- Libraries

```
import numpy as
np import pandas
as pd
import matplotlib.pyplot as tlp
%matplotlib inline
import seaborn as
ss
```

- Loading the dataset

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

a=pd.read_csv('/content/abalone.csv')

a.head()
```

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								

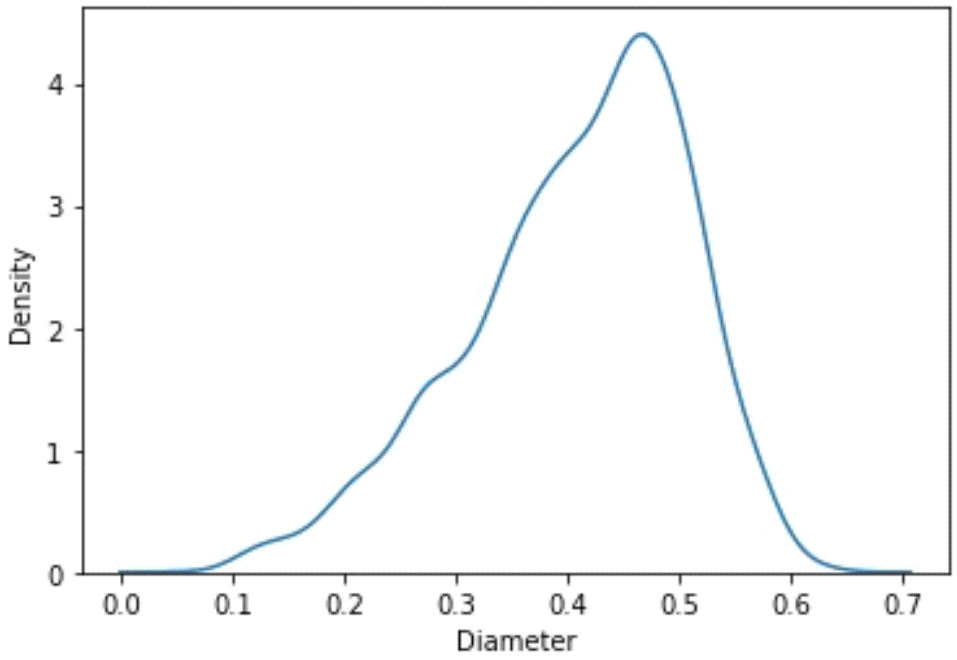
	Shell weight	Rings
0	0.150	15
1	0.070	7
2	0.210	9
3	0.155	10
4	0.055	7

```
a['age']=a['Rings']+1.
5
a=a.drop('Rings',axis=
1)
```

- A. univariate Analysis

```
ss.kdeplot(a['Diameter'])
```

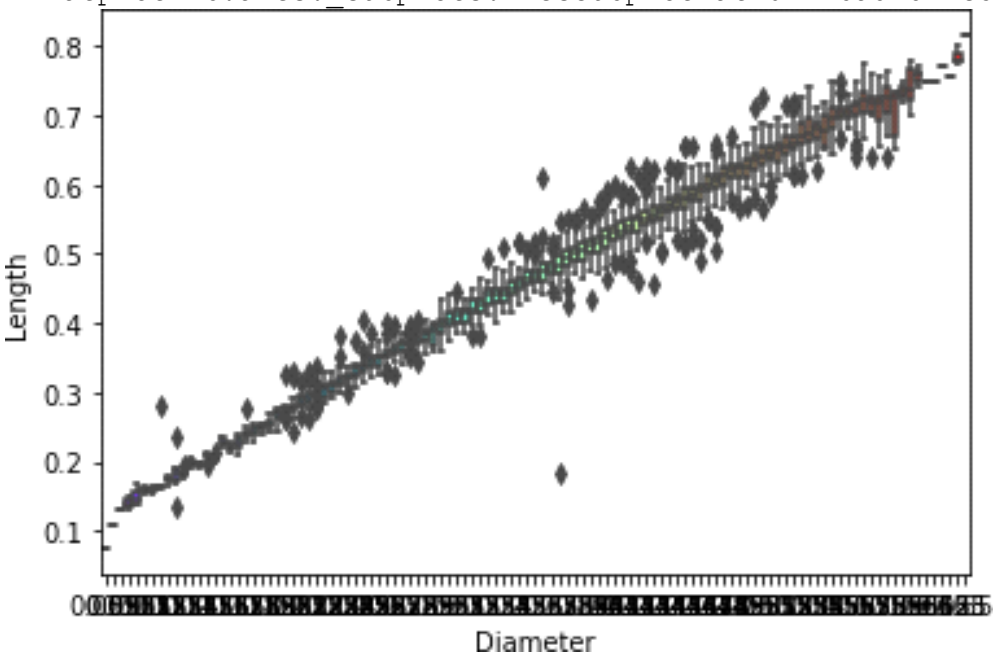
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbad1961d90>
```



