# 1. Load Data

In [2]:
```
# loading libraries
import numpy as np  # for mathematical operations & linear algebra
import pandas as pd  # for data processing
import matplotlib.pyplot as plt  # for plotting graphs & figures
import seaborn as sns  # for plotting graphs & figures
from scipy.stats import skew
%matplotlib inline

from sklearn.preprocessing import  StandardScaler  # for pre-processing
from sklearn.model_selection import  train_test_split, cross_val_score  # for
from sklearn.feature_selection import SelectKBest   # for pre-processing
from sklearn.metrics import r2_score, mean_squared_error   # for evaluation

 # machine learning models
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import  RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import  GradientBoostingRegressor
from sklearn.linear_model import  Ridge
from sklearn.svm import SVR
import warnings  # for logging
warnings.filterwarnings("ignore", category=DeprecationWarning)

import os  #  for os operations
```

In [3]:
```
# reading the dataset from current directory
data = pd.read_excel('Abalone Data Set.xlsx', header=None)

# settings column names
columns = ['Sex', 'Length', 'Diameter', 'Height',
           'Whole weight', 'Shucked weight', 'Viscera weight',
           'shell weight', 'Rings']
data.columns = columns
```

In [4]:
```
# settings target variable 'Age' which is apperently rings + 1.5
data['age'] = data['Rings'] + 1.5
data.drop('Rings', axis = 1, inplace = True)  # removing rings column as we c
```

In [5]:
```
print('This dataset has {} observations with {} features.'.format(data.shape
```

```
This dataset has 4177 observations with 9 features.
```

In [6]:
```
# printing 5 rows to see dataset is loaded successfully
data.head()
```

Out[6]:

| | Sex | Length | Diameter | Height | Whole weight | Shucked weight | Viscera weight | shell weight | age |
|---|---|---|---|---|---|---|---|---|---|
| **0** | M | 0.455 | 0.365 | 0.095 | 0.5140 | 0.2245 | 0.1010 | 0.150 | 16.5 |
| **1** | M | 0.350 | 0.265 | 0.090 | 0.2255 | 0.0995 | 0.0485 | 0.070 | 8.5 |
| **2** | F | 0.530 | 0.420 | 0.135 | 0.6770 | 0.2565 | 0.1415 | 0.210 | 10.5 |
| **3** | M | 0.440 | 0.365 | 0.125 | 0.5160 | 0.2155 | 0.1140 | 0.155 | 11.5 |
| **4** | I | 0.330 | 0.255 | 0.080 | 0.2050 | 0.0895 | 0.0395 | 0.055 | 8.5 |

# 2. Dataset Information

In [7]:
```
# printing column names
data.columns
```

Out[7]:
```
Index(['Sex', 'Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weig
ht',
       'Viscera weight', 'shell weight', 'age'],
      dtype='object')
```

In [8]:
```
# printing other information
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
Sex               4177 non-null object
Length            4177 non-null float64
Diameter          4177 non-null float64
Height            4177 non-null float64
Whole weight      4177 non-null float64
Shucked weight    4177 non-null float64
Viscera weight    4177 non-null float64
shell weight      4177 non-null float64
age               4177 non-null float64
dtypes: float64(8), object(1)
memory usage: 293.8+ KB
```

