**import libraries**

In [1]:

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**%matplotlib** inline

**import** seaborn **as** sns

In [2]:df **=** pd**.**read\_csv('../input/abalone.csv')

In [3]:df**.**head()

Out[3]:

|  | **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 [4]: df**.**describe()

Out[4]:

| **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 |

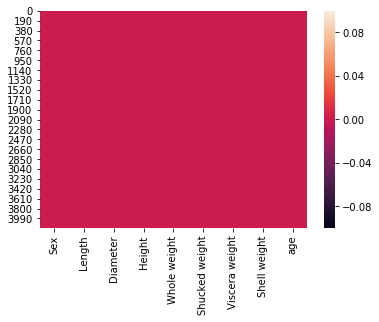
In [5]: df['age'] **=** df['Rings']**+**1.5

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

**EDA**

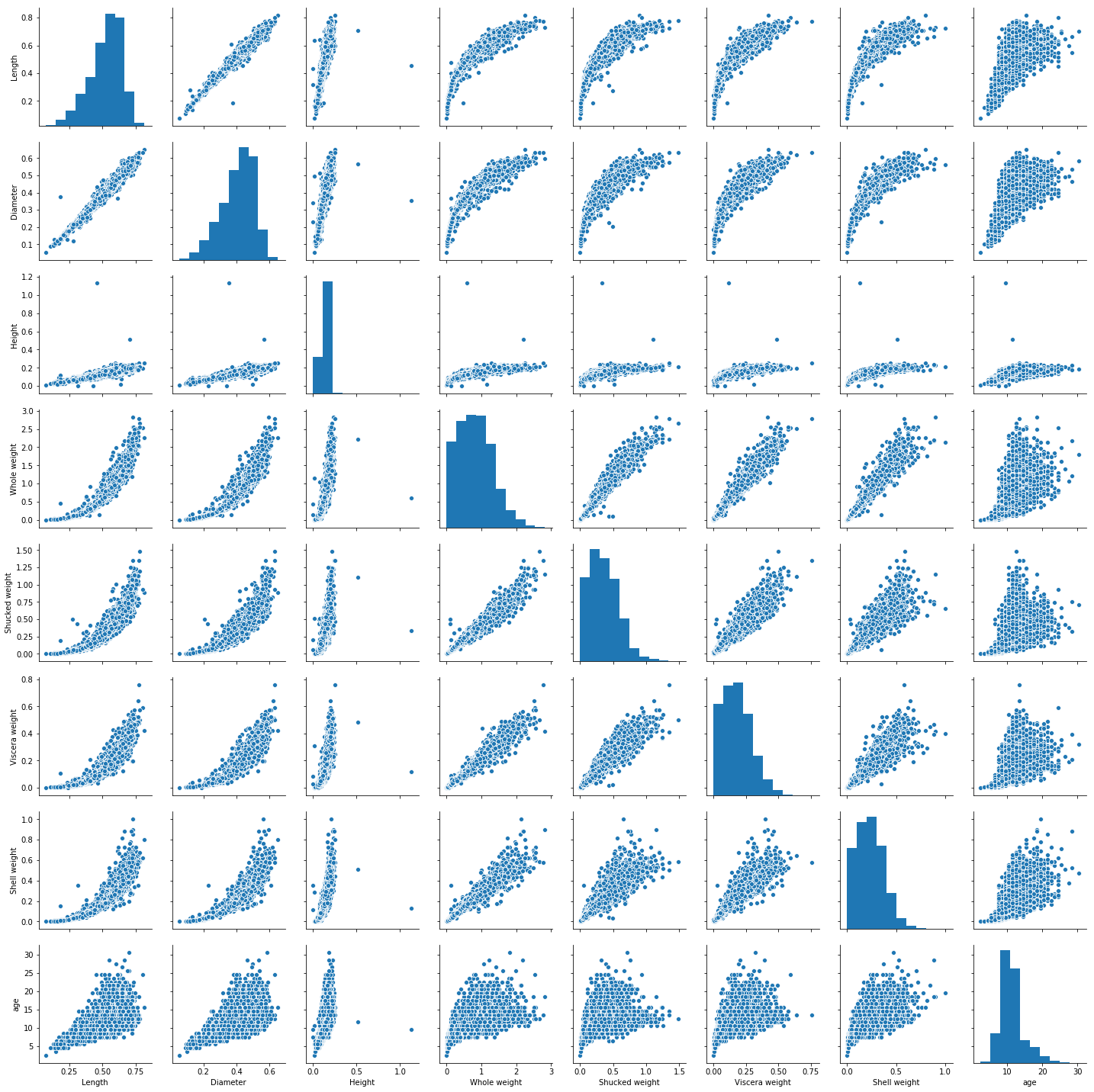
In [6]: sns**.**heatmap(df**.**isnull())

