**Assignment -4**

|  |  |
| --- | --- |
| **Assignment Date** | **08 November 2022** |
| **Student Name** | **S.THAMIZHINIYAN** |
| **Student Roll Number** | **812719104046** |
| **Maximum Marks** | **4 Marks** |



import pandas as pd import numpy as np

from matplotlib import pyplot as plt import seaborn as sns

from sklearn.1inear\_mode1 import LinearRegression



df=pd.read\_csv("/content/drive/NyDrive/Colab Notebooks/abalone.csv”)



d-F[ ’ age ' ] = d-F[ ' Rings ' ]+1.5

df = df.drop('Rings', axis = 1)

Univariate Analysis



df.hist(figsize=(20,10), grid=False, layout=(2, 4), bins = 3B)

[ array([[‹matplotlib.axes.\_subplots.AxesSubplot object at 8x7f3d1b8fb698>,

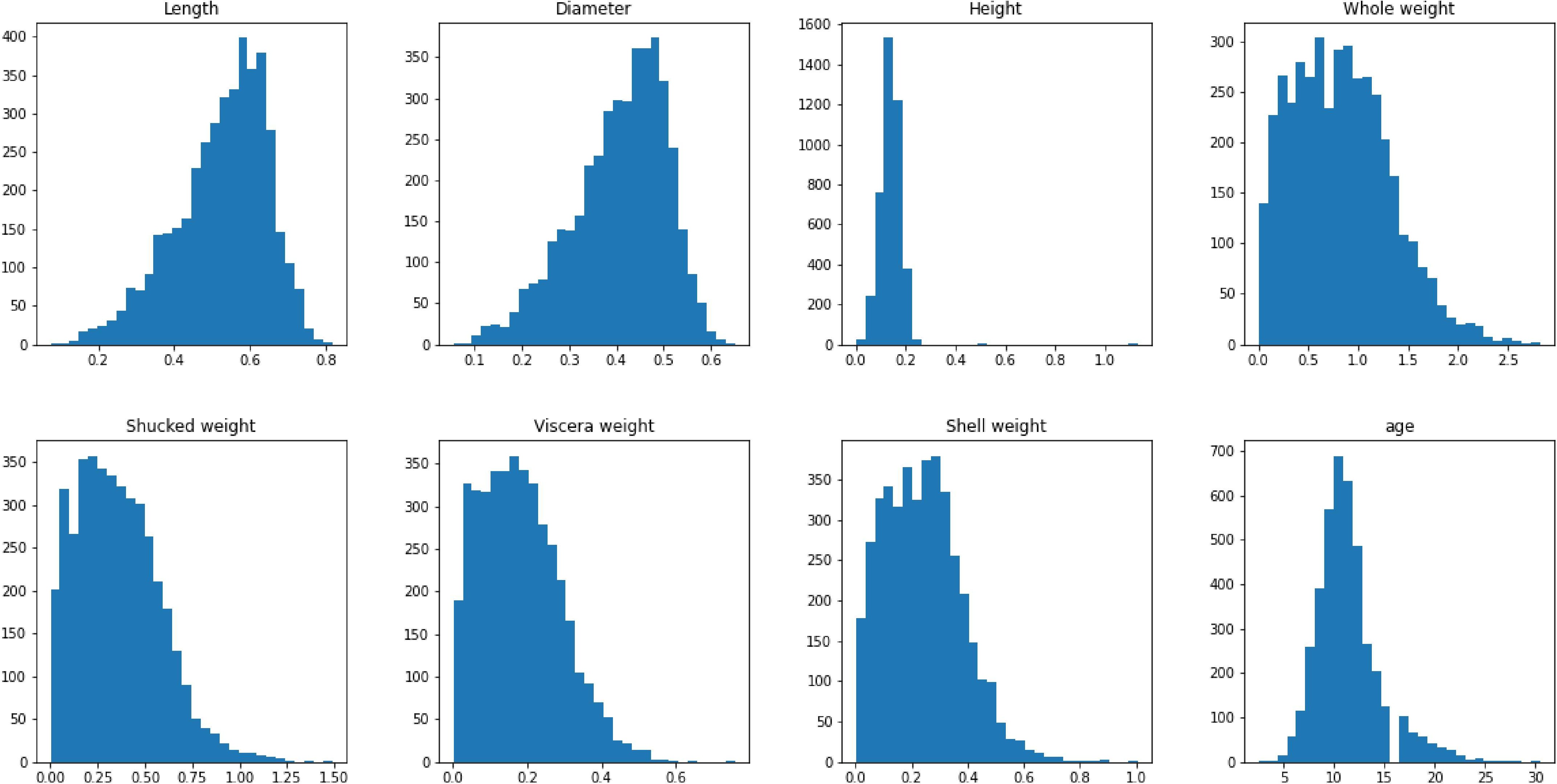
<matplotlib.axes.\_subplots.AxesSubplot object at 0x7f3d1ade4d98>,