In [9]:
```
# printing statistical inforamtion
data.describe()
```

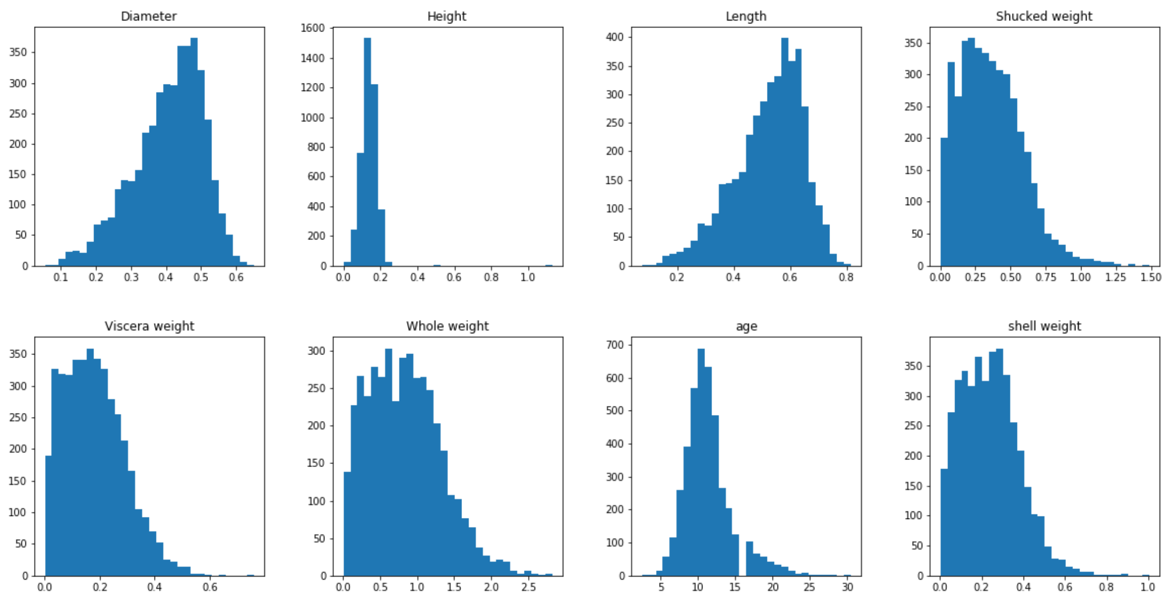
Out[9]:

| | Length | Diameter | Height | Whole weight | Shucked weight | Viscera weight | shell w |
|---|---|---|---|---|---|---|---|
| **count** | 4177.000000 | 4177.000000 | 4177.000000 | 4177.000000 | 4177.000000 | 4177.000000 | 4177.00 |
| **mean** | 0.523992 | 0.407881 | 0.139516 | 0.828742 | 0.359367 | 0.180594 | 0.23 |
| **std** | 0.120093 | 0.099240 | 0.041827 | 0.490389 | 0.221963 | 0.109614 | 0.13 |
| **min** | 0.075000 | 0.055000 | 0.000000 | 0.002000 | 0.001000 | 0.000500 | 0.00 |
| **25%** | 0.450000 | 0.350000 | 0.115000 | 0.441500 | 0.186000 | 0.093500 | 0.13 |
| **50%** | 0.545000 | 0.425000 | 0.140000 | 0.799500 | 0.336000 | 0.171000 | 0.23 |
| **75%** | 0.615000 | 0.480000 | 0.165000 | 1.153000 | 0.502000 | 0.253000 | 0.32 |
| **max** | 0.815000 | 0.650000 | 1.130000 | 2.825500 | 1.488000 | 0.760000 | 1.00 |

# Data preparation and visualization

In [10]:
```
# plotting data distributions for different features
data.hist(figsize=(20,10), grid=False, layout=(2, 4), bins = 30)
```