- Bi-Variate Analysis

```
ss.boxplot(x=a.Diameter,y = a.Length, palette='rainbow')
```

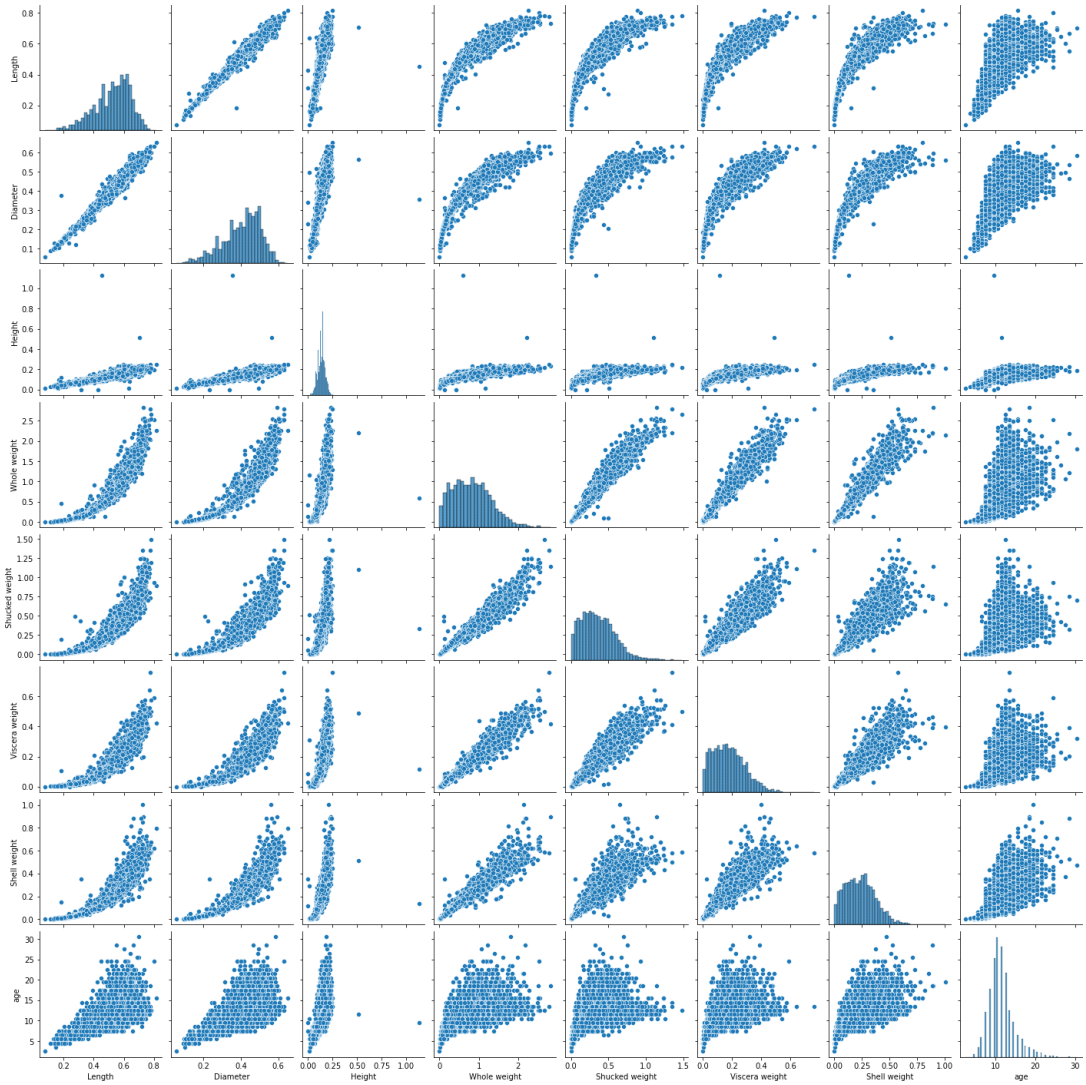
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbad1844c50>
```



- Multi-Variate Analysis

