

## **ASSIGNMENT 4 TITLE: DA Assignment 4 -Abalone Age Prediction**

**TEAM ID: PNT2022TMID27387**

**NAME: VISHNUKANTH M S**

### **IMPORTING LIBRARIES**

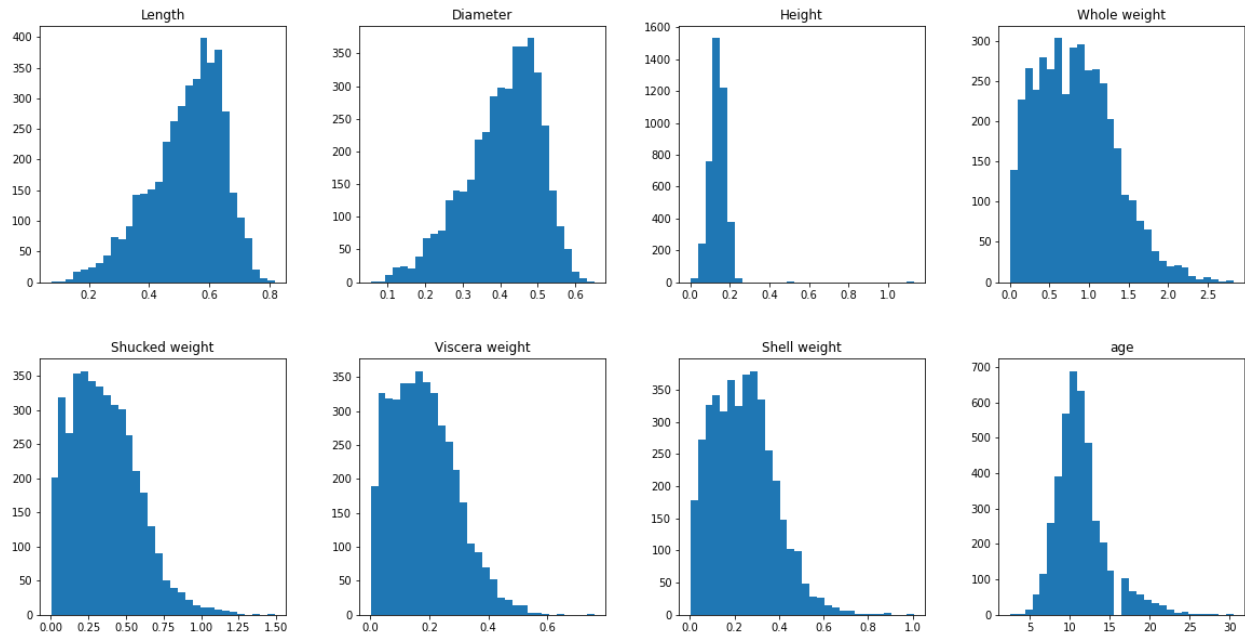
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
```

### **2. Load the dataset into the Google Colab**

```
df=pd.read_csv("/content/abalone.csv")
df['age'] = df['Rings']+1.5
df = df.drop('Rings', axis = 1)
```

### **3. UNIVARIATE ANALYSIS**

