Assignment - 4

Data Analytics – Python Programming

Assignment Date	18 October 2022
Student Name	Ms. N.Ashifa
Student Roll Number	19ITA05
Maximum Marks	2 Marks

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [2]:
```
import os
os.chdir("C:/Users/User/Desktop/Datasets")
```

In [3]:
```
df=pd.read_csv('abalone.csv')
```

In [4]:
```
df.head()
```

Out[4]:

	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	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

In [5]:
```
df.describe()
```

Out[5]:

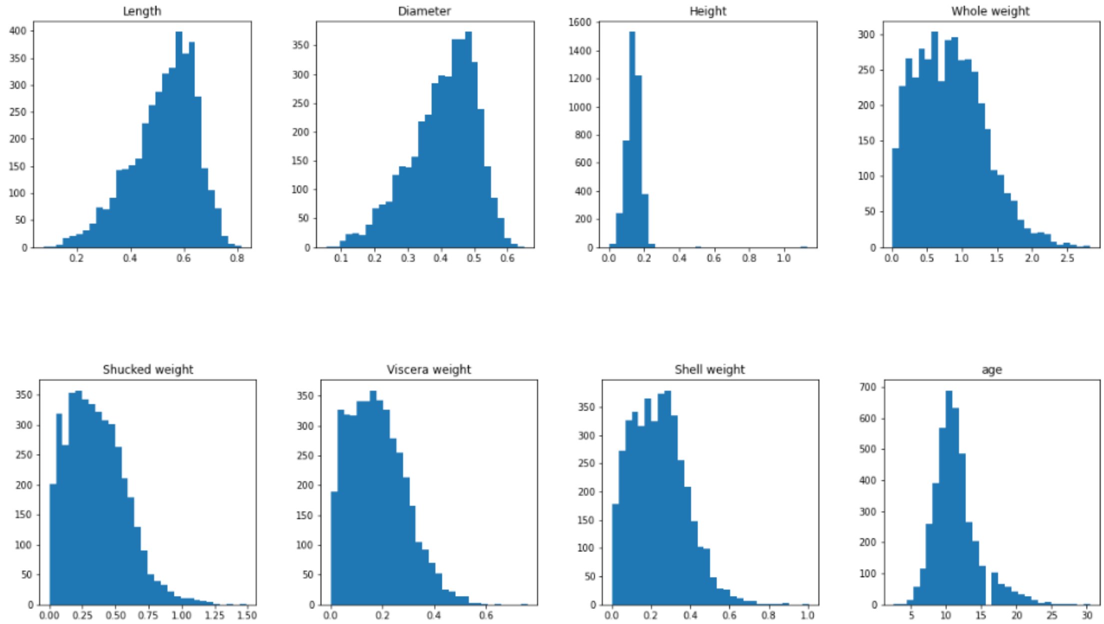
	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0.238831	9.933684
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.139203	3.224169
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.001500	1.000000
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.130000	8.000000
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.234000	9.000000
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.329000	11.000000
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.005000	29.000000

Exploratory Data Analysis

In [7]:
```
df['age'] = df['Rings']+1.5
df = df.drop('Rings', axis = 1)
```

In [8]:
```
df.hist(figsize=(20,10), grid=False, layout=(2, 4), bins = 30)
```

Out[8]:
```
array([[<AxesSubplot:title={'center':'Length'}>,
        <AxesSubplot:title={'center':'Diameter'}>,
        <AxesSubplot:title={'center':'Height'}>,
        <AxesSubplot:title={'center':'Whole weight'}>],
       [<AxesSubplot:title={'center':'Shucked weight'}>,
        <AxesSubplot:title={'center':'Viscera weight'}>,
        <AxesSubplot:title={'center':'Shell weight'}>,
        <AxesSubplot:title={'center':'age'}>]], dtype=object)
```



