ABALONE DATASET IBM ASSIGNMENT 4

IMPORTING LIBRARIES

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

UPLOADING

DATASET

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

```
df.head(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

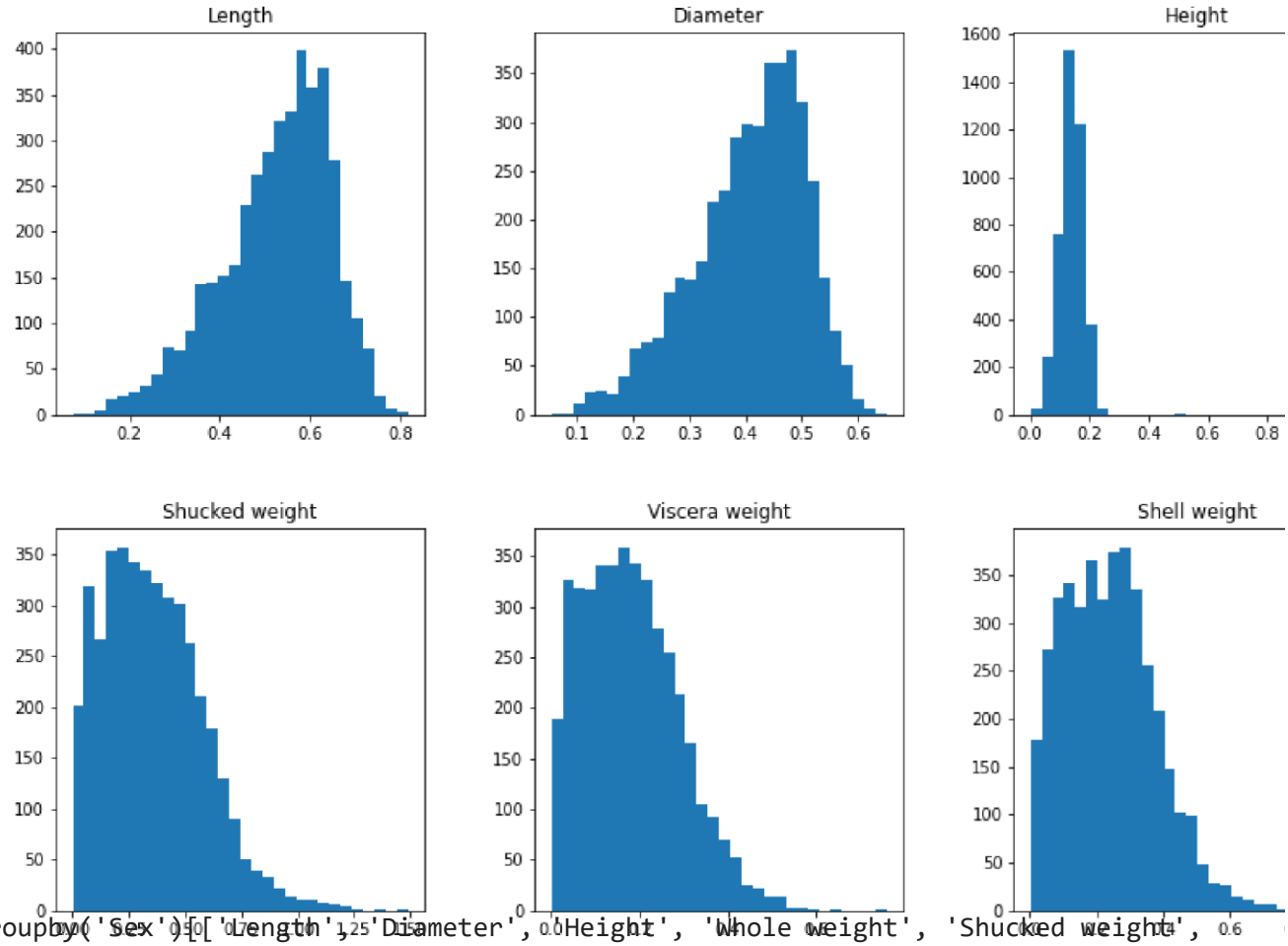
ADDING AGE ATTRIBUTE USING RING ATTRTIBUTE

```
df['age'] = df['Rings']+1.5 df
= df.drop('Rings', axis = 1)
```

UNIVARIATE ANALYSIS