```
df.hist(figsize=(20,10), grid=False, layout=(2, 4), bins = 30)
array([[
,
,
],
[
,
,
]],
dtype=object)
```



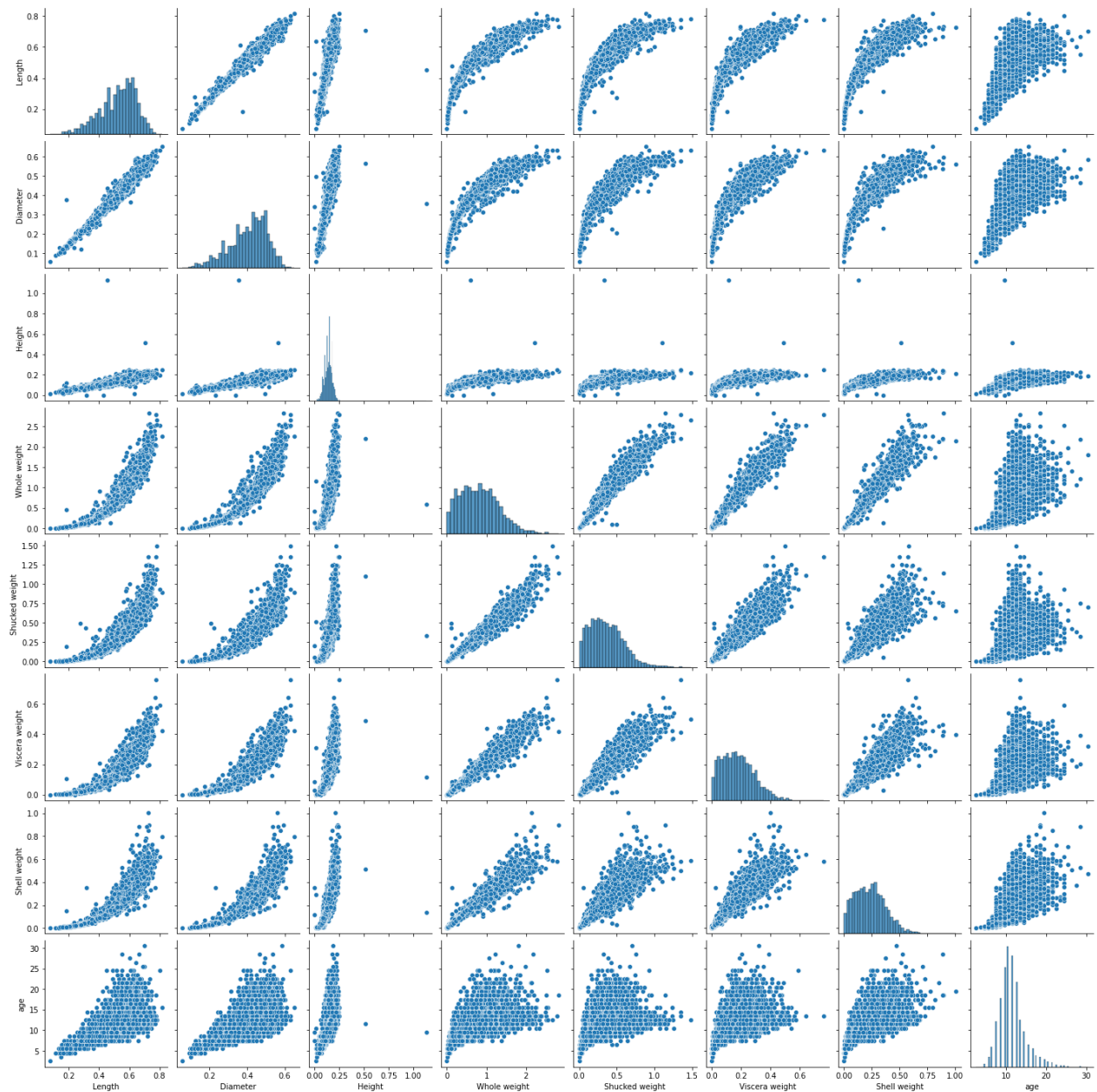
```
df.groupby('Sex')[['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
                   'Viscera weight', 'Shell weight', 'age']].mean().sort_values('age')
```

Out[ ]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	age
<b>Sex</b>								
<b>I</b>	0.42774	0.32649	0.10799	0.43136	0.19103	0.09201	0.12818	9.390462
	6	4	6	3	5	0	2	
<b>M</b>	0.56139	0.43928	0.15138	0.99145	0.43294	0.21554	0.28196	12.20549
	1	7	1	9	6	5	9	7
<b>F</b>	0.57909	0.45473	0.15801	1.04653	0.44618	0.23068	0.30201	12.62930
	3	2	1	2	8	9	0	4

### 3. BIVARIATE ANALYSIS & MULTIVARIATE ANALYSIS

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



#### 4. Descriptive statistics

```
df.describe()
```

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	age
<b>count</b>	4177.00 0000	4177.00 0000	4177.00 0000	4177.00 0000	4177.00 0000	4177.00 0000	4177.00 0000	4177.00 0000
<b>mean</b>	0.52399 2	0.40788 1	0.13951 6	0.82874 2	0.35936 7	0.18059 4	0.23883 1	11.4336 84
<b>std</b>	0.12009 3	0.09924 0	0.04182 7	0.49038 9	0.22196 3	0.10961 4	0.13920 3	3.22416 9
<b>min</b>	0.07500 0	0.05500 0	0.00000 0	0.00200 0	0.00100 0	0.00050 0	0.00150 0	2.50000 0
<b>25 %</b>	0.45000 0	0.35000 0	0.11500 0	0.44150 0	0.18600 0	0.09350 0	0.13000 0	9.50000 0
<b>50 %</b>	0.54500 0	0.42500 0	0.14000 0	0.79950 0	0.33600 0	0.17100 0	0.23400 0	10.5000 00
<b>75 %</b>	0.61500 0	0.48000 0	0.16500 0	1.15300 0	0.50200 0	0.25300 0	0.32900 0	12.5000 00
<b>max</b>	0.81500 0	0.65000 0	1.13000 0	2.82550 0	1.48800 0	0.76000 0	1.00500 0	30.5000 00

## 5. Check for Missing Values