Out[10]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000017A8CA7A400
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000017A8CA95630
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000017A8CABE898
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000017A8CD88B00
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x0000017A8CDB0D68
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000017A8CDD7FD0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000017A8CE0B278
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x0000017A8CE315C0
>]],
      dtype=object)
```




In [11]:
```
# feature information
numerical_features = data.select_dtypes(include=[np.number]).columns  # all f
categorical_features = data.select_dtypes(include=[np.object]).columns  # cha
```

In [12]:
```
numerical_features  # decimals
```

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

In [14]:
```
categorical_features  # Sex => 'M' (Male), 'F' (Female), 'I' (Immature)
```

Out[14]:
```
Index(['Sex'], dtype='object')
```

In [15]:
```
# finding skewness in different feature, so in order to understand the data d
skew_values = skew(data[numerical_features], nan_policy = 'omit')
dummy = pd.concat([pd.DataFrame(list(numerical_features), columns=['Features'
           pd.DataFrame(list(skew_values), columns=['Skewness degree'])], axi
dummy.sort_values(by = 'Skewness degree' , ascending = False)
```

Out[15]:

| | Features | Skewness degree |
|---|---|---|
| **2** | Height | 3.127694 |
| **7** | age | 1.113702 |
| **4** | Shucked weight | 0.718840 |
| **6** | shell weight | 0.620704 |
| **5** | Viscera weight | 0.591640 |
| **3** | Whole weight | 0.530768 |
| **1** | Diameter | -0.608979 |
| **0** | Length | -0.639643 |

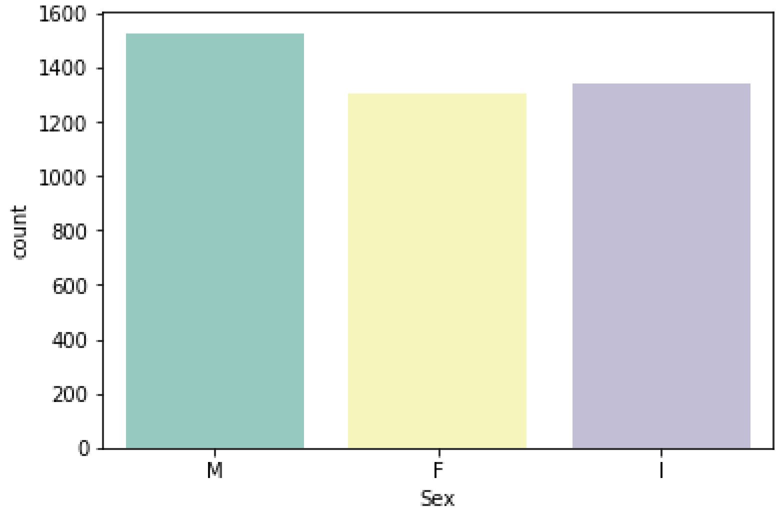
In [13]:
```
# checking & remocing missing values
missing_values = data.isnull().sum().sort_values(ascending = False)
percentage_missing_values = (missing_values/len(data))*100
pd.concat([missing_values, percentage_missing_values], axis = 1, keys= ['Miss
```

Out[13]:

| | Missing values | % Missing |
|---|---|---|
| **age** | 0 | 0.0 |
| **Shell weight** | 0 | 0.0 |
| **Viscera weight** | 0 | 0.0 |
| **Shucked weight** | 0 | 0.0 |
| **Whole weight** | 0 | 0.0 |
| **Height** | 0 | 0.0 |
| **Diameter** | 0 | 0.0 |
| **Length** | 0 | 0.0 |
| **Sex** | 0 | 0.0 |

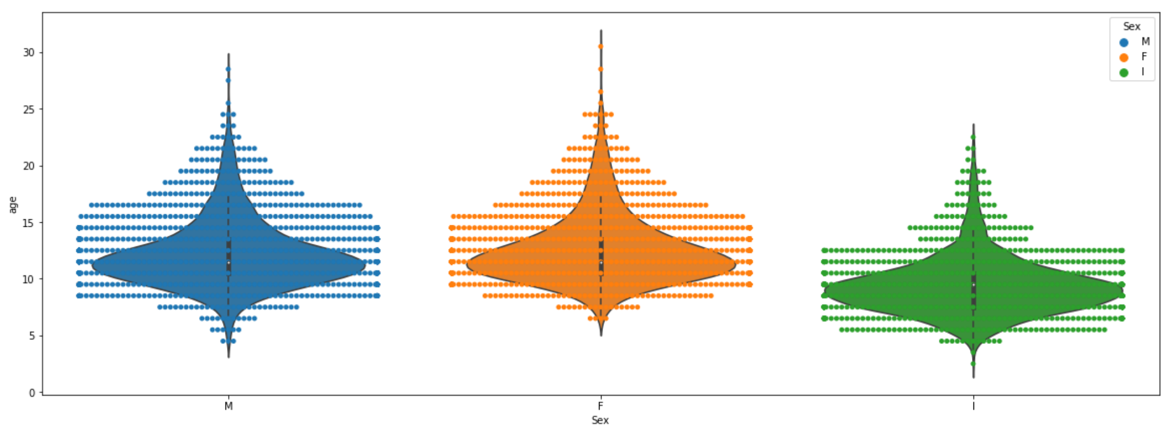
In [14]:
```
# plotting instance count based categorical feature 'Sex'
sns.countplot(x = 'Sex', data = data, palette="Set3")
```

Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x7f065ba94cf8>