```
df.hist(figsize=(20,10), grid=False, layout=(2, 4), bins = 30)
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fe3fa318610>,
<matplotlib.axes._subplots.AxesSubplot object at 0x7fe3fa2d79d0>,
            <matplotlib.axes._subplots.AxesSubplot object at 0x7fe3fa294210>,
            <matplotlib.axes._subplots.AxesSubplot object at 0x7fe3fa24b6d0>],
           [<matplotlib.axes._subplots.AxesSubplot object at 0x7fe3fa201bd0>,
            <matplotlib.axes._subplots.AxesSubplot object at 0x7fe3fa1c3110>,
            <matplotlib.axes._subplots.AxesSubplot object at 0x7fe3fa179690>,
    <matplotlib.axes._subplots.AxesSubplot object at 0x7fe3fa130ad0>]],
    dtype=object)
```



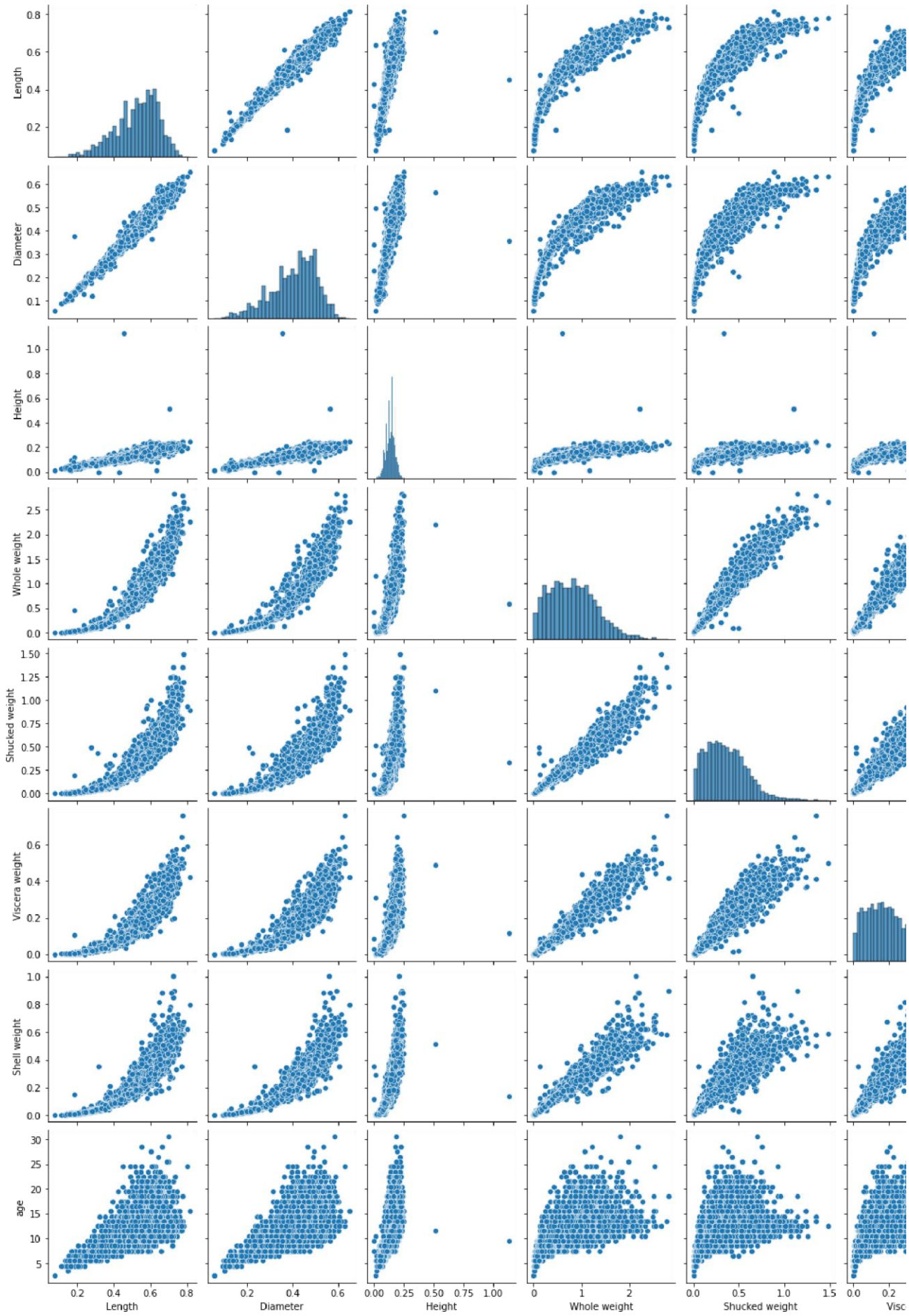