```
ss.pairplot(a)
```

```
<seaborn.axisgrid.PairGrid at 0x7fbad096df50>
```



- Descriptive Statistics

```
a.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

- Whole weight 4177 non-null float64
- Shucked weight 4177 non-null float64
- Viscera weight 4177 non-null float64

```
 •   Shell weight    4177 non-null   float64
 •   age             4177 non-null
     float64dtypes: float64(8), object(1)
memory usage: 293.8+

KB

a['Diameter'].describe

()
```

count	4177.000000
mean	0.407881
std	0.099240
min	0.055000
25%	0.350000
50%	0.425000
75%	0.480000
max	0.650000

```
Name: Diameter, dtype:

float64a['Sex'].value_counts

<bound method IndexOpsMixin.value_counts of 0       M
•       M
•       F
•       M
4       I
      ..
   4172

       F
   4173

       M
   4174

       M
   4175

       F
   4176

       M
Name: Sex, Length: 4177, dtype:
```

object> 5.Checking for missing values and

dealing with them`a.isnull()`

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	\
0	False	False	False	False	False	False	

1	False	False	False	False	False	False	
2	False	False	False	False	False	False	
3	False	False	False	False	False	False	
4	False	False	False	False	False	False	
...	...	...	...	...	...	...	
4172	False	False	False	False	False	False	
4173	False	False	False	False	False	False	
4174	False	False	False	False	False	False	
4175	False	False	False	False	False	False	
4176	False	False	False	False	False	False	

	Viscera weight	Shell	weight	age
0	False		False	False
1	False		False	False
2	False		False	False
3	False		False	False
4	False		False	False
...	...		...	...
4172	False		False	False
4173	False		False	False
4174	False		False	False
4175	False		False	False
4176	False		False	False

[4177 rows x 9

columns]