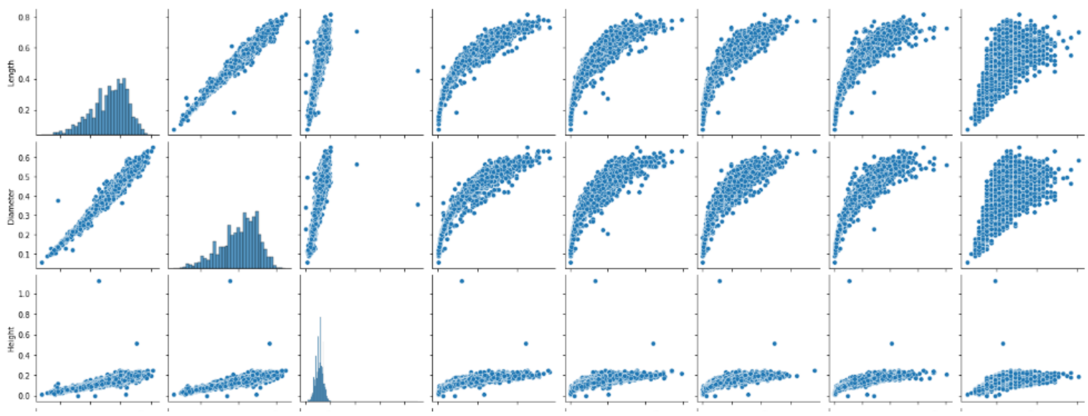
In [9]:
```
df.groupby('Sex')[['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
        'Viscera weight', 'Shell weight', 'age']].mean().sort_values('age')
```

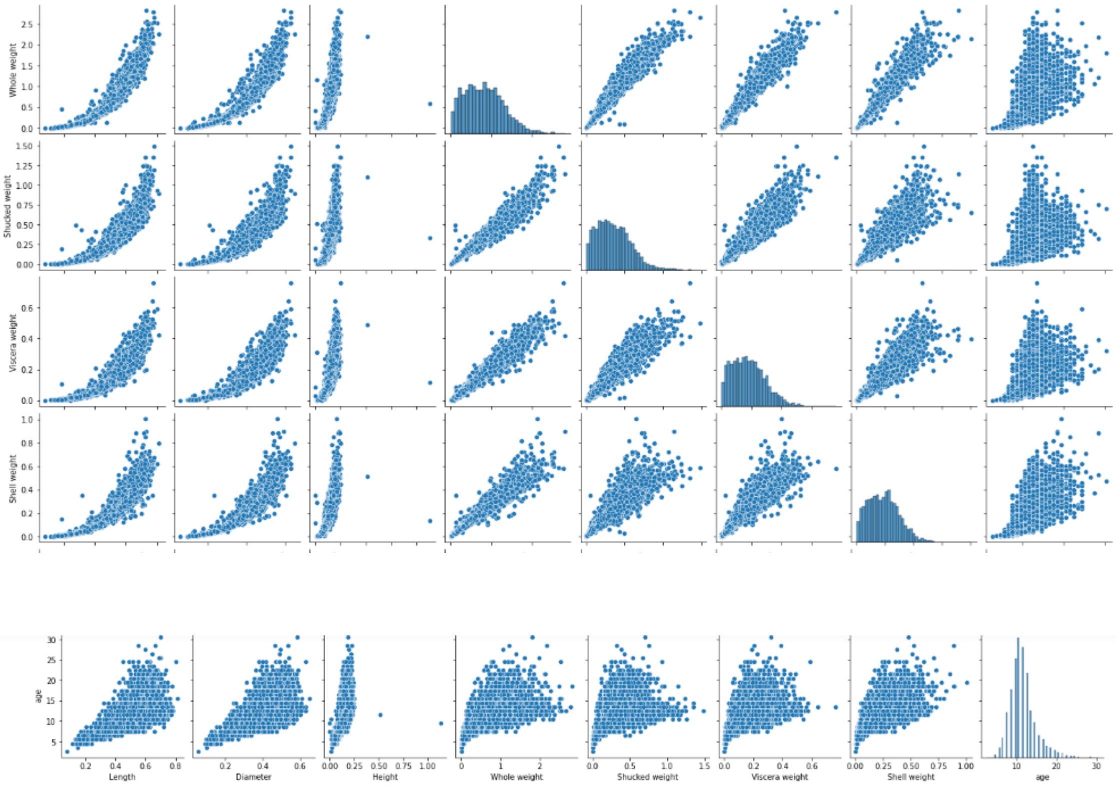
Out[9]:

	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 [10]:
```
numerical_features = df.select_dtypes(include = [np.number]).columns
sns.pairplot(df[numerical_features])
```

Out[10]: <seaborn.axisgrid.PairGrid at 0xec56105310>





In [11]:
```python
numerical_features = df.select_dtypes(include = [np.number]).columns
categorical_features = df.select_dtypes(include = [np.object]).columns
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_5940\3796453440.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
  categorical_features = df.select_dtypes(include = [np.object]).columns
```

In [12]:
```python
numerical_features
```

Out[12]:
```
Index(['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
       'Viscera weight', 'Shell weight', 'age'],
      dtype='object')
```

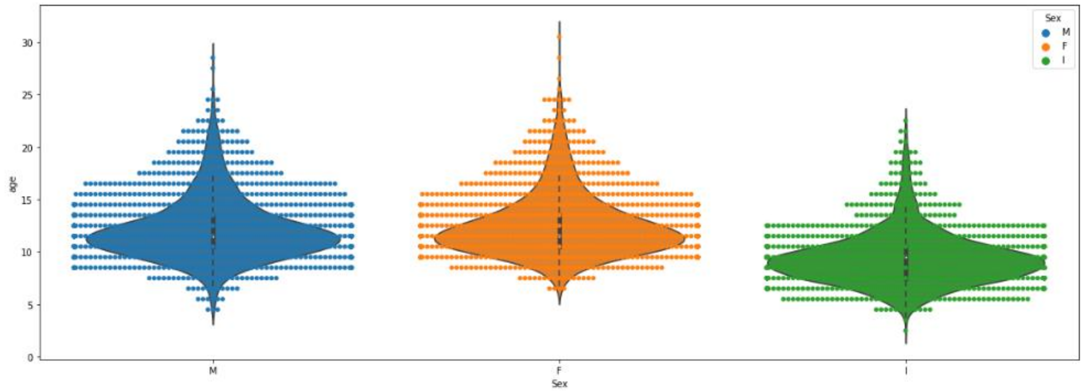
In [13]:
```python
categorical_features
```

Out[13]:
```
Index(['Sex'], dtype='object')
```

In [14]:
```python
plt.figure(figsize = (20,7))
sns.heatmap(df[numerical_features].corr(),annot = True)
```

Out[14]:
```
<AxesSubplot:>
```

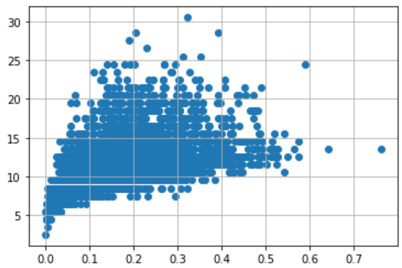
Out[15]: <AxesSubplot:xlabel='Sex', ylabel='age'>



Data Preprocessing

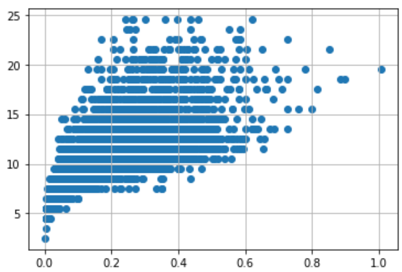
In [16]:
```
# outlier handling
df = pd.get_dummies(df)
dummy_df = df
```

In [17]:
```
var = 'Viscera weight'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)
```



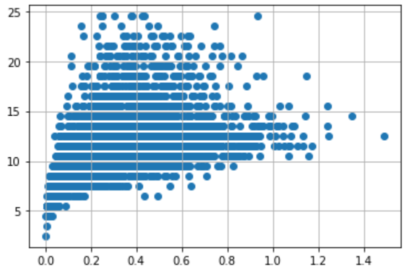
In [18]:
```
df.drop(df[(df['Viscera weight'] > 0.5) &
          (df['age'] < 20)].index, inplace = True)
df.drop(df[(df['Viscera weight']<0.5) & (
df['age'] > 25)].index, inplace = True)
```

In [19]:
```
var = 'Shell weight'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)
```