```
df.groupby('Sex')[['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
       'Viscera weight', 'Shell weight', 'age']].mean().sort_values('age')
```

Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shel
I	0.427746	0.326494	0.107996	0.431363	0.191035	0.092010	
M	0.561391	0.439287	0.151381	0.991459	0.432946	0.215545	
F	0.579093	0.454732	0.158011	1.046532	0.446188	0.230689	

Bi-Variate Analysis & Multi-Variate Analysis

```
numerical_features = df.select_dtypes(include = [np.number]).columns
sns.pairplot(df[numerical_features])
```

```
<seaborn.axisgrid.PairGrid at 0x7fe3fa53f5d0>
```


Length
Diameter
Height
Whole weight
Shucked weight
Viscera weight
Shell weight
age
Length
Diameter
Height
Whole weight
Shucked weight
Visc

DESCRIPTIVE STATISTICS

```
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	

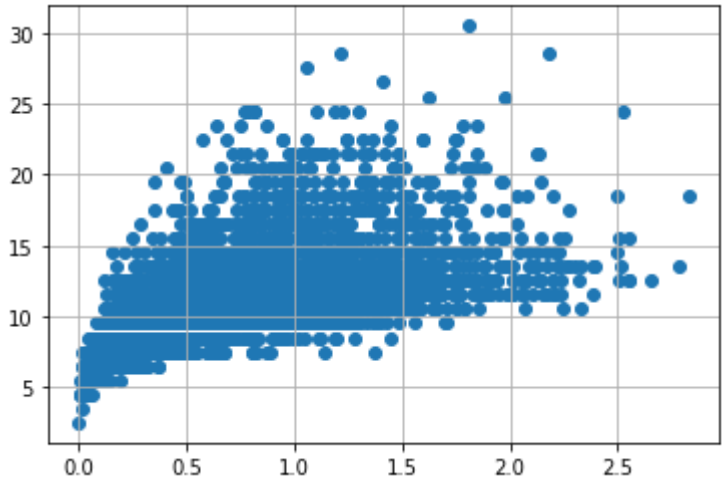
CHECKING MISSING /NULL VALUES

```
df.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
```

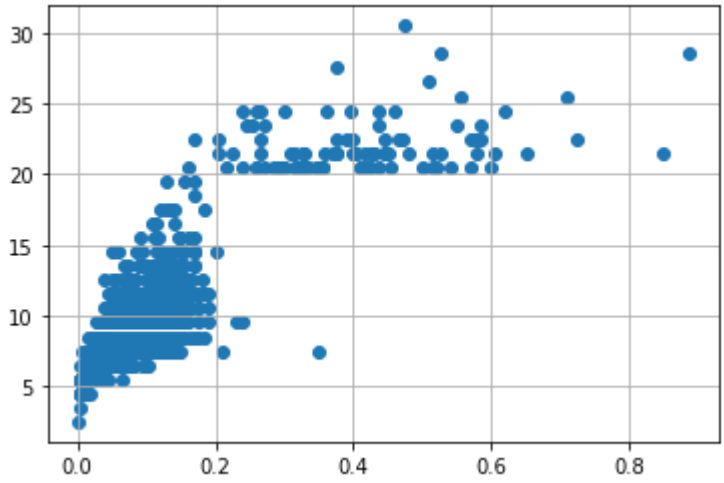
FIND AND REPLACE OUTLIERS

```
df = pd.get_dummies(df) dummy_data
 = df.copy() var = 'Whole weight'
plt.scatter(x = df[var], y =
df['age'],) plt.grid(True)
```