‹matplotlib.axes.\_subplots.AxesSubplot object at 0x7f3dladaa398›,

‹matplotlib.axes.\_subplots.AxesSubplot object at Bx7f3d1ad60998>], [‹matplotlib.axes.\_subplots.AxesSubplot object at 0x7f3dladl6f98›,

‹matplotlib.axes.\_subplots.AxesSubplot object at Bx7f3d1acda5d8>,

<matp1otlib.axes.\_subplots.AxesSubplot object at Bx7f3dlac8fc58›,

<matplotlib.axes.\_subplots.AxesSubplot object at 8x7f3dlac53ld8›]], dtype=object)

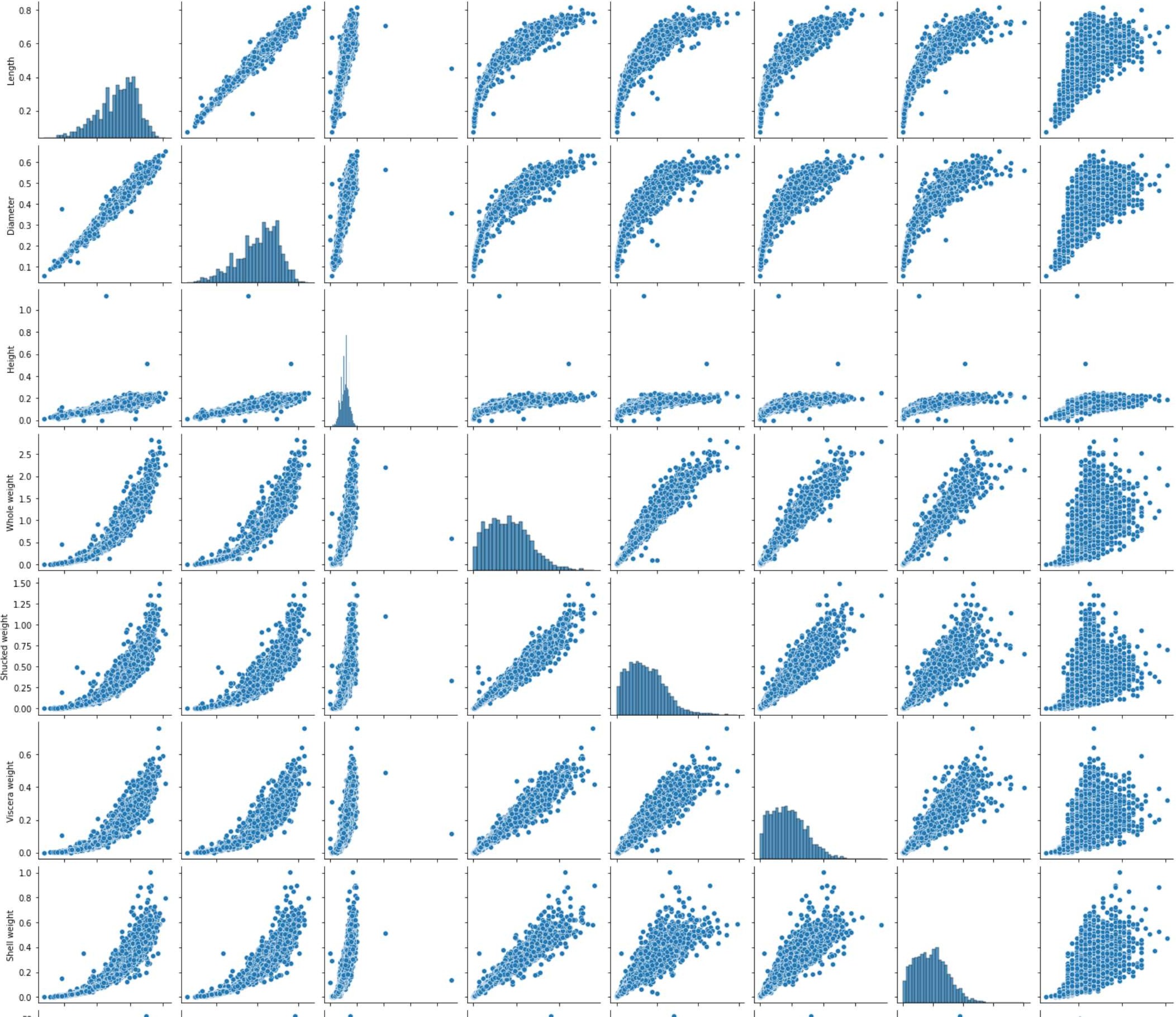
df.groupby('Sex')[['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight', ’Viscera weight’, 'Shell weight’, 'age']].meau().sort\_values('age')

