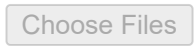
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt


import seaborn as sns


from sklearn.linear_model import LinearRegression
```

Loading the dataset

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



abalone.csv

- **abalone.csv**(text/csv) - 191962 bytes, last modified: 10/22/2022 - 100% done

```
Saving abalone.csv to abalone.csv
```

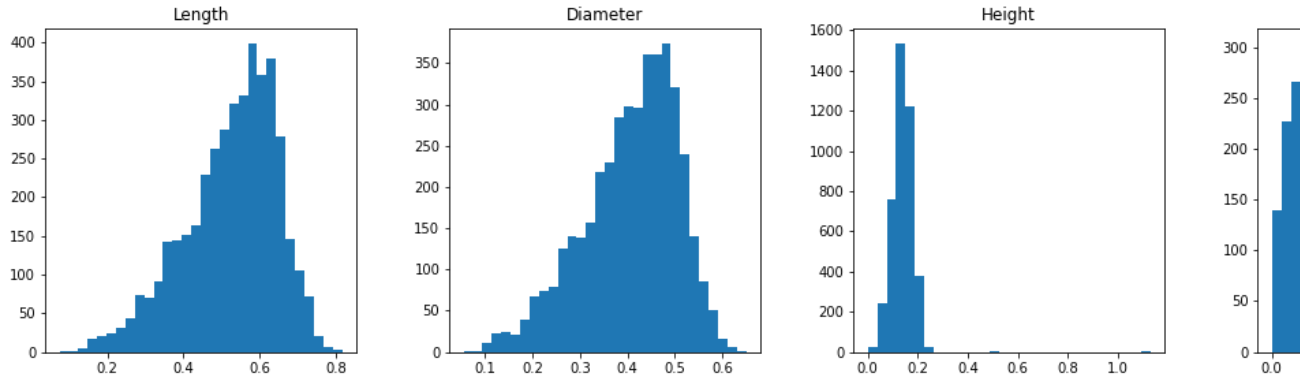
```
df=pd.read_csv('abalone.csv')
```

Univariate Analysis

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


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

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7efe69e67690>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7efe69e24310>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7efe69d892d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7efe69d418d0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7efe69cf7ed0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7efe69cba510>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7efe69cf1b90>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7efe69c34110>]],
      dtype=object)
```



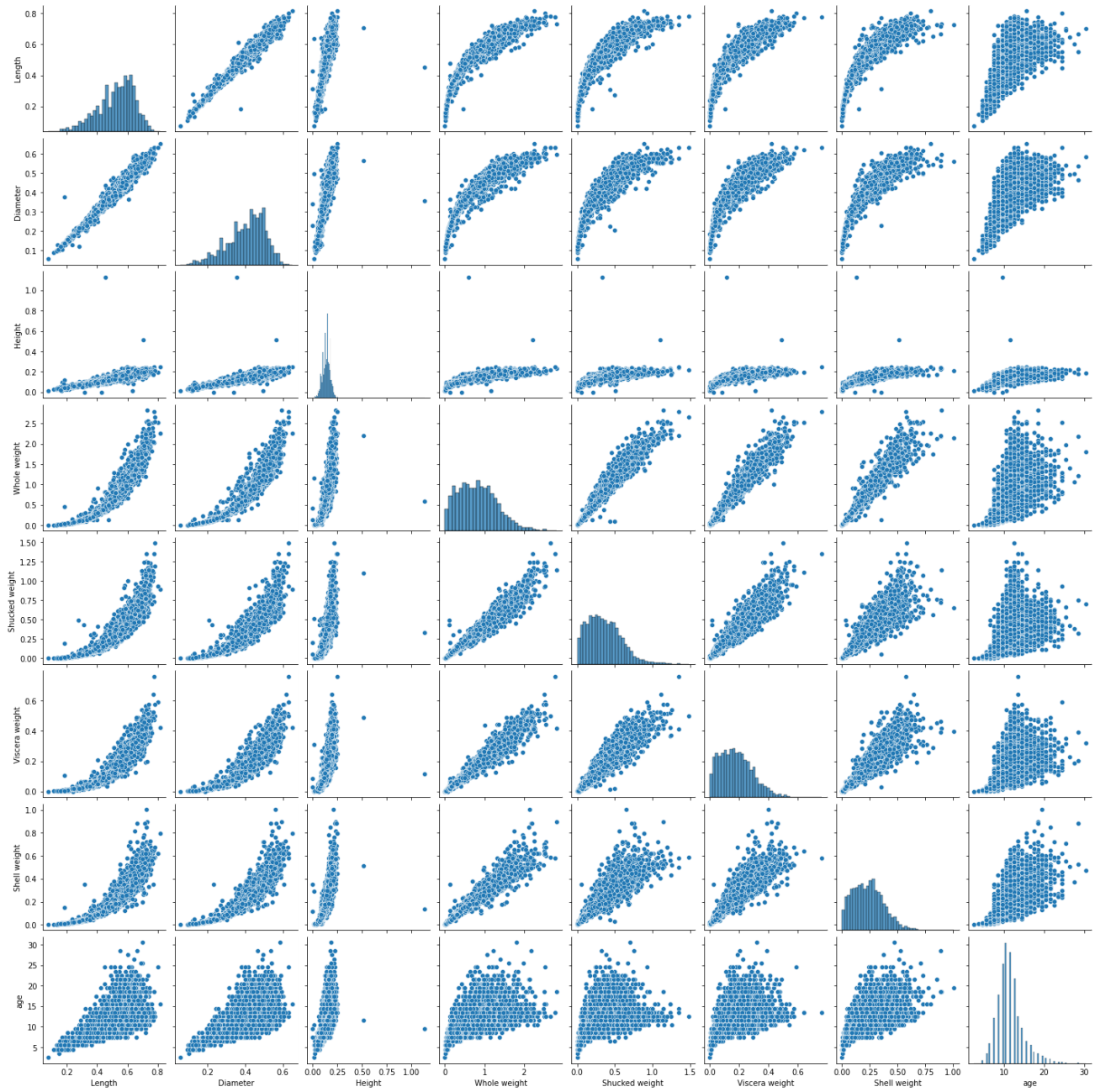

```
df.groupby('Sex')[['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
       'Viscera weight', 'Shell weight', 'age']].mean().sort_values('age')
```

Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	
I	0.427746	0.326494	0.107996	0.431363	0.191035	0.092010	0
M	0.561391	0.439287	0.151381	0.991459	0.432946	0.215545	0
F	0.579093	0.454732	0.158011	1.046532	0.446188	0.230689	0

Bivariate and Multivariate Analysis

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

```
<seaborn.axisgrid.PairGrid at 0x7efe69a9b210>
```



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	
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	

Checking for Missing Values

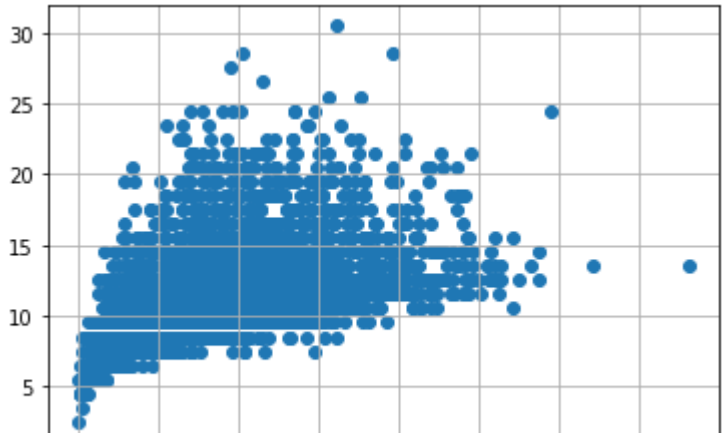
```
df.isnull().sum()
```

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

Outlier Handling

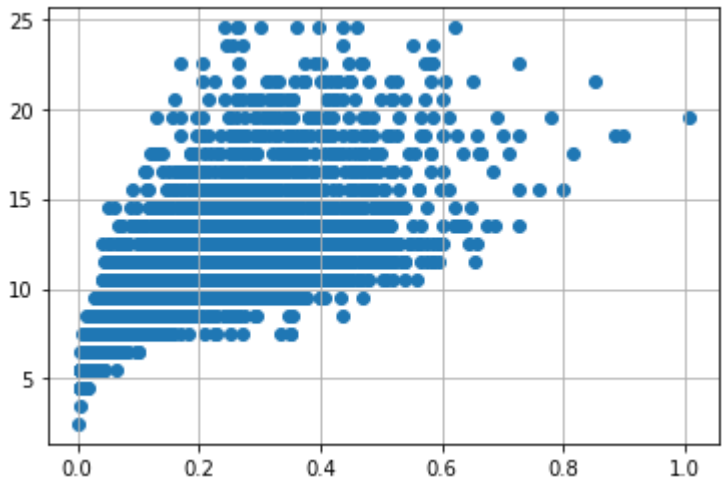
```
df = pd.get_dummies(df)
dummy_data = df.copy()
```