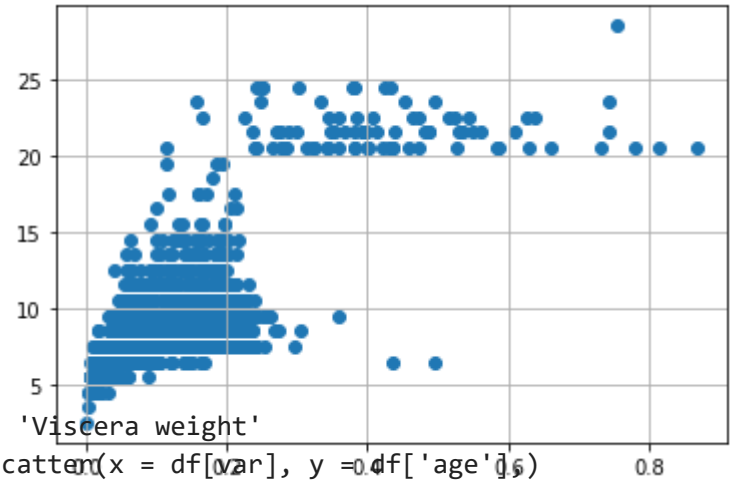

```
df.drop(df[(df['Whole weight']> 0.5) & (df['age'] < 20)].index, inplace=True)
df.drop(df[(df['Whole weight']<0.5) & (df['age'] > 25)].index, inplace=True)
```

```
var = 'Shell weight' plt.scatter(x = df[var], y = df['age'],) plt.grid(True)
#Outliers removal df.drop(df[(df['Shell weight']> 0.6) & (df['age'] <
25)].index, inplace=True) df.drop(df[(df['Shell weight']<0.8) & (df['age'] >
25)].index, inplace=True)
```

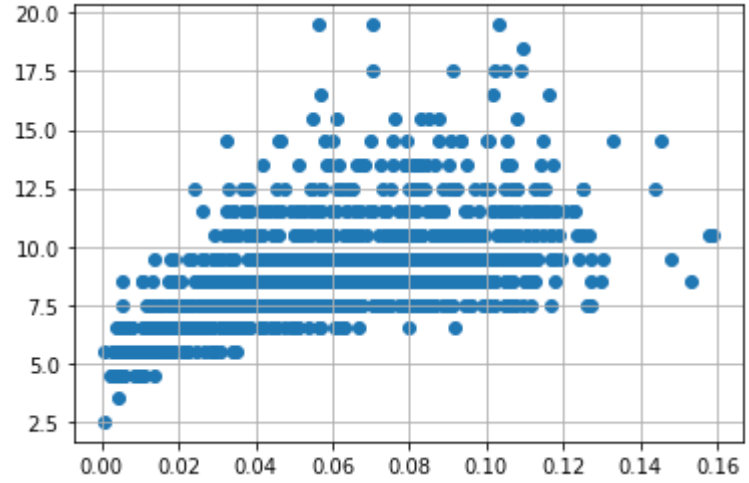



```
var = 'Shucked weight' plt.scatter(x =
df[var], y = df['age'],) plt.grid(True)
```

```
#Outlier removal
df.drop(df[(df['Shucked weight']>= 1) & (df['age'] < 20)].index, inplace=True)
df.drop(df[(df['Shucked weight']<1) & (df['age'] > 20)].index, inplace=True)
```

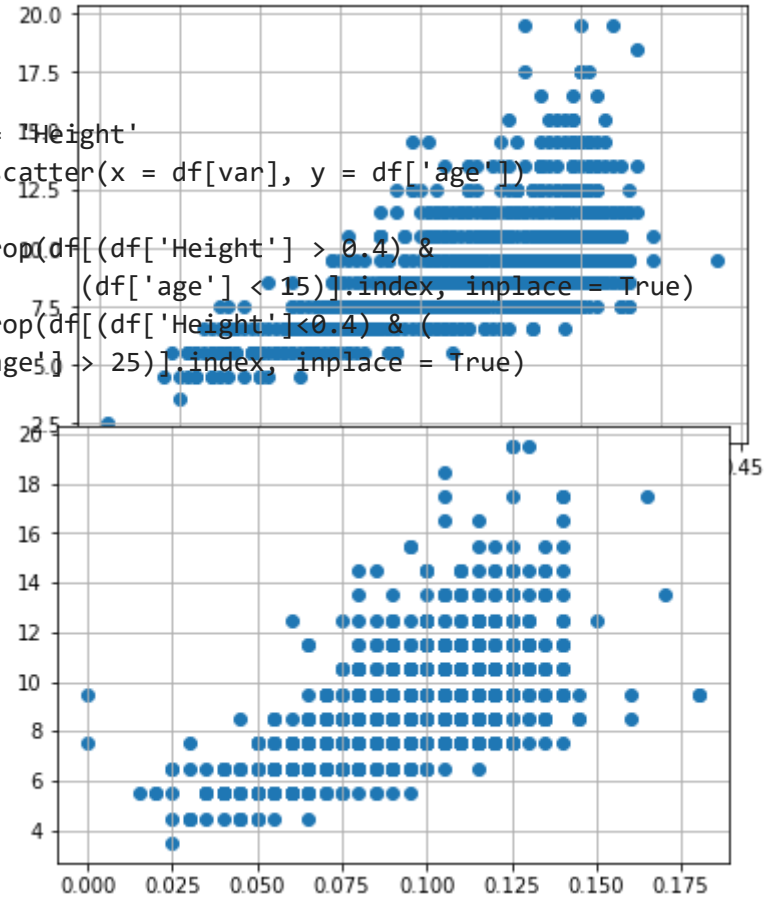



```
var = 'Viscera weight'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True) #
outliers removal
df.drop(df[(df['Viscera weight']> 0.5) & (df['age'] < 20)].index, inplace=True)
df.drop(df[(df['Viscera weight']<0.5) & (df['age'] > 25)].index, inplace=True)
```