Out[6]: <matplotlib.axes.\_subplots.AxesSubplot at 0x7fcc468da358>



In[7]: sns**.**pairplot(df)

Out[7]: <seaborn.axisgrid.PairGrid at 0x7fcc3caa8160>



In[8]: df**.**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]: numerical\_features **=** df**.**select\_dtypes(include **=** [np**.**number])**.**columns

categorical\_features **=** df**.**select\_dtypes(include **=** [np**.**object])**.**columns

In [10]: numerical\_features

Out[10]: Index(['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight', 'Viscera weight', 'Shell weight', 'age'],dtype='object')

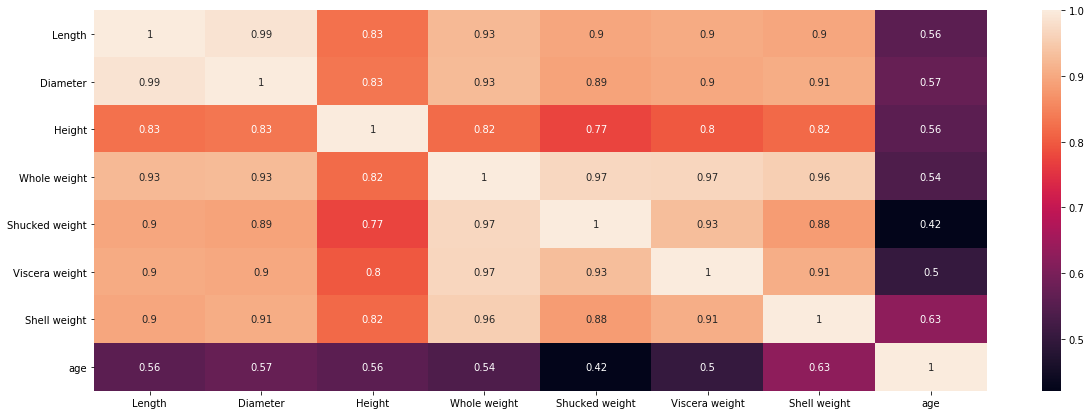
In [11]: categorical\_features

Out[11]: Index(['Sex'], dtype='object')

In[12]: plt**.**figure(figsize **=** (20,7))

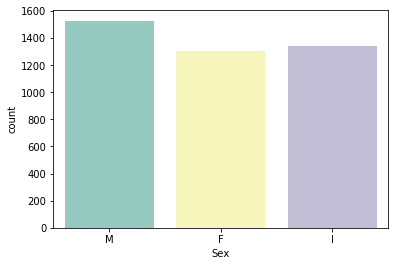
sns**.**heatmap(df[numerical\_features]**.**corr(),annot **=** **True**)

Out[12]: <matplotlib.axes.\_subplots.AxesSubplot at 0x7fcc29714dd8>



In[13]: sns**.**countplot(x **=** 'Sex', data **=** df, palette **=** 'Set3')

Out[13]: <matplotlib.axes.\_subplots.AxesSubplot at 0x7fcc26ba6748>



Male : age majority lies in between 7.5 years to 19 years

Female: age majority lies in between 8 years to 19 years

Immature: age majority lies in between 6 years to < 10 years

**Data Preprocessing**

In [15]:

*# outlier handling*

df **=** pd**.**get\_dummies(df)