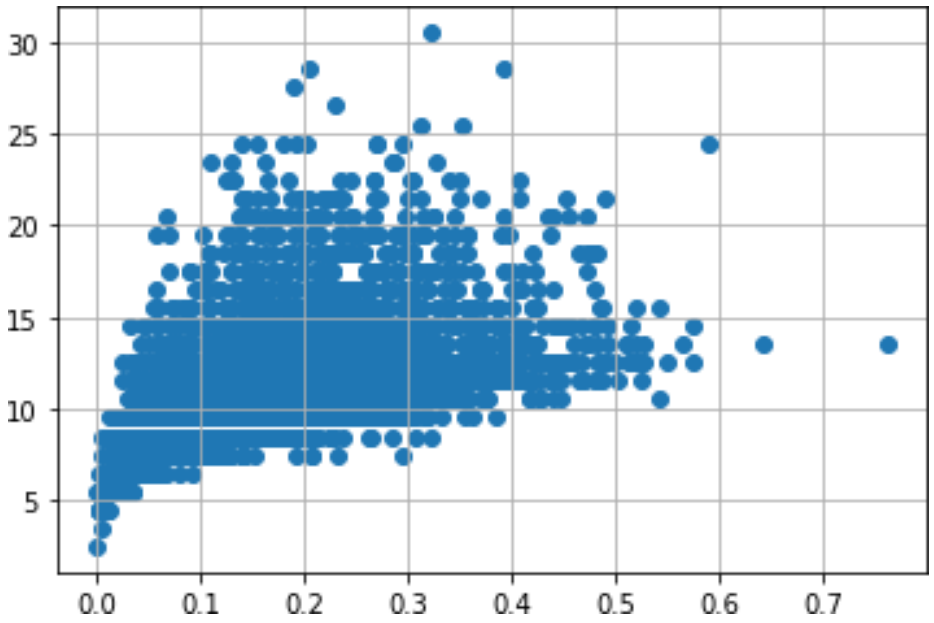
```
a.isnull().sum()
```

```
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
age               0
Sex_F             0
Sex_I             0
Sex_M             0
dtype: int64
```

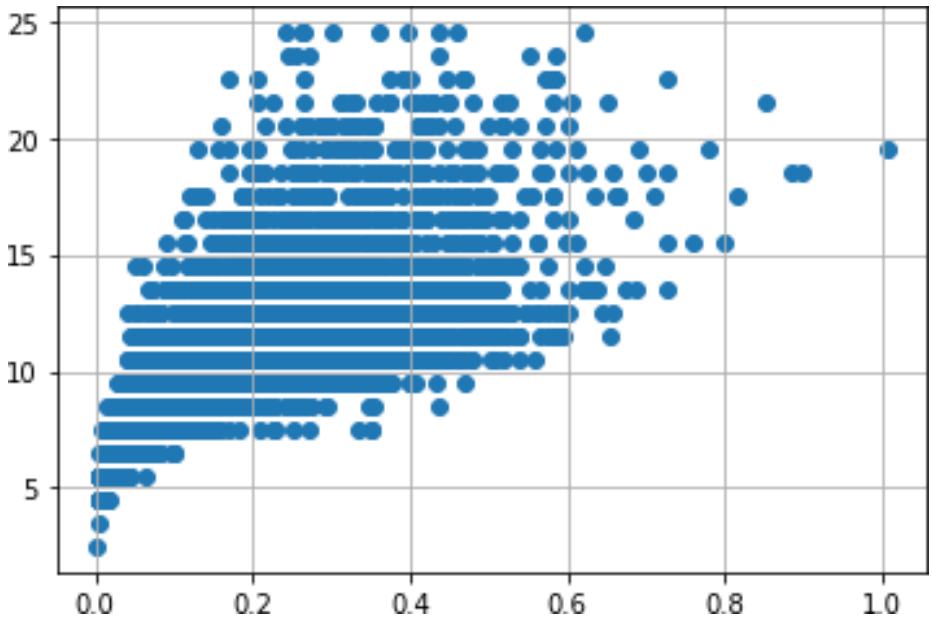
- Find the outliers and replace the outliers

```
a=pd.get_dummies(a

) dummy_a=a
```