In [16]:
```
# finding out age distribution such as the majority of data
plt.figure(figsize = (20,7))
sns.swarmplot(x = 'Sex', y = 'age', data = data, hue = 'Sex')
sns.violinplot(x = 'Sex', y = 'age', data = data)
```

Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x17a8cfdca20>



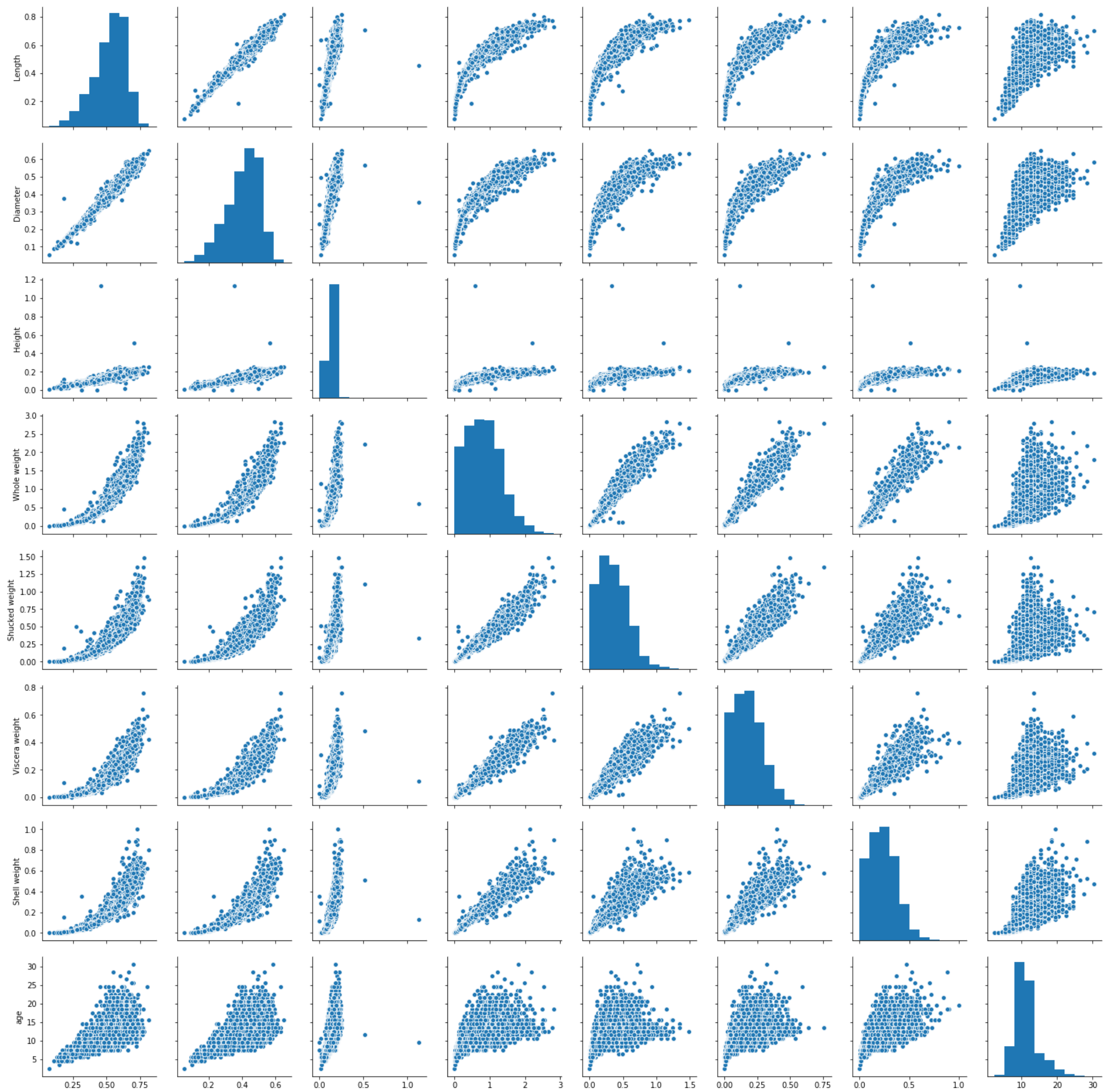
In [19]:
```
# finding out how other features co-related to 'Sex'
data.groupby('Sex')[['Length', 'Diameter', 'Height',
           'Whole weight', 'Shucked weight', 'Viscera weight',
           'shell weight', 'age']].mean().sort_values('age')
```

Out[19]:

| Sex | Length | Diameter | Height | Whole weight | Shucked weight | Viscera weight | shell weight | age |
|---|---|---|---|---|---|---|---|---|
| **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 |

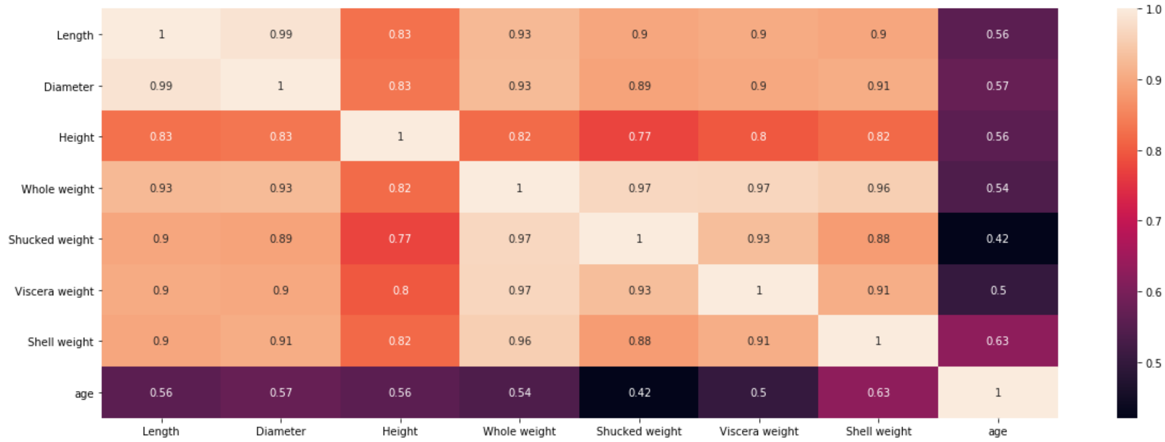
In [17]:
```
# bivariate analysis to see the co relations between features
sns.pairplot(data[numerical_features])
```

Out[17]: <seaborn.axisgrid.PairGrid at 0x7f065f74b438>



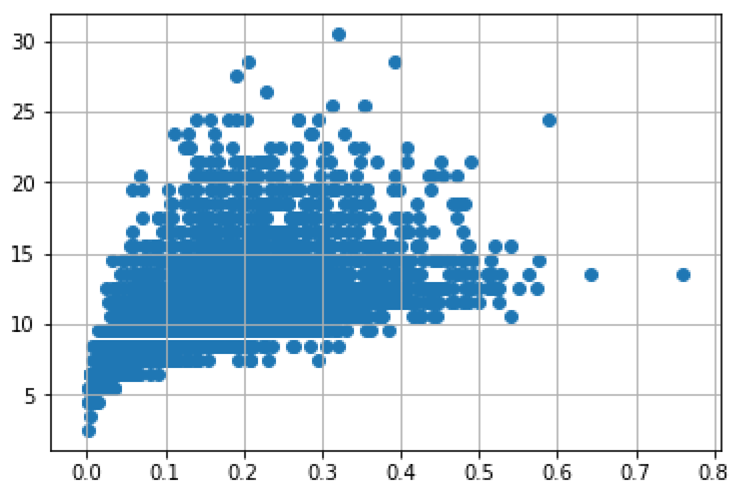
In [18]:
```
# plotting heatmap for numerical features to see the co-relations
plt.figure(figsize=(20,7))
sns.heatmap(data[numerical_features].corr(), annot=True)
```

Out[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f06546a2908>
```