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



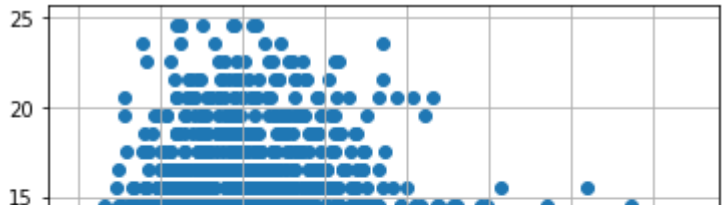

```
# outliers removal
df.drop(df[(df['Viscera weight']> 0.5) & (df['age'] < 20)].index, inplace=True)
df.drop(df[(df['Viscera weight']<0.5) & (df['age'] > 25)].index, inplace=True)
```

```
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)
```



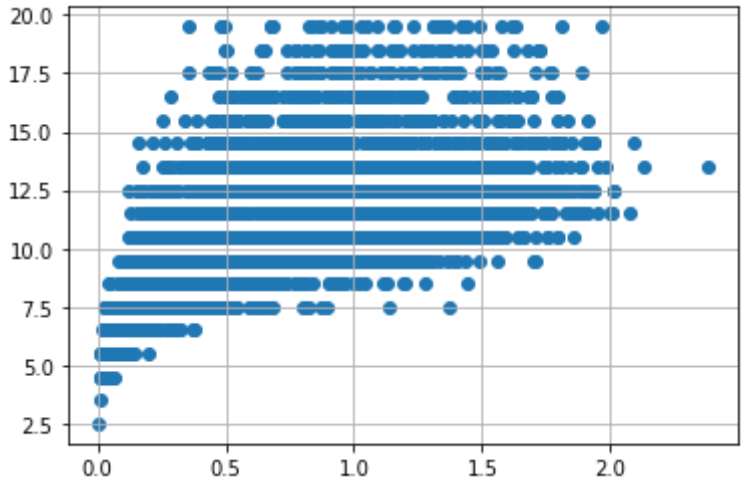

```
var = 'Shucked weight'
plt.scatter(x = df[var], y = df['age'],)
plt.grid(True)

#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)
```



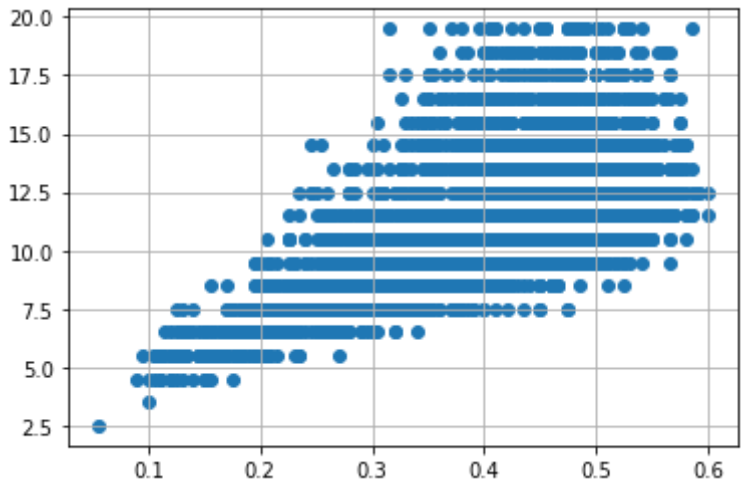

```
var = 'Whole weight'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)

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)
```

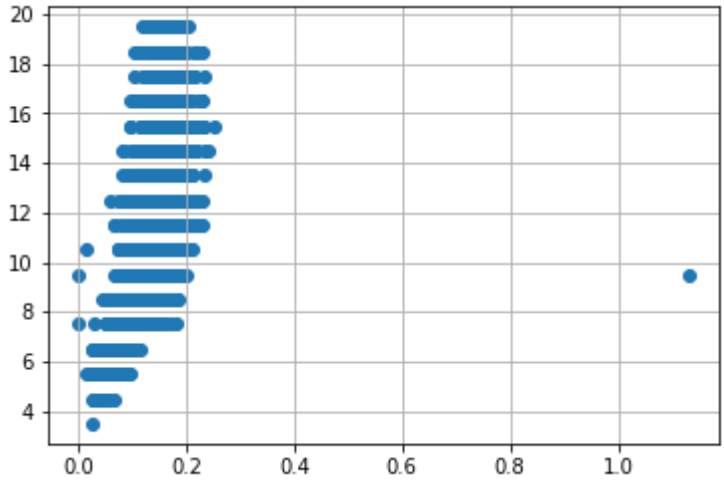



```
var = 'Diameter'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)

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)
```