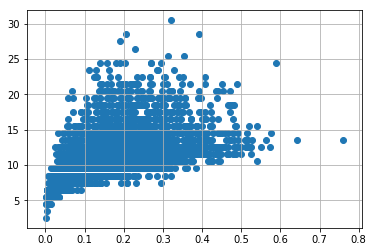
dummy\_df **=** df

In [16]:

var **=** 'Viscera weight'

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

plt**.**grid(**True**)



In[17]: df**.**drop(df[(df['Viscera weight'] **>** 0.5) **&**

(df['age'] **<** 20)]**.**index, inplace **=** **True**)

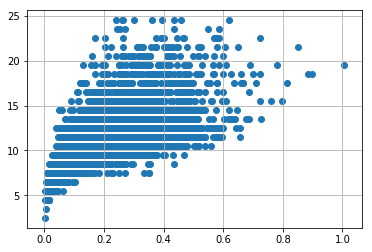
df**.**drop(df[(df['Viscera weight']**<**0.5) **&** (

df['age'] **>** 25)]**.**index, inplace **=** **True**)

In [18]:var **=** 'Shell weight'

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

plt**.**grid(**True**)



In[19]:df**.**drop(df[(df['Shell weight'] **>** 0.6) **&**

(df['age'] **<** 25)]**.**index, inplace **=** **True**)

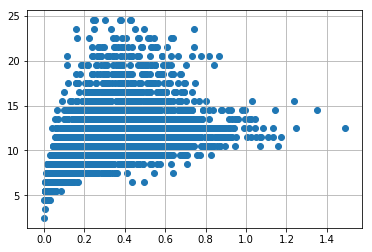
df**.**drop(df[(df['Shell weight']**<**0.8) **&** (

df['age'] **>** 25)]**.**index, inplace **=** **True**)

In [20]: var **=** 'Shucked weight'

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

plt**.**grid(**True**)



In[21]: df**.**drop(df[(df['Shucked weight'] **>=** 1) **&**

(df['age'] **<** 20)]**.**index, inplace **=** **True**)

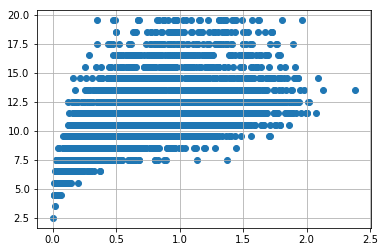
df**.**drop(df[(df['Viscera weight']**<**1) **&** (

df['age'] **>** 20)]**.**index, inplace **=** **True**)

In [22]: var **=** 'Whole weight'

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

plt**.**grid(**True**)



In[23]: df**.**drop(df[(df['Whole weight'] **>=** 2.5) **&**

(df['age'] **<** 25)]**.**index, inplace **=** **True**)

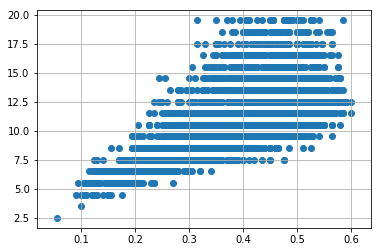
df**.**drop(df[(df['Whole weight']**<**2.5) **&** (

df['age'] **>** 25)]**.**index, inplace **=** **True**)

In [24]: var **=** 'Diameter'

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

plt**.**grid(**True**)



In[25]: 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**)

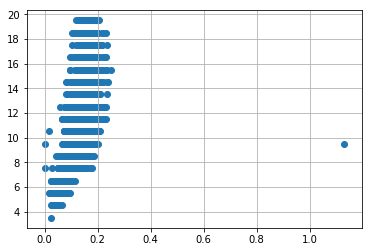
df**.**drop(df[(df['Diameter']**>=**0.6) **&** (

df['age'] **<** 25)]**.**index, inplace **=** **True**)

In [26]: var **=** 'Height'

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

plt**.**grid(**True**)



In[27]:df**.**drop(df[(df['Height'] **>** 0.4) **&**

(df['age'] **<** 15)]**.**index, inplace **=** **True**)