In [24]:
```
data = pd.get_dummies(data)
dummy_data = data.copy()
```

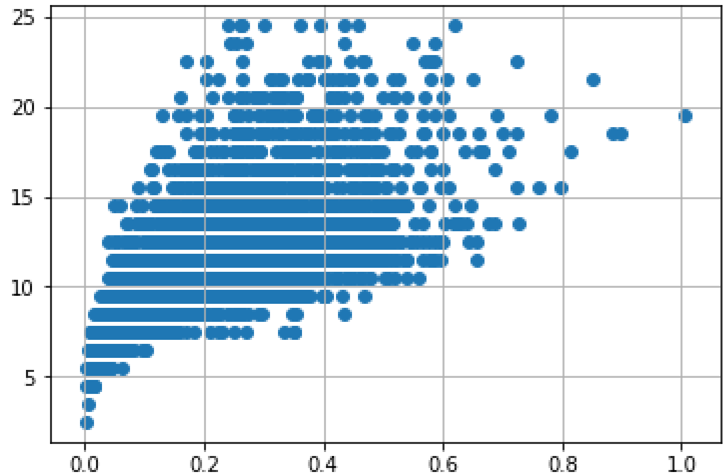
In [25]:
```
# scatter plot to see data points
var = 'Viscera weight'
plt.scatter(x = data[var], y = data['age'],)
plt.grid(True)
```



In [26]:
```
# removing outliers
data.drop(data[(data['Viscera weight']> 0.5) & (data['age'] < 20)].index, inp
data.drop(data[(data['Viscera weight']<0.5) & (data['age'] > 25)].index, inpl
```

In [30]:

```
# scatter plot to see data points
var = 'shell weight'
plt.scatter(x = data[var], y = data['age'],)
plt.grid(True)
```

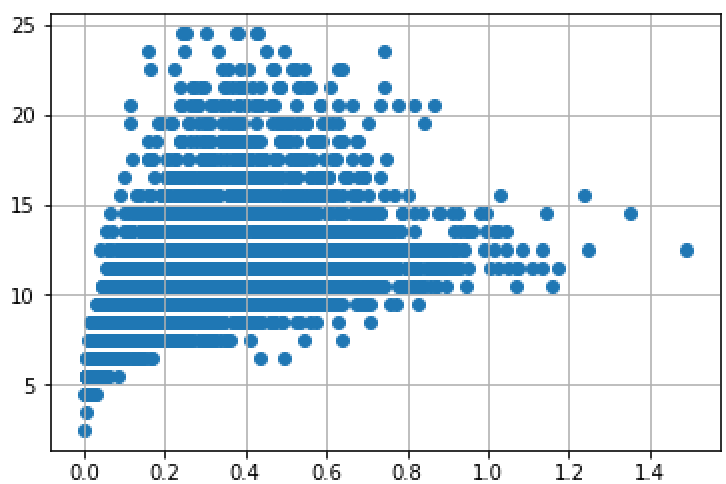


In [31]:

```
# removing outliers
data.drop(data[(data['shell weight']> 0.6) & (data['age'] < 25)].index, inpla
data.drop(data[(data['shell weight']<0.8) & (data['age'] > 25)].index, inplac
```

In [32]:

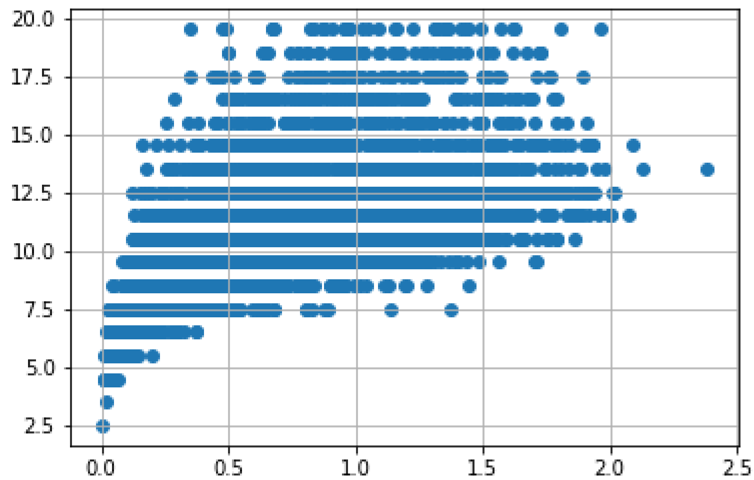
```
# scatter plot to see data points
var = 'Shucked weight'
plt.scatter(x = data[var], y = data['age'],)
plt.grid(True)
```



In [33]:

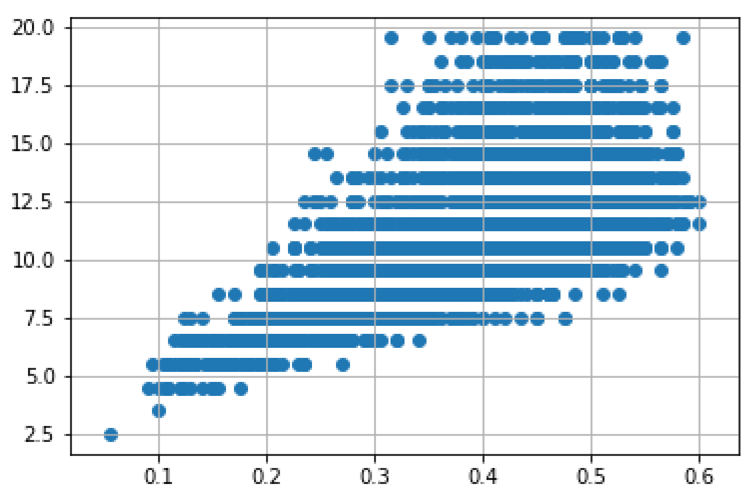
```
# removing outliers
data.drop(data[(data['Shucked weight']>= 1) & (data['age'] < 20)].index, inpl
data.drop(data[(data['Shucked weight']<1) & (data['age'] > 20)].index, inplac
```

In [34]:
```
# scatter plot to see data points
var = 'Whole weight'
plt.scatter(x = data[var], y = data['age'],)
plt.grid(True)
```



In [35]:
```
# removing outliers
data.drop(data[(data['Whole weight']>= 2.5) & (data['age'] < 25)].index, inpl
data.drop(data[(data['Whole weight']<2.5) & (data['age'] > 25)].index, inplac
```

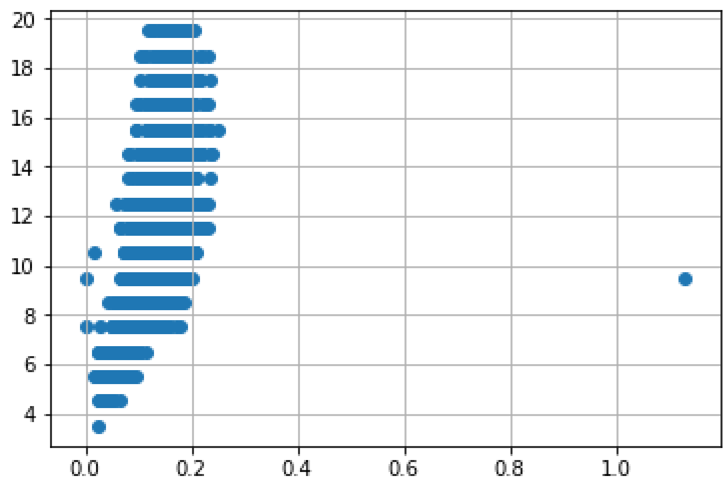
In [36]:
```
# scatter plot to see data points
var = 'Diameter'
plt.scatter(x = data[var], y = data['age'],)
plt.grid(True)
```