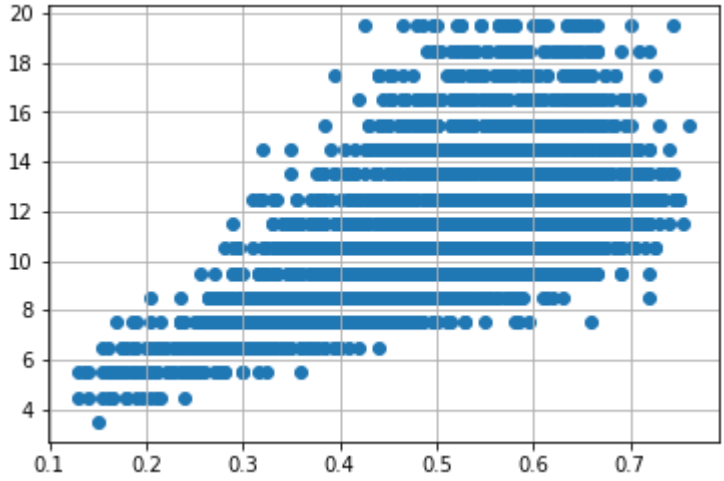


```
var = 'Height'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)
df.drop(df[(df['Height'] > 0.4) &
          (df['age'] < 15)].index, inplace = True)
df.drop(df[(df['Height']<0.4) & (
df['age'] > 25)].index, inplace = True)
```




```
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)
```



Categorical Column

```
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())
```

```
0.575    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
```

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

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

	Viscera weight	Shell weight	age	Sex_F	Sex_I	Sex_M
0	0.1010	0.1500	16.5	0	0	1
1	0.0485	0.0700	8.5	0	0	1
2	0.1415	0.2100	10.5	1	0	0
3	0.1140	0.1550	11.5	0	0	1
4	0.0395	0.0550	8.5	0	1	0
...	...	...	...	...	...	...
4172	0.2390	0.2490	12.5	1	0	0
4173	0.2145	0.2605	11.5	0	0	1
4174	0.2875	0.3080	10.5	0	0	1
4175	0.2610	0.2960	11.5	1	0	0
4176	0.3765	0.4950	13.5	0	0	1

3995 rows × 6 columns

Test, Train and 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)
```

Model Building

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

```
LinearRegression()
```

Train and Test the model

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

	Length	Diameter	Height	Whole weight	Shucked weight
2677	0.615	0.475	0.170	1.0550	0.5430
1248	0.395	0.295	0.095	0.2725	0.1150
4011	0.615	0.530	0.170	1.1200	0.5775
3673	0.595	0.460	0.155	1.0455	0.4565
2184	0.320	0.235	0.080	0.1485	0.0640

Feature Scaling

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

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

```
mlrpred
```

```
array([[ 0.2262845 ,  0.27306603, 10.90939164,  0.34415492,  0.20230076,
         0.45354432],
       [ 0.06026679,  0.08590296,  9.20837637,  0.12357292,  0.66228769,
         0.21413939],
       [ 0.23035955,  0.299738  , 11.69999354,  0.4457304 ,  0.05202141,
         0.50224819],
       [ 0.23023633,  0.29533161, 12.04464726,  0.39104792,  0.19061705,
         0.41833502],
       [ 0.03210049,  0.04424736,  8.18454259,  0.05523546,  0.76225986,
         0.18250468],
       [ 0.19028884,  0.2479983 , 11.41912715,  0.33405618,  0.31790386,
         0.34803996],
       [ 0.26253181,  0.33244139, 12.3415582 ,  0.43696486,  0.08340583,
         0.47962931],
       [ 0.10165739,  0.1509131 , 10.98040148,  0.2322796 ,  0.49895538,
         0.26876501],
       [ 0.41942795,  0.4816662 , 11.77502122,  0.54483571, -0.20325536,
         0.65841965]])
```

Measure the performance using metrics

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

```
-2.970693538485954
```

Colab paid products - Cancel contracts here

✓ 0s completed at 8:19 PM