```
df.isnull().sum()
```

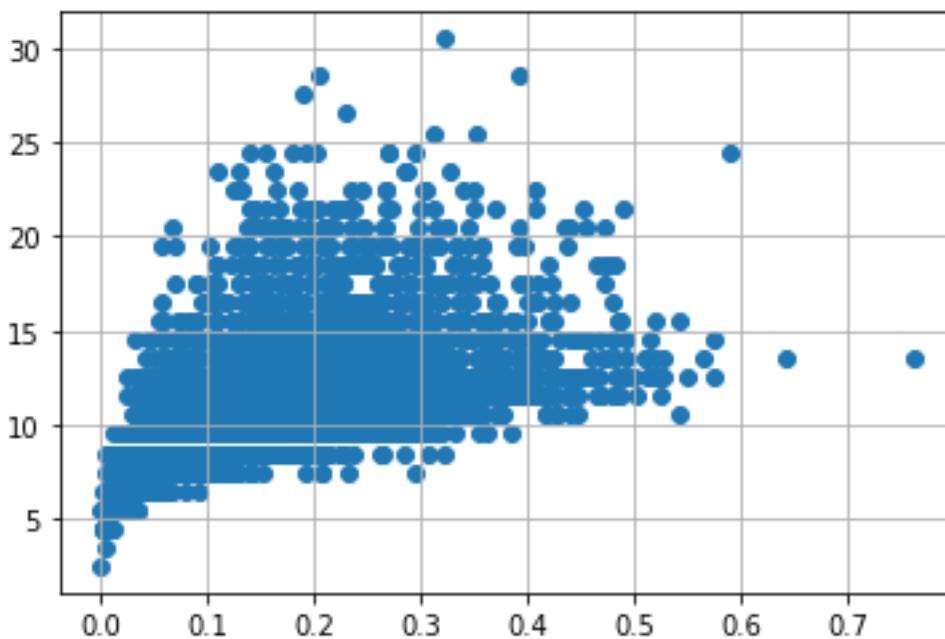
```
Sex      0
Length   0
Diameter  0
Height   0
```

```
Whole weight    0
Shucked weight  0
Viscera weight  0
Shell weight    0
age             0
dtype: int64
```

## 6. OUTLIER HANDLING

```
df = pd.get_dummies(df)
dummy_data = df.copy()
```

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

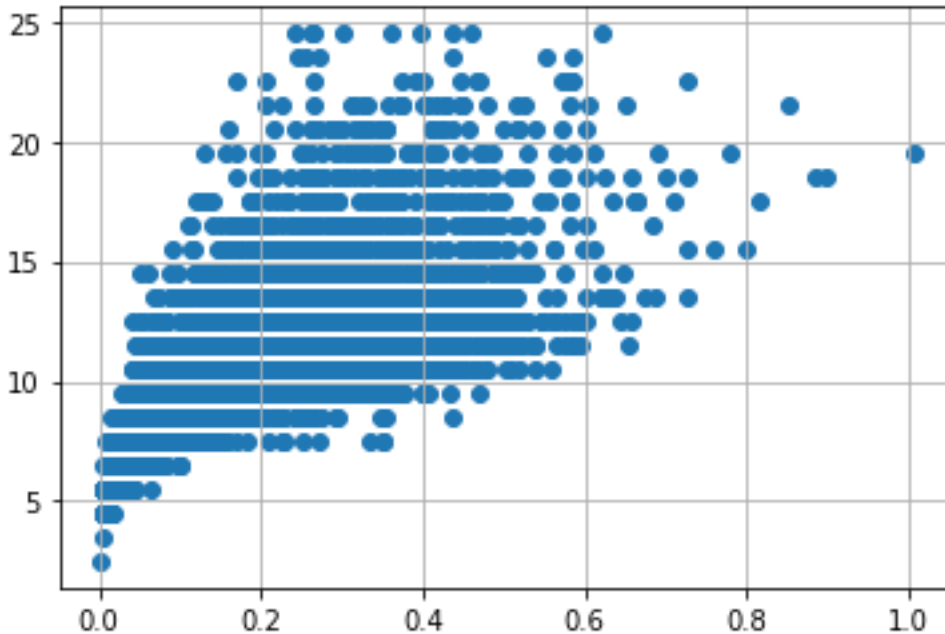