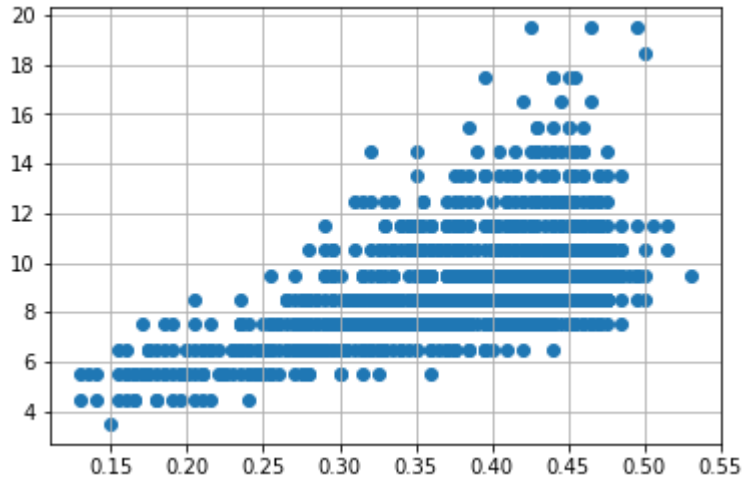



```
var = 'Diameter'
plt.scatter(x = df[var], y = df['age']) plt.grid(True)

df.drop(df[(df['Diameter'] <0.1) &
          (df['age'] < 5)].index, inplace = True)
df.drop(df[(df['Diameter']<0.6) & ( df['age'] >
25)].index, inplace = True)
df.drop(df[(df['Diameter']>=0.6) & ( df['age'] <
25)].index, inplace = True) plt.grid(
```

```
var = 'Height'
plt.scatter(x = df[var], y = df['age'])

df.drop(df[(df['Height'] > 0.4) &
           (df['age'] < 15)].index, inplace = True)
df.drop(df[(df['Height']<0.4) & (
df['age'] > 25)].index, inplace = True)
```




```
True)


var = 'Length' plt.scatter(x = df[var],
y = df['age']) plt.grid(True)

df.drop(df[(df['Length'] <0.1) &
           (df['age'] < 5)].index, inplace = True)
df.drop(df[(df['Length']<0.8) & ( df['age'] >
25)].index, inplace = True)
df.drop(df[(df['Length']>=0.8) & ( df['age'] <
25)].index, inplace = True)
```



CHECKING FOR CATEGORICAL COLUMNS

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

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: DeprecationWarning: `
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/re
```

```
numerical_features
```

```
Index(['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
'Viscera weight', 'Shell weight', 'age', 'Sex_F', 'Sex_I', 'Sex_M'],
dtype='object')
```

```
categorical_features
```

```
Index([], dtype='object')
```

ENCODING

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder() print(df.Length.value_counts())
```

```
0.440    49
0.450    49
0.375    39
0.460    39
0.435    39
..
0.150     1 0.490
1 0.135     1
0.505     1
0.530     1
Name: Length, Length: 77, dtype: int64
```