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sex | Length | Diameter | Height | whole weight | Shucked weight | Viscera weight | Shell weight | age |
|  | 0.427746 | 0.326494 | 0.107996 | 0.431363 | 0.191035 | 0.092010 | 0.128182 | 9.390462 |
| M | 0.561391 | 0.439287 | 0.151381 | 0.991459 | 0.432946 | 0.215545 | **0.281969** | **12.205497** |
| F | 0.579093 | 0.454732 | 0.158011 | 1.046532 | 0.446188 | 0.230689 | 0.302010 | 12.629304 |

### Bivariate Analysis

numerical\_features = df.select\_dtypes(include = [np.number]).columns sns.pairplot(df[numerical\_features])

<seaborn.axisgrid.PairGrid at 0x7f3d1a345650>



## Descriptive statistics

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| df.describe() |  | | | | | | | |
|  | Length | Diameter | Height | whole weight | Shucked weight | viscera weight | Shell weight | age |
| count | 4177.000000 | 4177.000000 | 4177.000000 | 4177.000000 | 4177.000000 | 4177.000000 | 4177.000000 | 4177.000000 |
| mean | 0.523992 | 0.407881 | 0.139516 | 0.828742 | 0.359367 | 0.180594 | 0.238831 | 11.433684 |
| std | 0.120093 | 0.099240 | 0.041827 | 0.490389 | 0.221963 | 0.109614 | 0.139203 | 3.224169 |
| min | 0.075000 | 0.055000 | 0.000000 | 0.002000 | 0.001000 | 0.000500 | 0.001500 | 2.500000 |
| 25% | 0.450000 | 0.350000 | 0.115000 | 0.441500 | 0.186000 | 0.093500 | 0.130000 | 9.500000 |
| 50% | 0.545000 | 0.425000 | 0.140000 | 0.799500 | 0.336000 | 0.171000 | 0.234000 | 10.500000 |
| 75% | 0.615000 | 0.480000 | 0.165000 | 1.153000 | 0.502000 | 0.253000 | 0.329000 | 12.500000 |
| max | 0.815000 | 0.650000 | 1.130000 | 2.825500 | 1.488000 | 0.760000 | 1.005000 | 30.500000 |

Check for missing values

df.isnull().sum()

### Outlier handling

df - pd.get\_dummies(df) dummy\_da ta = df . copy( ) var = 'Viscera weight'

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

# outliers removal

d-F. drop(df[ (d-F[ ' VI scera weight ' ] › 0. 5) & (df-[ ' age ' ] ‹ 20) ] . Index, inp1ace=True) df.drop(df[(df['Uiscera 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']‹8.8) 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'] < 28)].index, inplace=True) df.drop(df[(df['Shucked weight’]<1) & (df['age’] > 28)].iudex, inplace=True)

var = ' Nhole weight ’

pit . scatter (x = df-[var] , y = df[ ' age ' ] ) p1t . grid(True)

df.drop(df[(df['Whole weight’] ›= 2.5) &

(df[’age'] ‹ 25)].index, inplace = True)

df. drop(df-[ (df[ ' Nhole weight ’ ] ‹2. 5) & (

d-F[ ’ age ' ] › 25) ] . Index, 1nplace = True)



var = ' Diameter '

pit . scatter (x = df-[var] , y = df[ ' age ' ] ) p1t . grid(True)

df . d rop ( df- [ ( df [ ' Diazete n ' ] ‹8. 1) &

(df[’age'] ‹ 5)].index, inplace = True)

df. drop(df-[ (df[ ' Diameter ' ] ‹0. 6) & (

d-F[ ’ age ' ] › 25) ] . Index, 1nplace = True)

d-F- . drop(df-[ (d1°[ ' Diameter ' ] ›=0. 6) & (

df-[ ’ age ' ] ‹ 25) ] . Index, 1nplace = True)

var = ' Height '

p1t . scatter (x - df[var] , y - df[ ' age ' ] ) p1t . grid(True)

d-F. drop(d-I- [ (df-[ ' Height ' ] › 6 . 4) &

(df[ ’ age ' ] ‹ 15) ] . Index, Inplace = True)

d-F. drop(df-[ (d-F[ ' Height ' ] ‹0. 4) & (

d-I°[ ’ age ' ] › 25) ] . index, 1nplace = True)