```
# outliers removal
df.drop(df[(df['Viscera weight']> 0.5) & (df['age'] < 20)].index, inplace=True)
df.drop(df[(df['Viscera weight']<0.5) & (df['age'] > 25)].index, inplace=True)
```

```

var = '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)

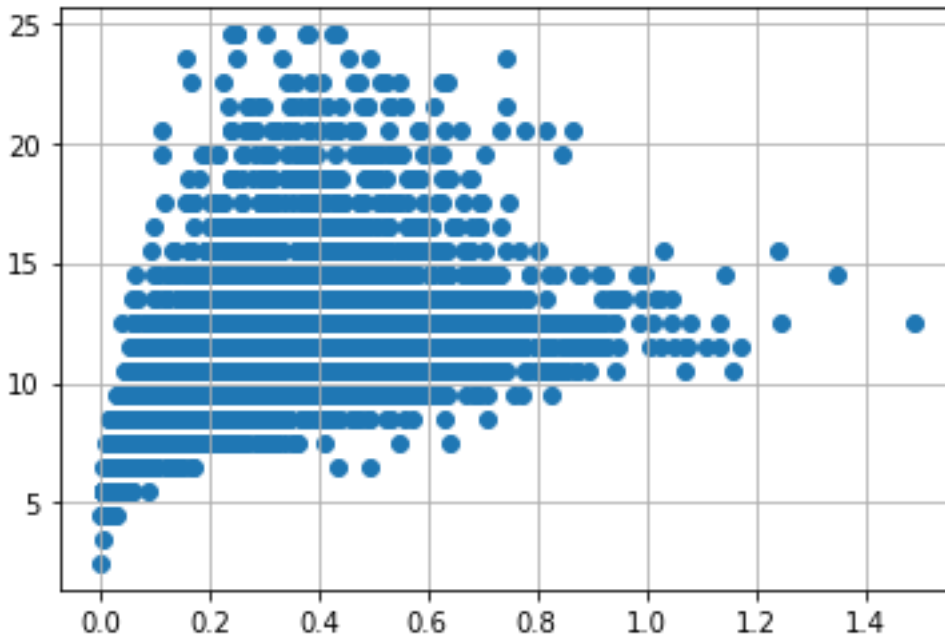
```



```

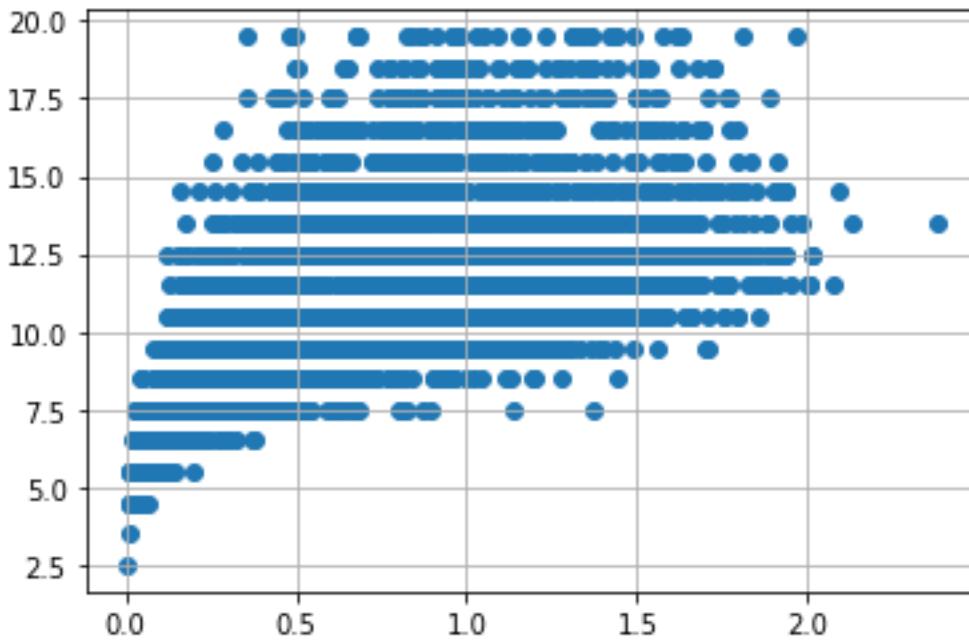
var = 'Shucked weight'
plt.scatter(x = df[var], y = df['age'],)
plt.grid(True)
#Outlier removal
df.drop(df[(df['Shucked weight']>= 1) & (df['age'] < 20)].index, inplace=True)