SPLITTING DATA INTO DEPENDENT AND INDEPENDENT VARIABLES

```
x=df.iloc[:,:5] x
```

	Length	Diameter	Height	Whole weight	Shucked weight

1	0.350	0.265	0.090	0.2255	0.0995	
4	0.330	0.255	0.080	0.2050	0.0895	
5	0.425	0.300	0.095	0.3515	0.1410	
11	0.430	0.350	0.110	0.4060	0.1675	
14	0.470	0.355	0.100	0.4755	0.1675	
...	...	...	...	...	...	
4162	0.385	0.255	0.100	0.3175	0.1370	
4163	0.390	0.310	0.085	0.3440	0.1810	
4164	0.390	0.290	0.100	0.2845	0.1255	
4165	0.405	0.300	0.085	0.3035	0.1500	
4166	0.475	0.365	0.115	0.4990	0.2320	

```
y=df.iloc[:,1238 rows × 5 columns5:]
y
```

	Viscera weight	Shell weight	age	Sex_F	Sex_I	Sex_M
1	0.0485	0.070	8.5	0	0	1
4	0.0395	0.055	8.5	0	1	0
5	0.0775	0.120	9.5	0	1	0
11	0.0810	0.135	11.5	0	0	1
14	0.0805	0.185	11.5	1	0	0
...	...	...	...	...	...	...
4162	0.0680	0.092	9.5	0	0	1
4163	0.0695	0.079	8.5	0	1	0
4164	0.0635	0.081	8.5	0	1	0
4165	0.0505	0.088	8.5	0	1	0
4166	0.0885	0.156	11.5	0	1	0

1238 rows × 6 columns

SPLITTING DATA INTO TRAINING AND TESTING SET

```
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)
```

BUILDING THE MODEL

```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler() x_train=ss.fit_transform(x_train)
```

```
from sklearn.linear_model import LinearRegression
mlr=LinearRegression() mlr.fit(x_train,y_train)
```

```
LinearRegression()
```

FEATURE SCALING

```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
x_train=ss.fit_transform(x_train)
mlrpred=mlr.predict(x_test[0:9]) mlrpred
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has featur
f"X has feature names, but {self.__class__.__name__} was fitted without"
array([[0.07190038, 0.10338295, 9.55289859, 0.1289135 , 0.66542744,
0.20565906],
       [0.07112613, 0.10235605, 9.51189003, 0.1267115 , 0.67035623,
        0.20293228],
       [0.07029407, 0.10117733, 9.46862407, 0.12191234, 0.67942165,
0.19866601],
       [0.06623994, 0.09546054, 9.32030542, 0.10796573, 0.7029605 ,
0.18907377],
       [0.07475707, 0.10743069, 9.63305074, 0.14267247, 0.64249976,
0.21482777],
       [0.06544668, 0.09447413, 9.30086558, 0.1049846 , 0.70869955,
        0.18631585],
       [0.06521001, 0.09417477, 9.27584415, 0.1011369 , 0.71577285,
0.18309025],
       [0.07621426, 0.10865426, 9.63668674, 0.14780016, 0.63229706,
0.21990277],
       [0.07512972, 0.10714957, 9.59365441, 0.14404076, 0.63875888,
        0.21720036]])
```

TRAINING AND TESTING THE MODEL

```
x_test[0:5]
```

	Length	Diameter	Height	Whole weight	Shucked weight

645	0.445	0.330	0.120	0.347	0.1200
3603	0.420	0.325	0.110	0.325	0.1245
2731	0.410	0.315	0.100	0.300	0.1240
617	0.320	0.240	0.085	0.170	0.0655
2014	0.470	0.375	0.105	0.441	0.1670

```
y_test[0:5]
```

	Viscera weight	Shell weight	age	Sex_F	Sex_I	Sex_M
645	0.0840	0.1050	12.5	0	1	0
3603	0.0755	0.1025	8.5	0	1	0
2731	0.0575	0.1000	9.5	0	1	0
617	0.0470	0.0490	8.5	0	0	1
2014	0.0865	0.1450	11.5	0	1	0

MEASUREMENT OF PERFORMANCE USING METRICS

```
from sklearn.metrics import r2_score
r2_score(mlr.predict(x_test),y_test)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:444: UserWarning: X has featur
f"X has feature names, but {self.__class__.__name__} was fitted without" -
307.8528438815492
```