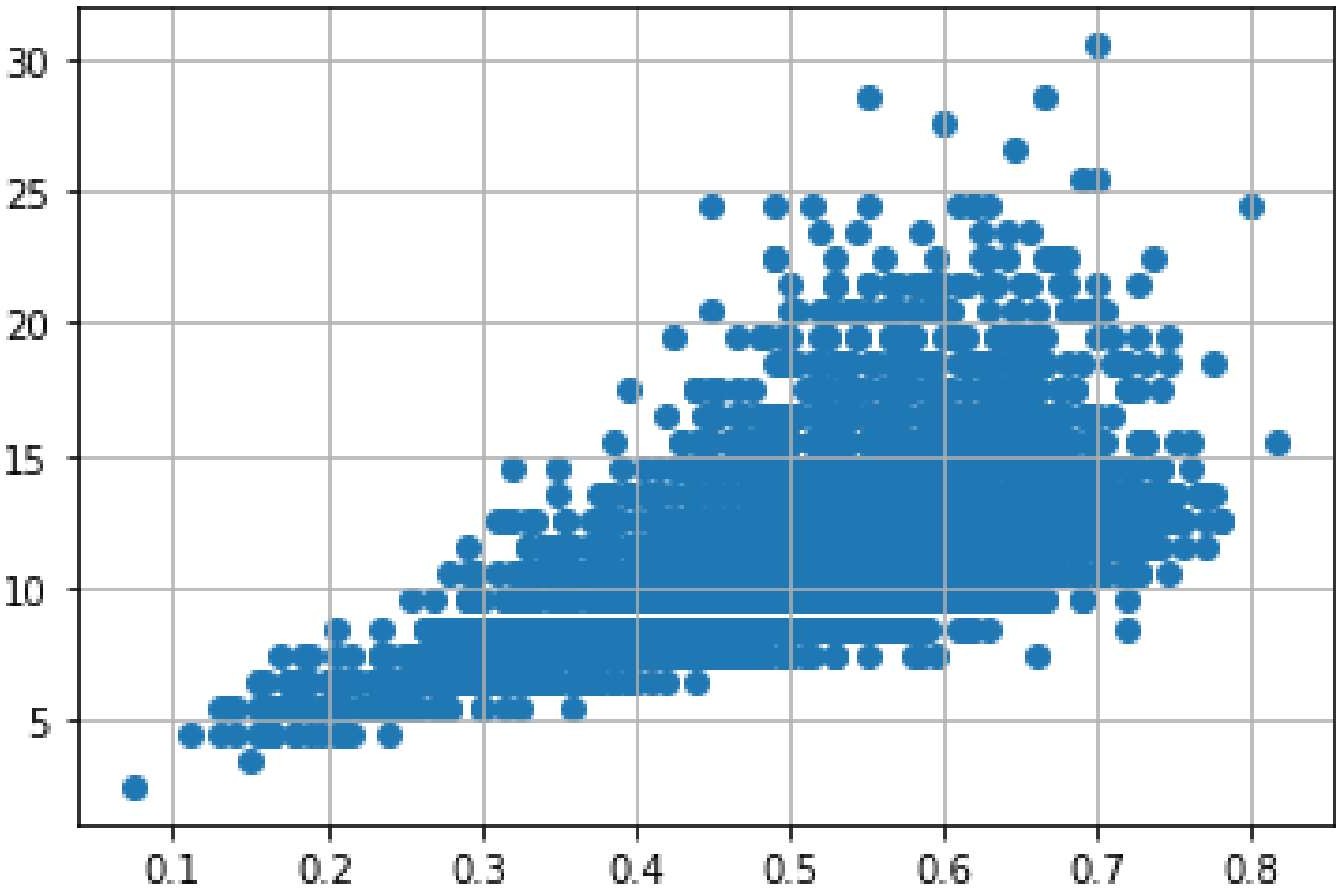
var = 'Length'