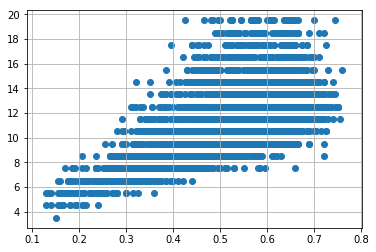
df**.**drop(df[(df['Height']**<**0.4) **&** (

df['age'] **>** 25)]**.**index, inplace **=** **True**)

In [28]: var **=** 'Length'

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

plt**.**grid(**True**)



In[29]: 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 [30]: X **=** df**.**drop('age', axis **=** 1)

y **=** df['age']

In [31]: **from** sklearn.preprocessing **import** StandardScaler

**from** sklearn.model\_selection **import** train\_test\_split, cross\_val\_score

**from** sklearn.feature\_selection **import** SelectKBest

In [32]:

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)

/opt/conda/lib/python3.6/site-packages/sklearn/preprocessing/data.py:645: DataConversionWarning: Data with input dtype uint8, float64 were all converted to float64 by StandardScaler.

return self.partial\_fit(X, y)

/opt/conda/lib/python3.6/site-packages/sklearn/base.py:464: DataConversionWarning: Data with input dtype uint8, float64 were all converted to float64 by StandardScaler.

# Model Selection

**1)Linear regression**

In [33]: **from** sklearn.linear\_model **import** LinearRegression

In [34]: lm **=** LinearRegression()

lm**.**fit(X\_train, y\_train)

Out[34]: LinearRegression(copy\_X=True, fit\_intercept=True, n\_jobs=None,

normalize=False)

In [35]: y\_train\_pred **=** lm**.**predict(X\_train)

y\_test\_pred **=** lm**.**predict(X\_test)

In [36]:

**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 :3.551893

Mean Squared error of testing set :3.577687

In [37]:

**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**)

return self.fit(X, \*\*fit\_params).transform(X)

R2 Score of training set:0.54

R2 Score of testing set:0.53

**2)Ridge**

In [38]: **from** sklearn.linear\_model **import** Ridge

In [39]:

ridge\_mod **=** Ridge(alpha**=**0.01, normalize**=True**)

ridge\_mod**.**fit(X\_train, y\_train)

ridge\_mod**.**fit(X\_test, y\_test)

ridge\_model\_pred **=** ridge\_mod**.**predict(X\_test)

ridge\_mod**.**score(X\_train, y\_train)

Out[39]: 0.5307346478347332

In [40]: ridge\_mod**.**score(X\_test, y\_test)

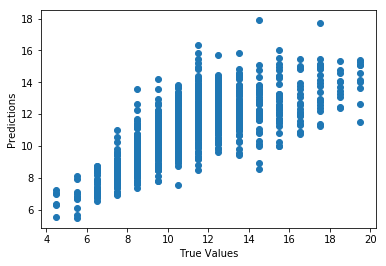
Out[40]: 0.5272608729607438

In [41]: plt**.**scatter(y\_test, ridge\_model\_pred)

plt**.**xlabel('True Values')

plt**.**ylabel('Predictions')

Out[41]: Text(0, 0.5, 'Predictions')



**3) RandomForestRegression**

In [46]: **from** sklearn.ensemble **import** RandomForestRegressor

In [47]: regr **=** RandomForestRegressor(max\_depth**=**2, random\_state**=**0,

n\_estimators**=**100)

In [48]: regr**.**fit(X\_train, y\_train)

regr**.**fit(X\_test, y\_test)

Out[48]:

RandomForestRegressor(bootstrap=True, criterion='mse', max\_depth=2,

max\_features='auto', max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, n\_estimators=100, n\_jobs=None,

oob\_score=False, random\_state=0, verbose=0, warm\_start=False)

In [49]: y\_train\_pred **=** regr**.**predict(X\_train)

y\_test\_pred **=** regr**.**predict(X\_test)

regr**.**score(X\_train, y\_train)

Out[49]: 0.4287379777803546

In [50]: regr**.**score(X\_test, y\_test)

Out[50]: 0.43753106247261264