df.drop(df[(df['Shucked weight']<1) & (df['age'] > 20)].index, inplace=True)

```



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

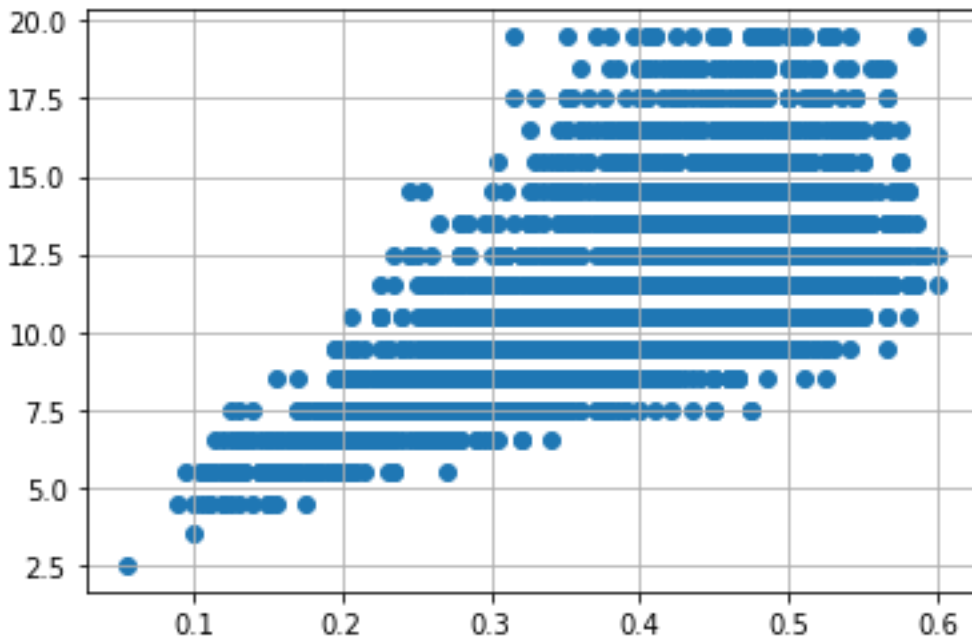
df.drop(df[(df['Whole weight'] >= 2.5) &
          (df['age'] < 25)].index, inplace = True)
df.drop(df[(df['Whole weight'] < 2.5) & (
df['age'] > 25)].index, inplace = True)
```



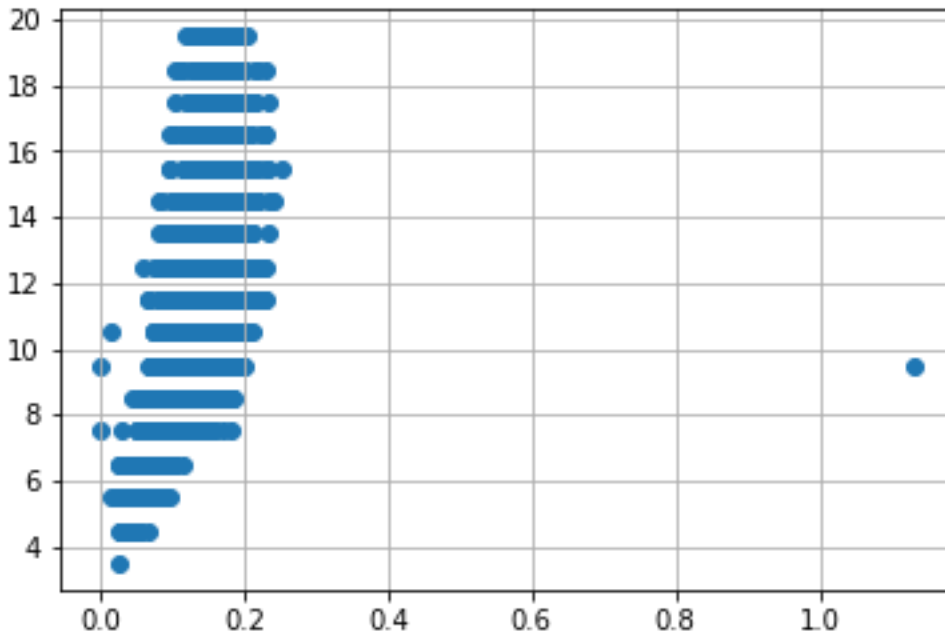
```
var = 'Diameter'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)

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)
```



