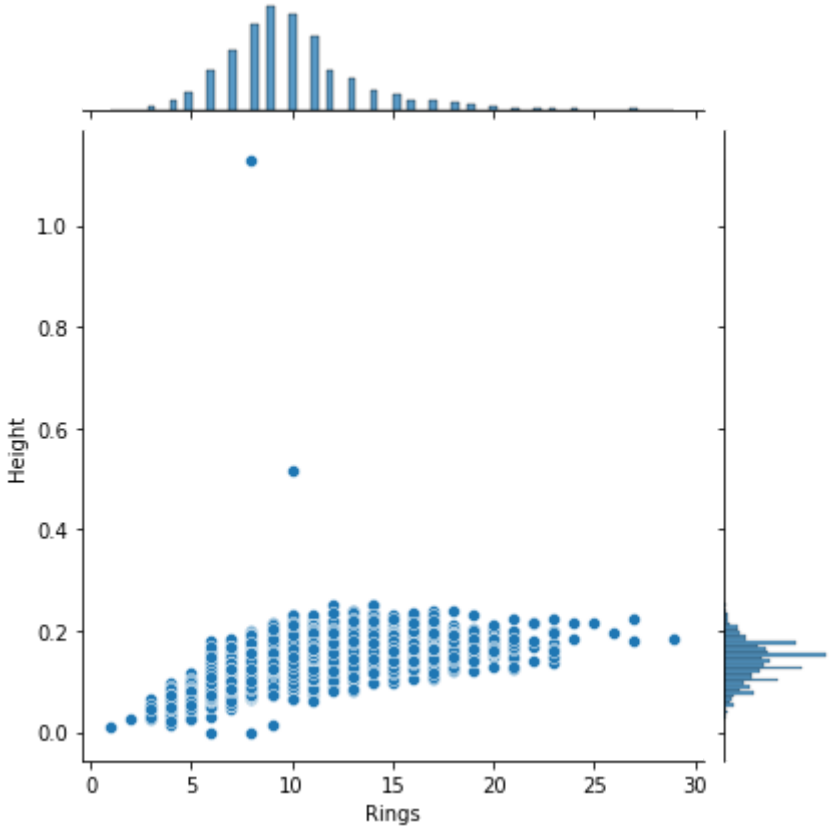
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<seaborn.axisgrid.JointGrid at 0x7fa5de6c8510>
```



Bivariant analysis

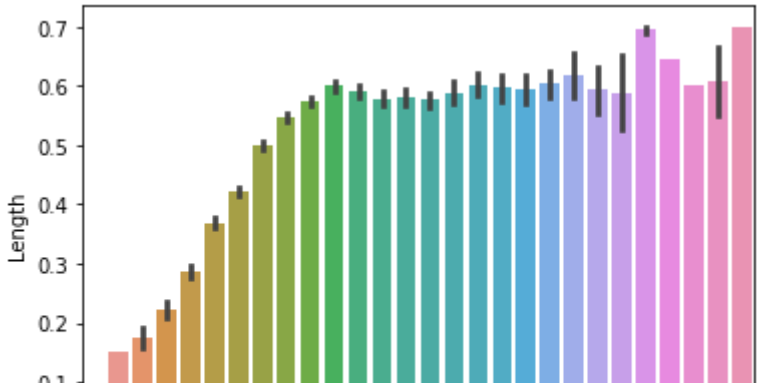
```
sns.jointplot(df.Rings,df.Height)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<seaborn.axisgrid.JointGrid at 0x7fa5de7a8850>
```