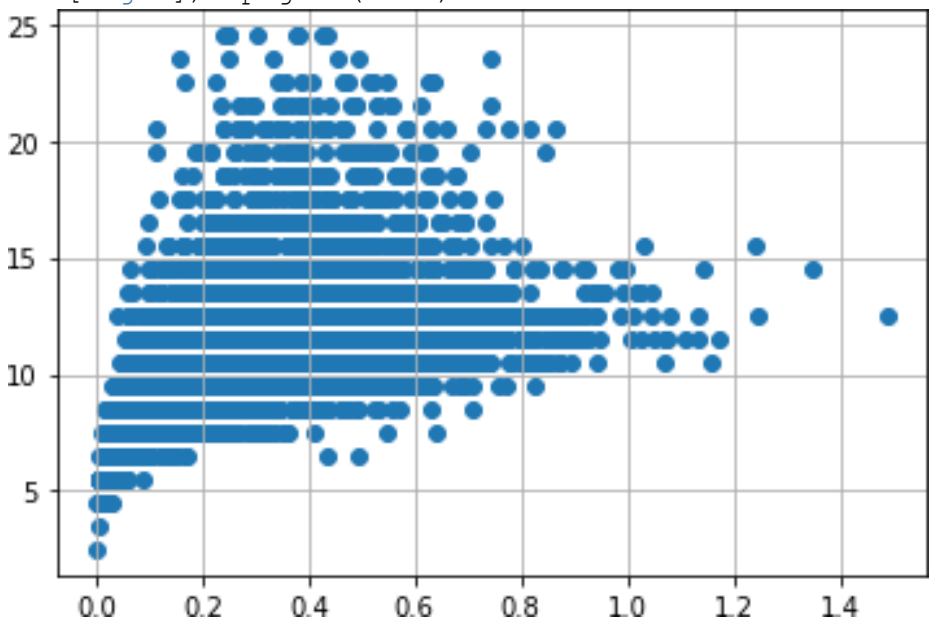
```
var='Viscera

weight'

tlp.scatter(x=a[var],y=a['age'])
tlp.grid(True)
```




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

var='Shell weight'
tlp.scatter(x=a[var],y=a['age'])
tlp.grid(True)
```

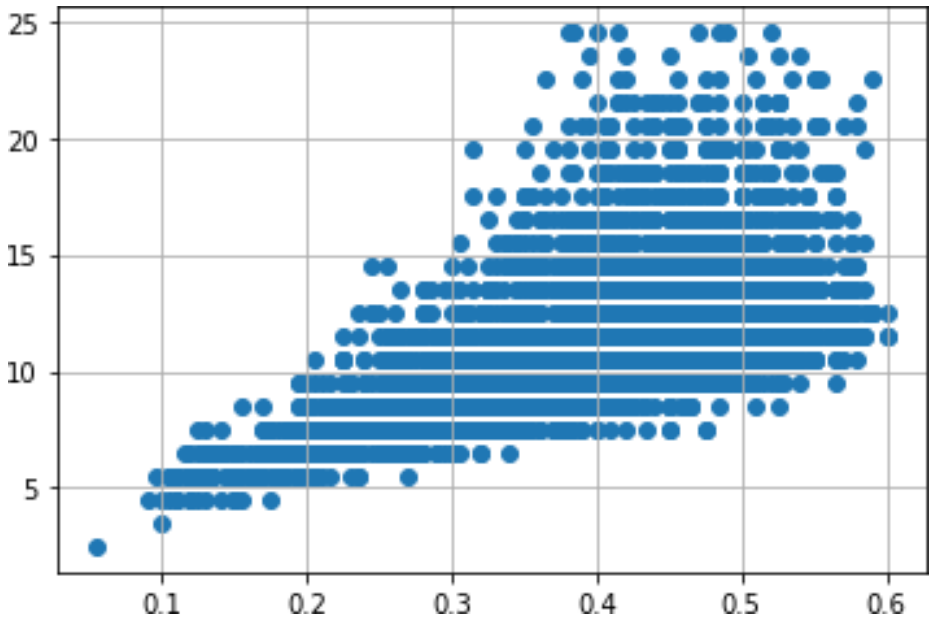



```
a.drop(a[(a['Shell weight'] > 0.6) &
          (a['age'] < 25)].index, inplace = True)

a.drop(a[(a['Shell weight']<0.8) & (
a['age'] > 25)].index, inplace = True)