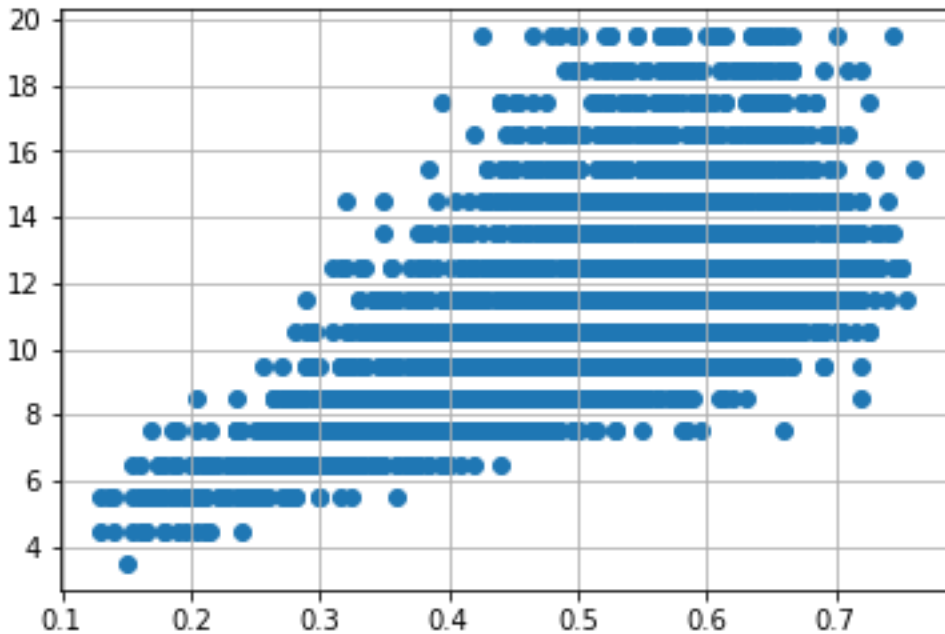


```
var = 'Height'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)
df.drop(df[(df['Height'] > 0.4) &
          (df['age'] < 15)].index, inplace = True)
df.drop(df[(df['Height'] < 0.4) & (
df['age'] > 25)].index, inplace = True)
```



```
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)
```



## 7. Categorical columns

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

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

```
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())
```

0.575 93

0.625 91

0.580 89

0.550 89

0.620 83

..

0.220 2

0.150 1

0.755 1

0.135 1

0.760 1

Name: Length, Length: 126, dtype: int64

## 8. Split the dependent and independent variables

```
x=df.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

	Length	Diameter	Height	Whole weight	Shucked weight
<b>4173</b>	0.590	0.440	0.135	0.9660	0.4390
<b>4174</b>	0.600	0.475	0.205	1.1760	0.5255
<b>4175</b>	0.625	0.485	0.150	1.0945	0.5310
<b>4176</b>	0.710	0.555	0.195	1.9485	0.9455

3995 rows × 5 columns

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

	Viscera weight	Shell weight	age	Sex_F	Sex_I	Sex_M
<b>0</b>	0.1010	0.1500	16.5	0	0	1
<b>1</b>	0.0485	0.0700	8.5	0	0	1
<b>2</b>	0.1415	0.2100	10.5	1	0	0
<b>3</b>	0.1140	0.1550	11.5	0	0	1
<b>4</b>	0.0395	0.0550	8.5	0	1	0
...	...	...	...	...	...	...
<b>4172</b>	0.2390	0.2490	12.5	1	0	0
<b>4173</b>	0.2145	0.2605	11.5	0	0	1

	Viscera weight	Shell weight	age	Sex_F	Sex_I	Sex_M
<b>4174</b>	0.2875	0.3080	10.5	0	0	1
<b>4175</b>	0.2610	0.2960	11.5	1	0	0
<b>4176</b>	0.3765	0.4950	13.5	0	0	1

3995 rows × 6 columns

## 9. Feature Scaling

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

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

```
mlrpred
```

## 10. Train , Test , Split

```
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)
```

## 11. Model building

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

## 12 & 13. Train and Test the model

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

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

#### 14. Measure the performance using metrics

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