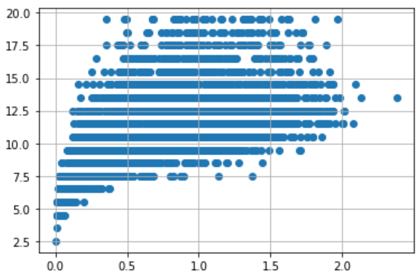
In [21]:
```
df.drop(df[(df['Shell weight'] > 0.6) &
          (df['age'] < 25)].index, inplace = True)
df.drop(df[(df['Shell weight']<0.8) & (
df['age'] > 25)].index, inplace = True)
```

In [20]:
```
var = 'Shucked weight'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)
```



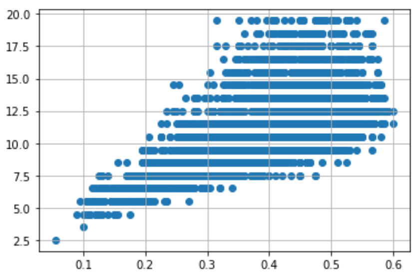
In [22]:
```
df.drop(df[(df['Shucked weight'] >= 1) &
           (df['age'] < 20)].index, inplace = True)
df.drop(df[(df['Viscera weight']<1) & (
df['age'] > 20)].index, inplace = True)
```

In [23]:
```
var = 'Whole weight'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)
```



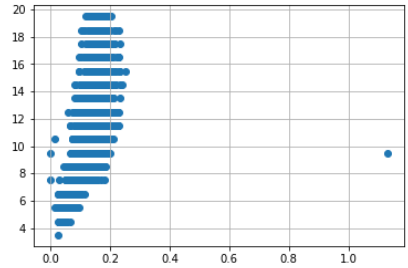
In [24]:
```
df.drop(df[(df['Whole weight'] >= 2.5) &
           (df['age'] < 25)].index, inplace = True)
df.drop(df[(df['Whole weight']<2.5) & (
df['age'] > 25)].index, inplace = True)
```

In [25]:
```
var = 'Diameter'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)
```



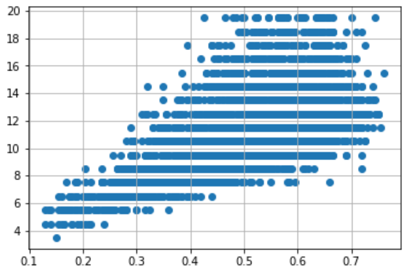
In [26]:
```
df.drop(df[(df['Diameter'] <0.1) &
           (df['age'] < 5)].index, inplace = True)
df.drop(df[(df['Diameter']<0.6) & (
df['age'] > 25)].index, inplace = True)
df.drop(df[(df['Diameter']>=0.6) & (
df['age'] < 25)].index, inplace = True)
```

In [27]:
```
var = 'Height'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)
```



In [28]:
```
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 [29]:
```
var = 'Length'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)
```



In [30]:
```
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)
```

Feature Selection and Standardization

In [31]:
```
X = df.drop('age', axis = 1)
y = df['age']
```

In [32]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.feature_selection import SelectKBest
```

In [33]:
```
standardScale = StandardScaler()
standardScale.fit_transform(X)

selectkBest = SelectKBest()
X_new = selectkBest.fit_transform(X, y)

X_train, X_test, y_train, y_test = train_test_split(X_new, y, test_size = 0.25)
```

Linear Regression

In [34]:
```
from sklearn.linear_model import LinearRegression
```

In [35]:
```
lm = LinearRegression()
lm.fit(X_train, y_train)
```

Out[35]:
```
LinearRegression()
```

In [36]:
```
y_train_pred = lm.predict(X_train)
y_test_pred = lm.predict(X_test)
```

In [37]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error
s = mean_squared_error(y_train, y_train_pred)
print('Mean Squared error of training set :%2f'%s)

p = mean_squared_error(y_test, y_test_pred)
print('Mean Squared error of testing set :%2f'%p)
```

```
Mean Squared error of training set :3.544594
Mean Squared error of testing set :3.618508
```

In [38]:
```
from sklearn.metrics import r2_score
s = r2_score(y_train, y_train_pred)
print('R2 Score of training set:%.2f'%s)

p = r2_score(y_test, y_test_pred)
print('R2 Score of testing set:%.2f'%p)u
```

```
R2 Score of training set:0.53
R2 Score of testing set:0.54
```