In [37]:
```
# removing outliers
data.drop(data[(data['Diameter']<0.1) & (data['age'] < 5)].index, inplace=Tru
data.drop(data[(data['Diameter']<0.6) & (data['age'] > 25)].index, inplace=Tr
data.drop(data[(data['Diameter']>=0.6) & (data['age']< 25)].index, inplace=Tr
```

In [38]:

```
# scatter plot to see data points
var = 'Height'
plt.scatter(x = data[var], y = data['age'],)
plt.grid(True)
```

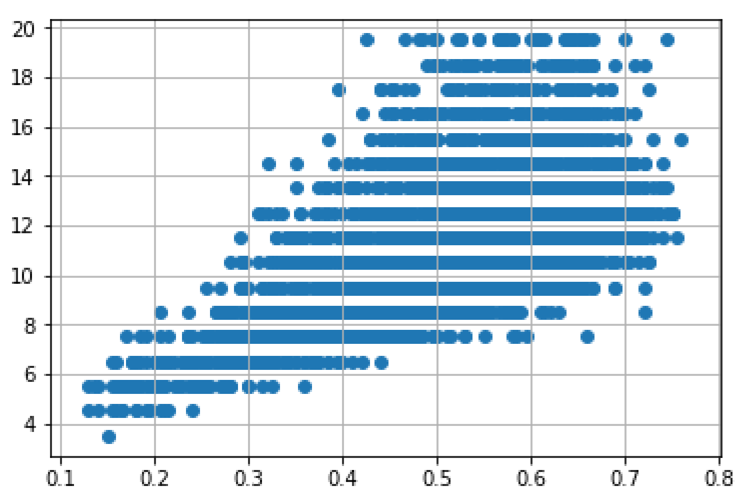



In [39]:

```
# removing outliers
data.drop(data[(data['Height']>0.4) & (data['age'] < 15)].index, inplace=True
data.drop(data[(data['Height']<0.4) & (data['age'] > 25)].index, inplace=True
```

In [40]:

```
# scatter plot to see data points
var = 'Length'
plt.scatter(x = data[var], y = data['age'],)
plt.grid(True)
```




In [41]:

```
# removing outliers
data.drop(data[(data['Length']<0.1) & (data['age'] < 5)].index, inplace=True)
data.drop(data[(data['Length']<0.8) & (data['age'] > 25)].index, inplace=True
data.drop(data[(data['Length']>=0.8) & (data['age']< 25)].index, inplace=True
```

# Preprocessing, Modeling, Evaluation

In [42]:

```
# splitting features and target feature from dataset
X = data.drop('age', axis = 1)
y = data['age']
```

In [43]:

```
# spiiting the dataset into train test sets for fitting in the model
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 = 0.2
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\preprocessing\data.py:62
5: DataConversionWarning: Data with input dtype uint8, float64 were all con
verted to float64 by StandardScaler.
  return self.partial_fit(X, y)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:462: DataConvers
ionWarning: Data with input dtype uint8, float64 were all converted to floa
t64 by StandardScaler.
  return self.fit(X, **fit_params).transform(X)
```

In [44]:

```
np.random.seed(10)
def rmse_cv(model, X_train, y):
    rmse =- (cross_val_score(model, X_train, y, scoring='neg_mean_squared_err
    return(rmse*100)

# different machine learning models
models = [LinearRegression(),
             Ridge(),
             SVR(),
             RandomForestRegressor(),
             GradientBoostingRegressor(),
             KNeighborsRegressor(n_neighbors = 4),]

names = ['LR','Ridge','svm','GNB','RF','GB','KNN']

# model performance
for model,name in zip(models,names):
    score = rmse_cv(model,X_train,y_train)
    print("{}    : {:.6f}, {:4f}".format(name,score.mean(),score.std()))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:24
6: FutureWarning: The default value of n_estimators will change from 10 i
n version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:24
6: FutureWarning: The default value of n_estimators will change from 10 i
n version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:24
6: FutureWarning: The default value of n_estimators will change from 10 i
n version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:24
6: FutureWarning: The default value of n_estimators will change from 10 i
n version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)

GNB    : 376.079123, 32.418777
RF    : 335.557481, 35.006948
GB    : 376.665185, 26.405289
```