Abalone age prediction

In [9]:

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

In [ ]:

```
from google.colab import drive
drive.mount('/content/drive')
```

In [8]:

```
df=pd.read_csv("/content/drive/MyDrive/Colab Notebooks/abalone.csv")
```

In [10]:

```
df
```

Out[10]:

	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	0.1500	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7
...	...	...	...	...	...	...	...	...	...
4172	F	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11
4173	M	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	10
4174	M	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9
4175	F	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10
4176	M	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12

4177 rows × 9 columns

In [11]:

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

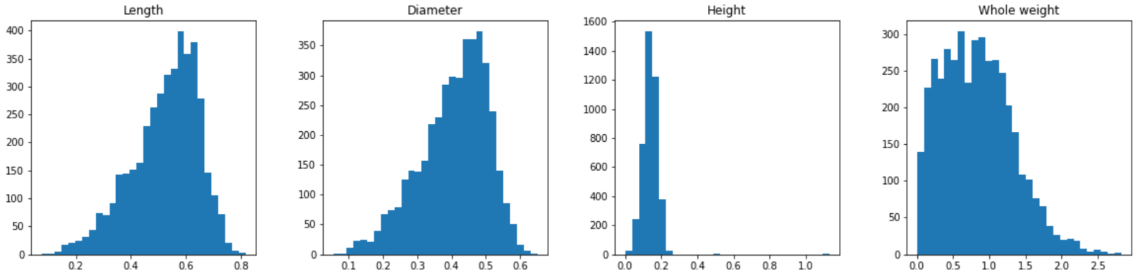
Univariate Analysis

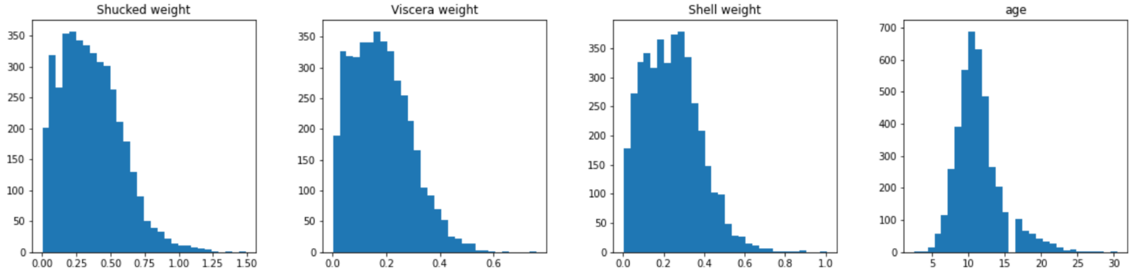
In [12]:

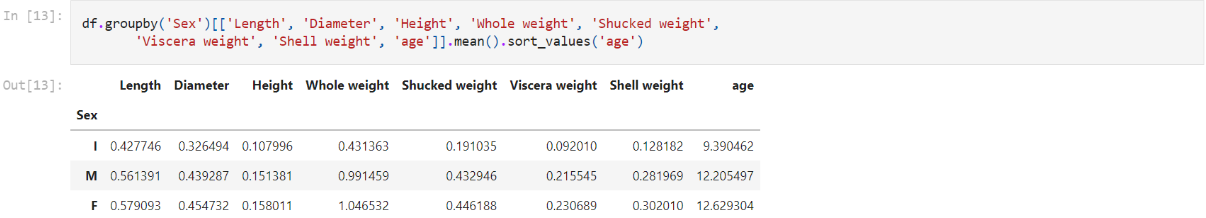
```
df.hist(figsize=(20,10), grid=False, layout=(2, 4), bins = 30)
```

Out[12]:

```
array([[,
        ,
        ,
        ],
       [,
        ,
        ,
        ]],
      dtype=object)
```





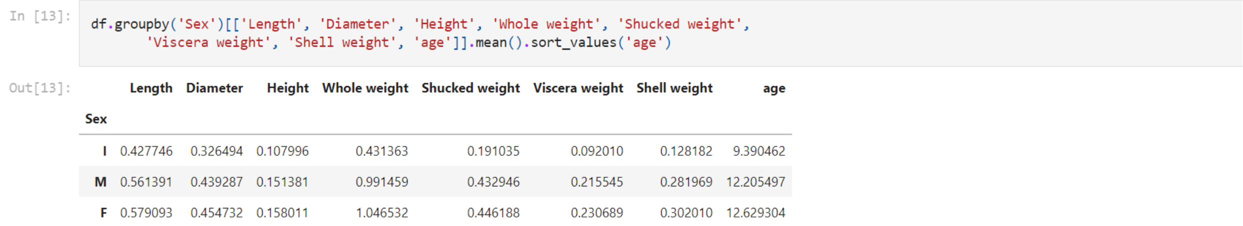


In [13]:

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

Out[13]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	age
Sex								
I	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



In [13]:

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

Out[13]:

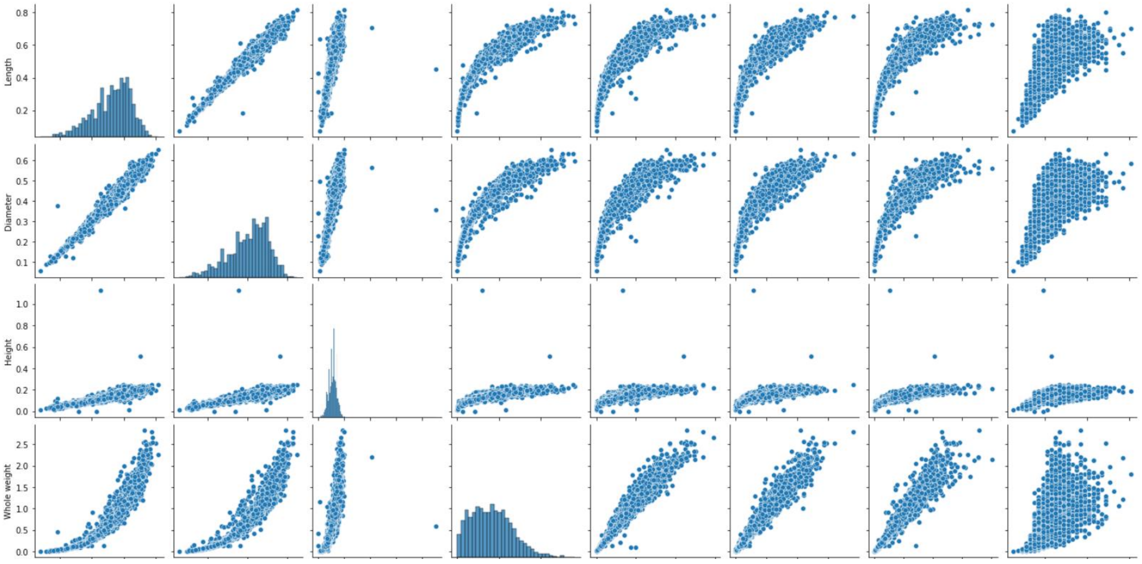
	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	age
Sex								
I	0.427746	0.326494	0.107996	0.431363	0.191035	0.092010	0.128182	9.390462
M	0.561391	0.439287	0.151381	0.991459	0.432946	0.215545	0.281969	12.205497
F	0.579093	0.454732	0.158011	1.046532	0.446188	0.230689	0.302010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

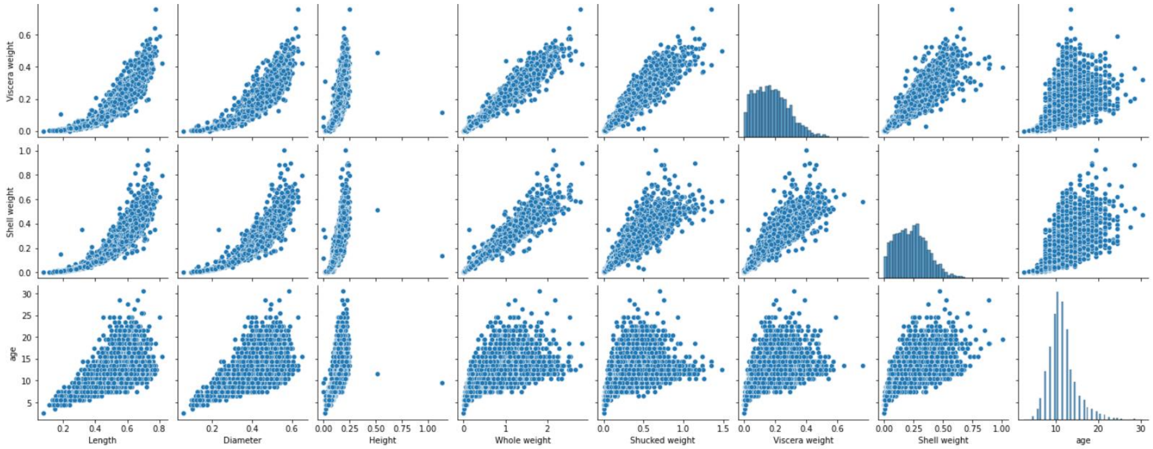
BiVariate Analysis

In [14]:

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

Out[14]:





Descriptive statistics

In [15]:

```
df.describe()
```

Out[15]:

	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

In [16]:

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

Out[16]:

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

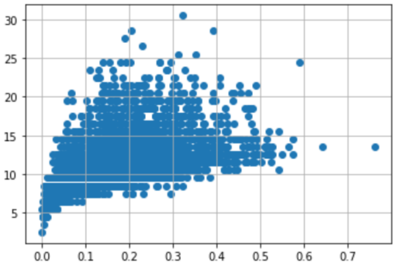
outlier handling

In [17]:

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

In [18]:

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

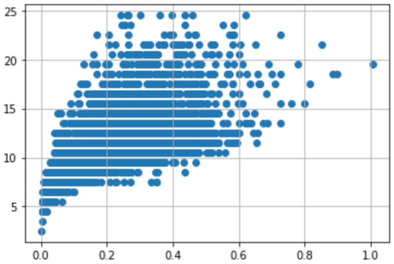


In [19]:

```
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)
```

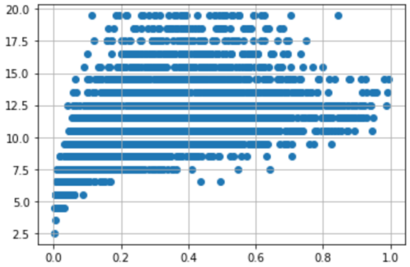
In [20]:

```
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)
```



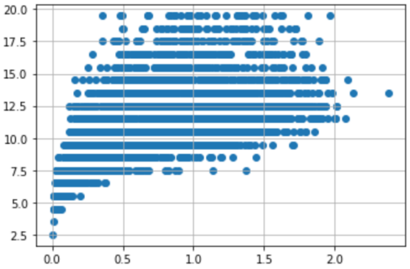
In [22]:

```
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)
```



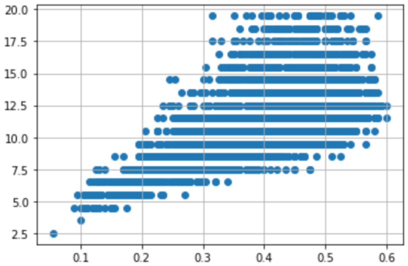
In [23]:

```
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)
```



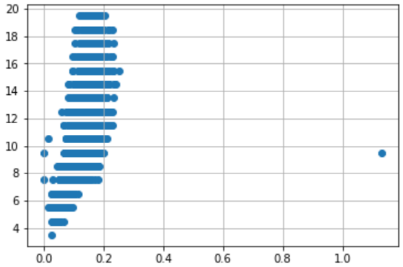
In [24]:

```
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)
```



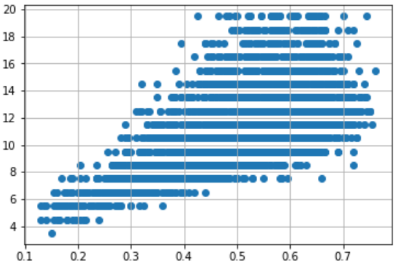
In [25]:

```
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)
```



In [26]:

```
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 columns

In [27]:
```
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: `np.object` is a deprecated alias for the builtin `object`. To s
ilence 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
```

In [28]:
```
numerical_features
```

Out[28]:
```
Index(['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
       'Viscera weight', 'Shell weight', 'age', 'Sex_F', 'Sex_I', 'Sex_M'],
      dtype='object')
```

In [29]:
```
categorical_features
```

Out[29]:
```
Index([], dtype='object')
```

Encoding

In [30]:
```
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
```

In [31]:
```
x=df.iloc[:,:5]
x
```

Out[31]:

	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

3995 rows × 5 columns

In [32]:
```
y=df.iloc[:,5:]
y
```

Out[32]:

	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

Train,test and split

In [33]:

```
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

In [34]:

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

Out[34]:

```
LinearRegression()
```

Train and Test Model

In [35]:

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

Out[35]:

	Length	Diameter	Height	Whole weight	Shucked weight
2004	0.375	0.275	0.085	0.220	0.1090
3712	0.705	0.530	0.170	1.564	0.6120
2987	0.555	0.405	0.190	1.406	0.6115
954	0.490	0.385	0.125	0.649	0.3200
998	0.590	0.455	0.145	1.063	0.5155

In [36]:

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

Out[36]:

	Viscera weight	Shell weight	age	Sex_F	Sex_I	Sex_M
2004	0.0500	0.0605	8.5	0	1	0
3712	0.3940	0.4400	11.5	1	0	0
2987	0.3420	0.3890	11.5	0	0	1
954	0.1240	0.1695	9.5	0	0	1
998	0.2445	0.2500	9.5	1	0	0

Feature Scaling

In [37]:

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

In [38]:

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

In [39]:

```
mlrpred
```

Out[39]:

```
array([[ 0.0476285 ,  0.06219962,  8.42477533,  0.07531777,  0.72782656,
         0.19685567],
       [ 0.34974881,  0.44424547, 14.13676118,  0.53984116, -0.05794599,
         0.51810483],
       [ 0.31257796,  0.37203018, 11.98437421,  0.41947151,  0.06396427,
         0.51656422],
       [ 0.13868341,  0.17801784, 10.0446649 ,  0.2430197 ,  0.42836323,
         0.32861707],
       [ 0.22960774,  0.277616  , 10.77070037,  0.33403542,  0.23525842,
         0.43070616],
       [ 0.14771098,  0.20785194, 11.64525528,  0.30198317,  0.37976566,
         0.31825117],
       [ 0.11725785,  0.16788838, 10.9720736 ,  0.2540245 ,  0.4548768 ,
         0.2910987 ],
       [ 0.19199145,  0.28850383, 13.84172837,  0.4654836 ,  0.15828322,
         0.37623318],
       [ 0.22850031,  0.2916064 , 11.99647354,  0.38440562,  0.17960126,
         0.43599312]])
```

Performance measure

In [40]:

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

Out[40]:

```
-3.3656939541439423
```