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

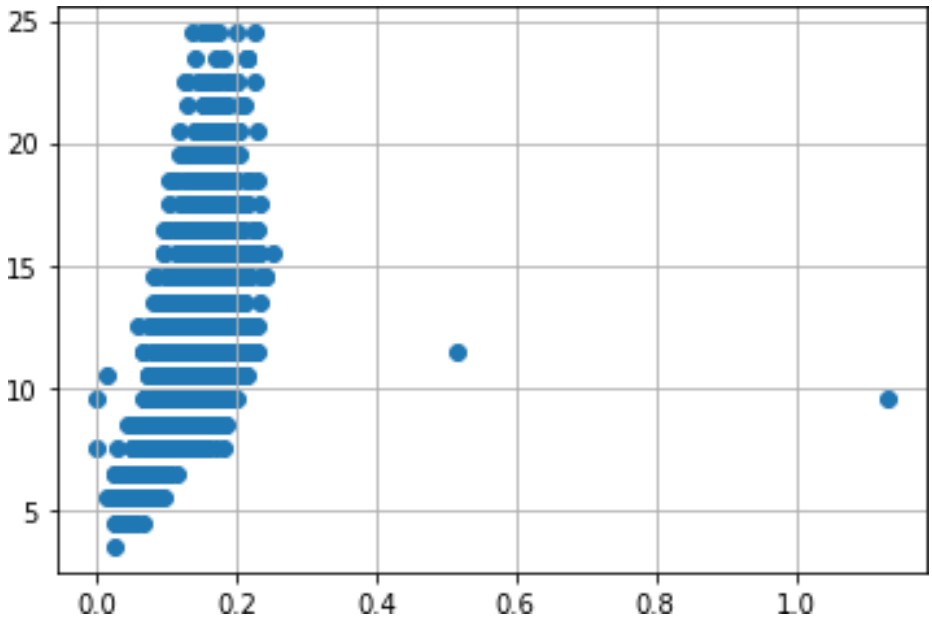



```
a.drop(a[(a['Whole weight'] >= 2.5) &
          (a['age'] < 25)].index, inplace = True)
a.drop(a[(a['Whole weight']<2.5) & (
```

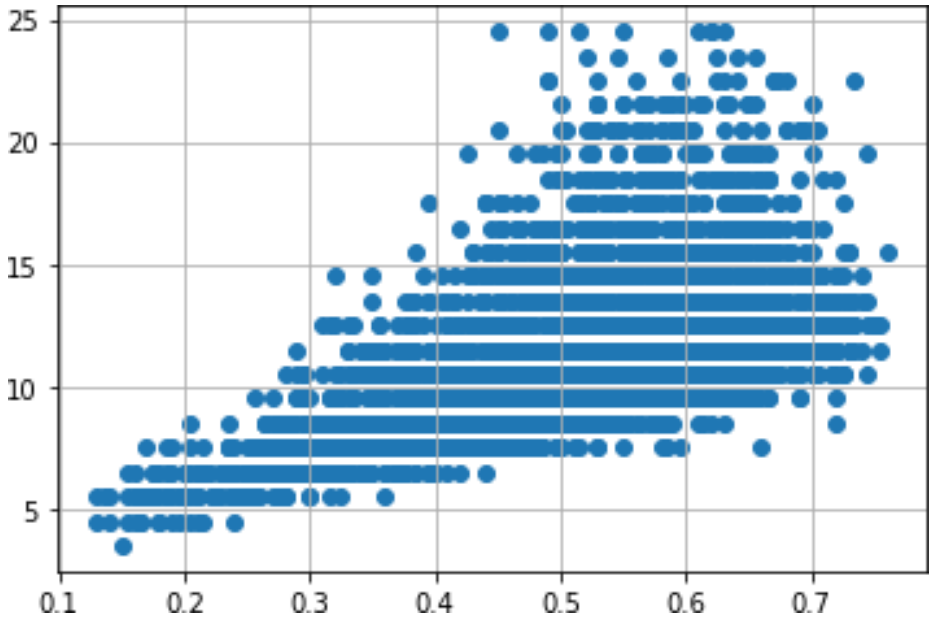
```
a['age'] > 25)].index, inplace = True)

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




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

var = 'Height'
tlp.scatter(x = a[var], y =
a['age'])tlp.grid(True)
```