```
sns.barplot(df.Rings,df.Length)
```

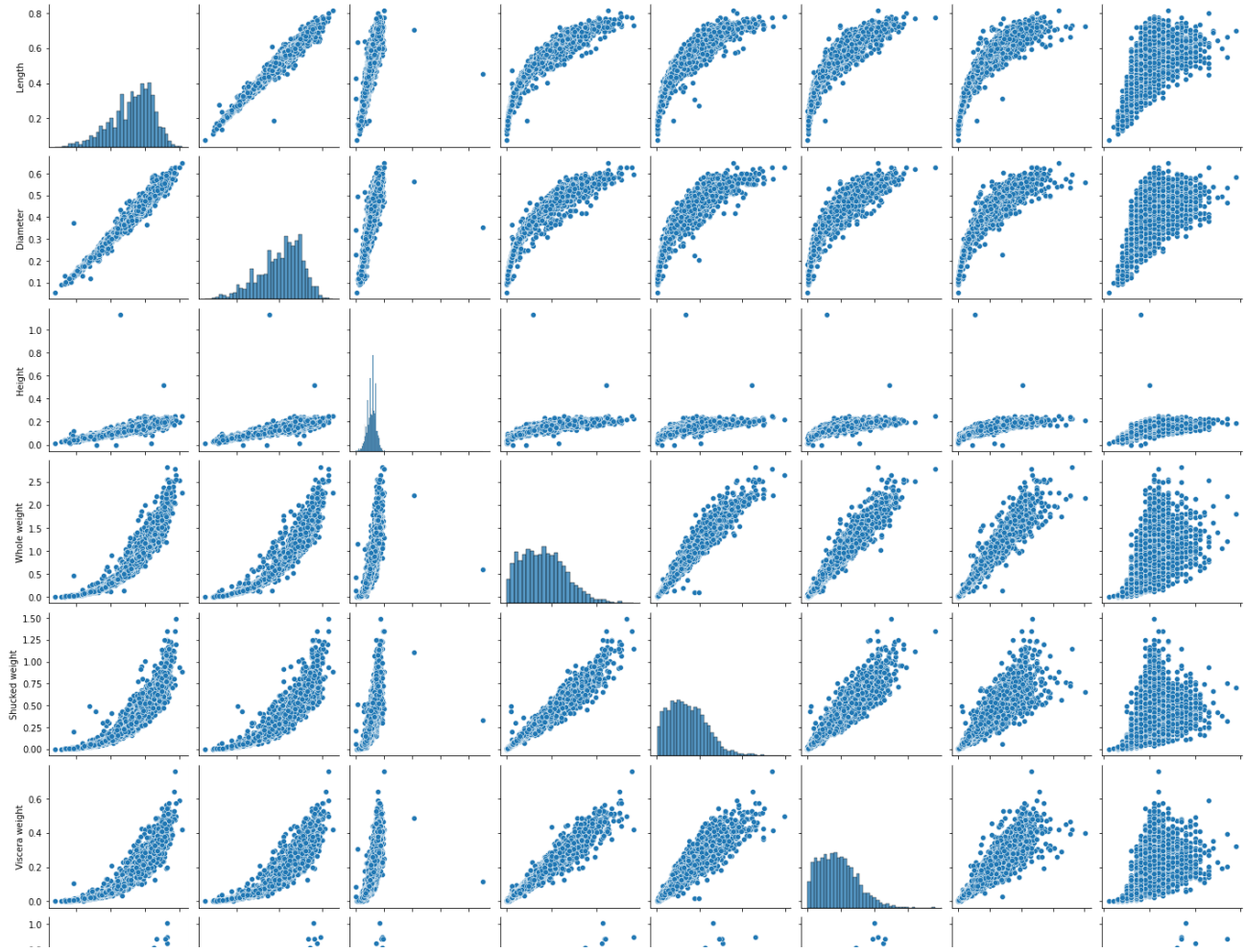
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7fa5de0d62d0>
```



Multivariant analysis

```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7fa5dc6ec1d0>
```



input varaible

```
from sklearn.preprocessing import MinMaxScaler
scalar=MinMaxScaler()
df_new=df.iloc[:, :-1]
df_new1=df_new.iloc[:, 1:]
```

```
df_new1
```

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550

split depandant and independant variable

4172	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490

```
x=df_new1
y=df['Rings']
```

split test and train data

```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

build KNN model

```
from sklearn.neighbors import KNeighborsClassifier
knn= KNeighborsClassifier()
```

predict the data

```
def knn_alg(X_train, y_train, X_test, y_test, N):
    knn = KNeighborsClassifier(n_neighbors=N)
    knn.fit = (X_train, y_train)

    try:
        knn.predict(X_test)
    except NotFittedError as e:
        print(repr(e))
```

evaluate our model

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

```
accuracy_score(y_test,pred)
confusion_matrix(y_test,pred)
```

+ Code + Text

Colab paid products - Cancel contracts here