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

df.drop(df[(df['Leugth’] ‹8.1) &

(df['age'] < 5)].index, inplace = True) dfdropd[df['Leugth]‹0.8) & (

df['age'] › 25)].index, inplace = True) df.dropd[df['Length]>=8.8) & ( df[’age'] ‹ 25)].iudex, inplace = True)



Categorical columns

numerical\_features = df.select\_dtypes(include = [np.number]).columns categorica1\_features = df.select\_dtypes(include = [np.object]).columns

/usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:2: Deprecationwarning: ’up.object’ is a deprecated alias for the builtin ’object’ To siler Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.8-notes.html#deprecations



numerical\_features

Index( [ ’ Length ’, ’ DI ameter', ’ Height ', ' Mhole weight ' , ' Shucked weight ' , ’Uiscera weight', 'Shell weight', 'age’],

dtype=’object’)



categonica l\_featunes

Index( [ ’ Sex ' ] , dtype= ' obj ect ' )

ENCODING



from sklearn.preprocessing import LabelEncoder le=LabelEncoder()

print(df.Sex.value\_counts())

M 1525

1 1341

F 1301

Name: Sex, dtype: int64

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x=df.iloc[:, | :5] |  | | | |
|  | Sex | Length | Diameter | HeIght | Nhole we1ght |
| 0 | M | 0.455 | 0.365 | 0.095 | 0.5140 |
| 1 | M | 0.350 | 0.265 | 0.090 | 0.2255 |
| 2 | F | 0.530 | 0.420 | 0.135 | 0.6770 |
| 3 | M | 0.440 | 0.365 | 0.125 | 0.5160 |
| 4 |  | 0.330 | 0.255 | 0.080 | 0.2050 |
| 4172 | F | 0.565 | 0.450 | 0.165 | 0.8870 |
| 4173 | M | 0.590 | 0.440 | 0.135 | 0.9660 |
| 4174 | M | 0.600 | 0.475 | 0.205 | 1.1760 |
| 4175 | F | 0.625 | 0.485 | 0.150 | 1.0945 |
| 4176 | M | 0.710 | 0.555 | 0.195 | 1.9485 |

4167 rows • 5 columns



y=df.iloc[:,5:]

|  |  |  |  |
| --- | --- | --- | --- |
| Shucked weight | Vlscera weight | Shell weight | age |
| 0.2245 | 0.1010 | 0.1500 |  |
| 1 0.0995 | 0.0485 | 0.0700 | 8.5 |
| 2 0.2565 | 0.1415 | 0.2100 |  |
| 0.2155 | 0.1140 | 0.1550 | 11.5 |
| 4 0.0895 | 0.0395 | 0.0550 |  |
| **4172** 0.3700 | 0.2390 | 0.2490 |  |
| **4173** 0.4390 | 0.2145 | 0.2605 | 11.5 |
| **4174** 0.5255 | 0.2875 | 0.3080 |  |
| **4175** 0.5310 | 0.2610 | 0.2960 | 11.5 |
| 4176 0.9455 | 0.3765 | 0.4950 |  |
| 4167 rows 4 columns |  |  |  |

# Train, Test, Split

from sk1earn.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)

Train and Test model

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x\_test [6 : 5 ] |  | | | | |
|  | Sex | Length | Diameter | Height | Nhole we1ght |
| 661 |  | 0.535 | 0.450 | 0.170 | 0.781 |
| 370 | F | 0.650 | 0.545 | 0.165 | 1.566 |
| 2272 | M | 0.635 | 0.510 | 0.210 | 1.598 |
| 1003 | M | 0.595 | 0.455 | 0.150 | 1.044 |
| 1145 | M | 0.580 | 0.455 | 0.195 | 1.859 |
| y\_test[0:5] |  |  |  |  |  |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Shucked we1ght | vlscera we1ght | Shell we1ght | age |
| 661 | 0.3055 | 0.1555 | 0.295 |  |
| 370 | 0.6645 | 0.3455 | 0.415 | 17.5 |
| **2272** | 0.6535 | 0.2835 | 0.580 |  |
| **1003** | 0.5180 | 0.2205 | 0.270 | 10.5 |
| **1145** | 0.9450 | 0.4260 | 0.441 |  |

#### Feature Scaling

from sklearn.preprocessing import StandardScaler ss=StandardScaler() x\_train=ss.fit\_transform(x\_train) mlrpred=mlr.predict(x\_test[B:9])

mlrpred

Performance measure

I-rom sklearn .metric s Import r2\_score r2\_s core(m1r . predict (x\_test) , y\_test )

0.5597133867640833