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

var = 'Length'
tlp.scatter(x = a[var], y =
a['age'])tlp.grid(True)
```




```
a.drop(a[(a['Length'] <0.1) &
          (a['age'] < 5)].index, inplace = True)
```

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

- Checking for categorical columns

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

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2:
DeprecationWarning: `np.object` is a deprecated alias for the builtin
`object`. To silence this warning, use `object` by itself. Doing this
will not modify any behavior and is safe.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-
notes.html#deprecations
```

```
numerical_features
```

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

```
categorical_features
```

```
Index([],

dtype='object')
```

Encoding

```
from sklearn.preprocessing import
LabelEncoderle=LabelEncoder()
print(a.Length.value_counts())
```

0.550	93
0.575	93
0.625	93
0.580	92
0.600	86
	..
0.755	2
0.220	2
0.150	1
0.135	1
0.760	1
Name:	Length, Length: 126, dtype: int64

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

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

```
[4096 rows x 5

columns]

y=a.iloc[:,:5]

y
```

	Length	Diameter	Height	Whole	weight	Shucked	weight
0	0.455	0.365	0.095		0.5140		0.2245

1	0.350	0.265	0.090	0.2255	0.0995
2	0.530	0.420	0.135	0.6770	0.2565
3	0.440	0.365	0.125	0.5160	0.2155
4	0.330	0.255	0.080	0.2050	0.0895
...	...	...	...	...	...
4172	0.565	0.450	0.165	0.8870	0.3700
4173	0.590	0.440	0.135	0.9660	0.4390
4174	0.600	0.475	0.205	1.1760	0.5255
4175	0.625	0.485	0.150	1.0945	0.5310
4176	0.710	0.555	0.195	1.9485	0.9455
[4096	rows x	5 columns]			

- Spliting the data into training and testing

```
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.linear_model import LinearRegression
mlr=LinearRegression()
```

```
mlr.fit(x_train,y_trai

n)LinearRegression()
```

11.Training the model

- Testingthe model

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

	Length	Diameter	Height	Whole	weight	Shucked	weight
2358	0.610	0.485	0.210		1.3445		0.5350
723	0.525	0.410	0.130		0.9900		0.3865
2535	0.640	0.500	0.180		1.4995		0.5930
2717	0.345	0.255	0.095		0.1830		0.0750
29	0.575	0.425	0.140		0.8635		0.3930

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

	Length	Diameter	Height	Whole	weight	Shucked	weight
2358	0.610	0.485	0.210		1.3445		0.5350
723	0.525	0.410	0.130		0.9900		0.3865
2535	0.640	0.500	0.180		1.4995		0.5930
2717	0.345	0.255	0.095		0.1830		0.0750
29	0.575	0.425	0.140		0.8635		0.3930

- Scaling the independent variables

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

mlrpred=mlr.predict(x_test[0:9])
```

mlrpred									
array([[0.61	,	0.485	,	0.21	,	1.3445,	0.535],
[0.525	,	0.41	,	0.13	,	0.99 ,	0.3865],
[0.64	,	0.5	,	0.18	,	1.4995,	0.593],
[0.345	,	0.255	,	0.095	,	0.183 ,	0.075],
[0.575	,	0.425	,	0.14	,	0.8635,	0.393],
[0.57	,	0.48	,	0.18	,	0.9395,	0.399],
[0.61	,	0.485	,	0.165	,	1.087 ,	0.4255],
[0.635	,	0.505	,	0.17	,	1.2635,	0.512],
[0.53	,	0.41	,	0.155	,	0.7155,	0.2805]])

- Measuring the performance using metrics

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

```
1.0
```