```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

In [3]:

```
df = pd.read_csv('https://drive.google.com/file/d/15iPJ-
fdxEr11Yqg7CKvdLJoecgUqEpXV/view?usp=share_link')
```

In [ ]:

```
df.head()
```

Out[ ]:

	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]:

	utm_source	utm_campaign	docs-cwsd:""	0.8	0.9	0.111	0.112	0.115	0.116	0]	...	2.3	5.2	rw	104	0].7	[[null.2	null.199	null.200	https://www.gstatic.com/og/_/js/k=og.qtm.en_US.aWFQyecCGuU.O/rt=j/m=qabr,q_dnp,qapid/exm=qaaw,qadd,qaid,qein,qhaw,qhbr,qhch,qhga,qhid,qhin,qhpr/d=1/ed=1/rs=AA2YrTsqVU4o3fzN8hEtIA3Fcx7sOx6IpA]]]]	};this.gbar_=this.gbar_||{};(function(_){var window=this;
count	2.000000	2.000000	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

	utm_source	utm_campaign	docs-cwsd:""	0.8	0.9	0.11	0.12	0.15	0.16	0]	...	2.3	5.2	rw	104	0].7	[[null.2	null.199	null.200	https://www.gstatic.com/og/_/js/k=og.qtm.en_US.aWFQyecCGuU.O/rt=j/m=qabr,q_dnp,qapid/exm=qaaw,qadd,qaid,qein,qhaw,qhbr,qhch,qhga,qhid,qhin,qhpr/d=1/ed=1/rs=AA2YrTsqVU4o3fzN8hEtIA3Fcx7sOx6IpA]]]]	};this.gbar_=this.gbar_||{};(function(_){var window=this;
mean	1.500000	1.000000	0.0	255.0	255.0	234.0	237.0	234.0	237.0	0.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
std	2.121320	1.414214	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
min	0.000000	0.000000	0.0	255.0	255.0	234.0	237.0	234.0	237.0	0.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25%	0.750000	0.500000	0.0	255.0	255.0	234.0	237.0	234.0	237.0	0.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
50%	1.500000	1.000000	0.0	255.0	255.0	234.0	237.0	234.0	237.0	0.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
75%	2.250000	1.500000	0.0	255	255	234	237	234	237	0.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

	utm_source	utm_campaign	docs-cwsd:""	0.8	0.9	0.11	0.12	0.15	0.16	0]	...	2.3	5.2	rw	104	0].7	[[null.2	null.199	null.200	https://www.gstatic.com/og/_/js/k=og.qtm.en_US.aWFQyecCGuU.O/rt=j/m=qabr,q_dnp,qapid/exm=qaaw,qadd,qaid,qein,qhaw,qhbr,qhch,qhga,qhid,qhin,qhpr/d=1/ed=1/rs=AA2YrTsqVU4o3fzN8hEtIA3Fcx7sOx6IpA]]]]	};this.gbar_=this.gbar_||{};(function(_){var window=this;
	00	00		.0	.0	.0	.0	.0	.0												
max	3.000000	2.000000	0.0	255.0	255.0	234.0	237.0	234.0	237.0	0.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

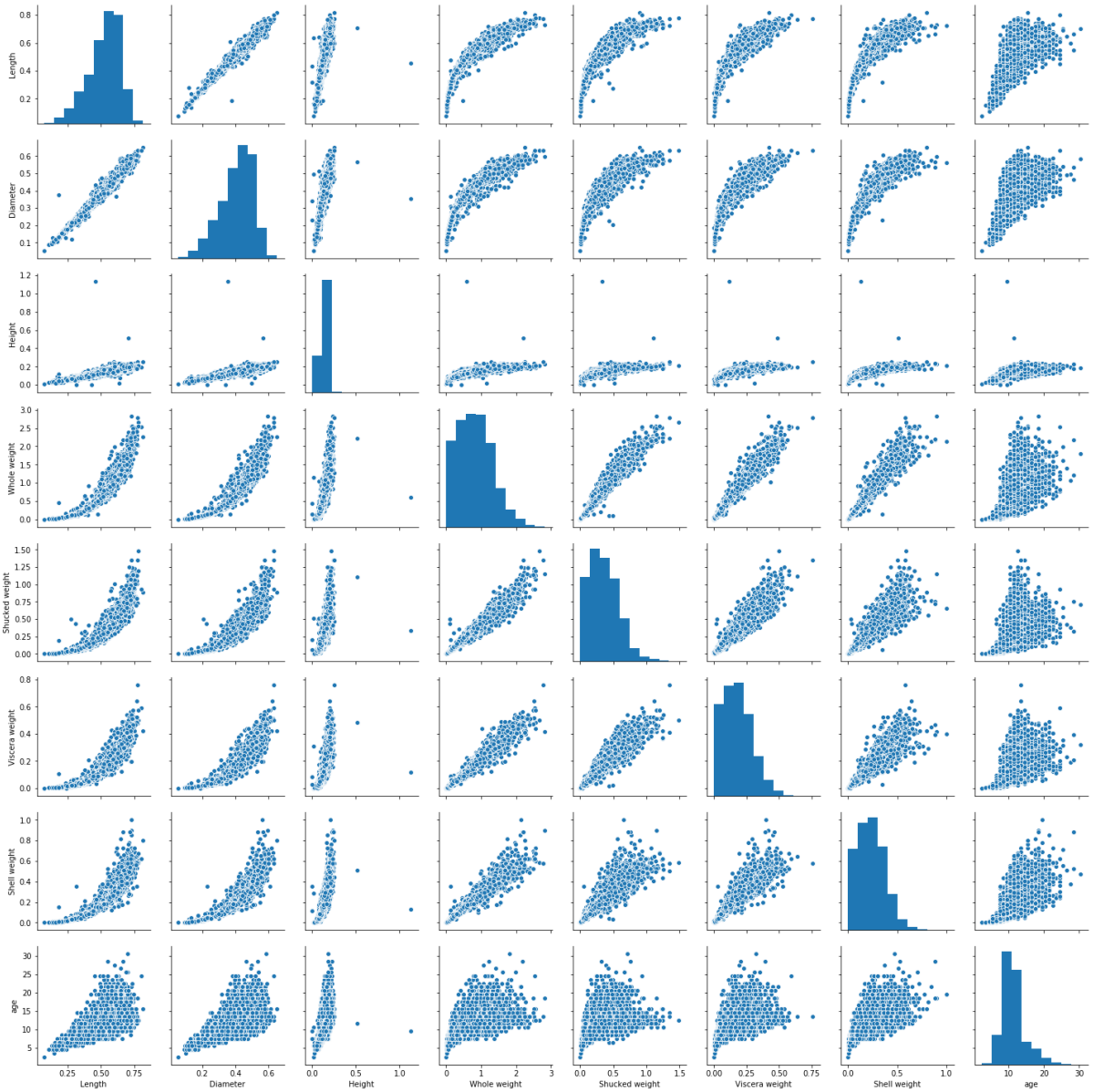
8 rows × 479 columns

EDA

In [ ]:

```
sns.pairplot(df)
```

Out[ ]:



In [ ]:

```
df.info()
```

```
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 [ ]:

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

In [ ]:

```
numerical_features
```

Out[ ]:

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

In [ ]:

```
categorical_features
```

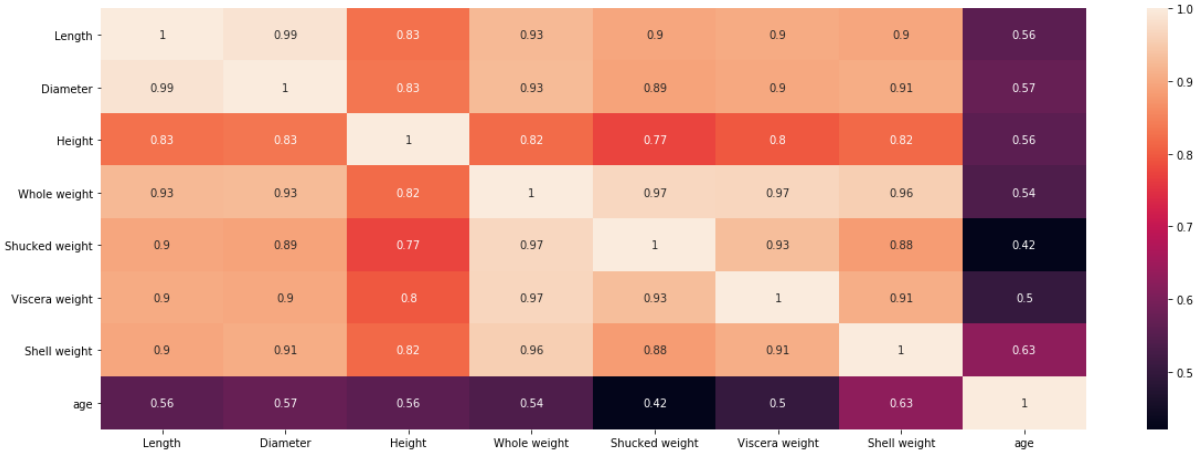
Out[ ]:

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

In [ ]:

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

Out[ ]:

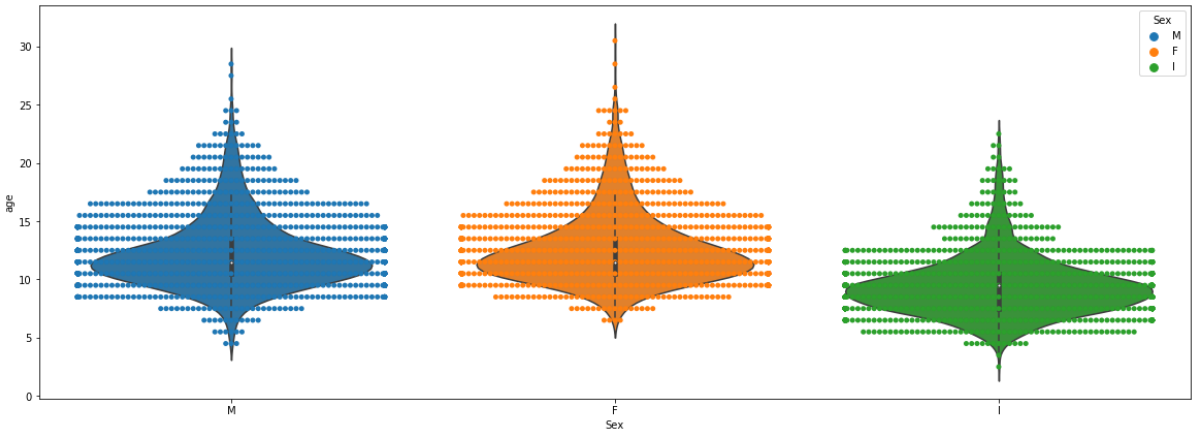


In [ ]:

```
plt.figure(figsize = (20,7))
sns.swarmplot(x = 'Sex', y = 'age', data = df, hue = 'Sex')
sns.violinplot(x = 'Sex', y = 'age',data = df)
```

```
/opt/conda/lib/python3.6/site-packages/scipy/stats/stats.py:1713: FutureWar
ning: Using a non-tuple sequence for multidimensional indexing is deprecate
d; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be
interpreted as an array index, `arr[np.array(seq)]`, which will result eith
er in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[ ]:



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

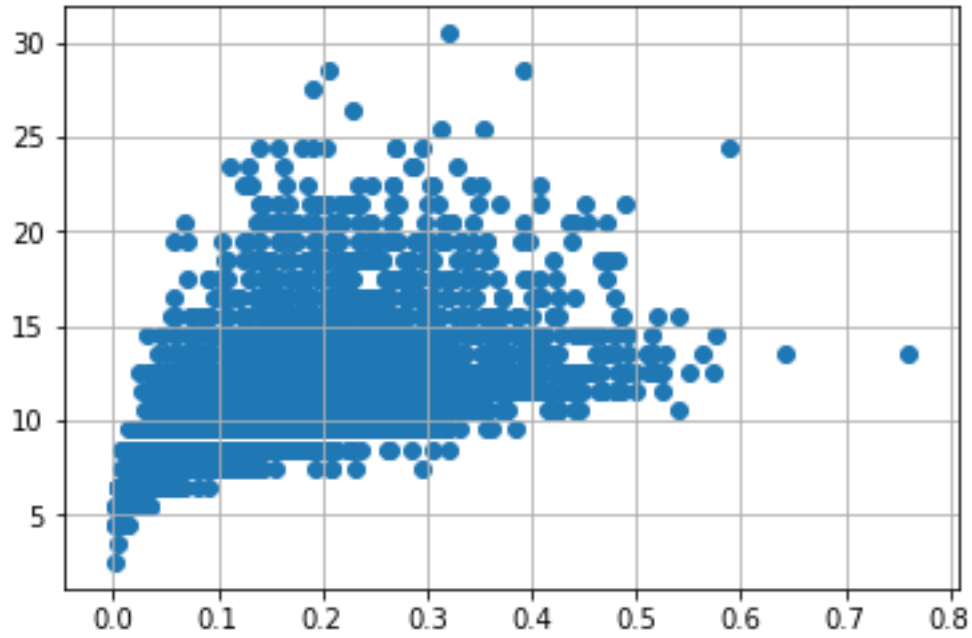
Data Preprocessing

In [ ]:

```
# outlier handling
df = pd.get_dummies(df)
dummy_df = df
```

In [ ]:

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

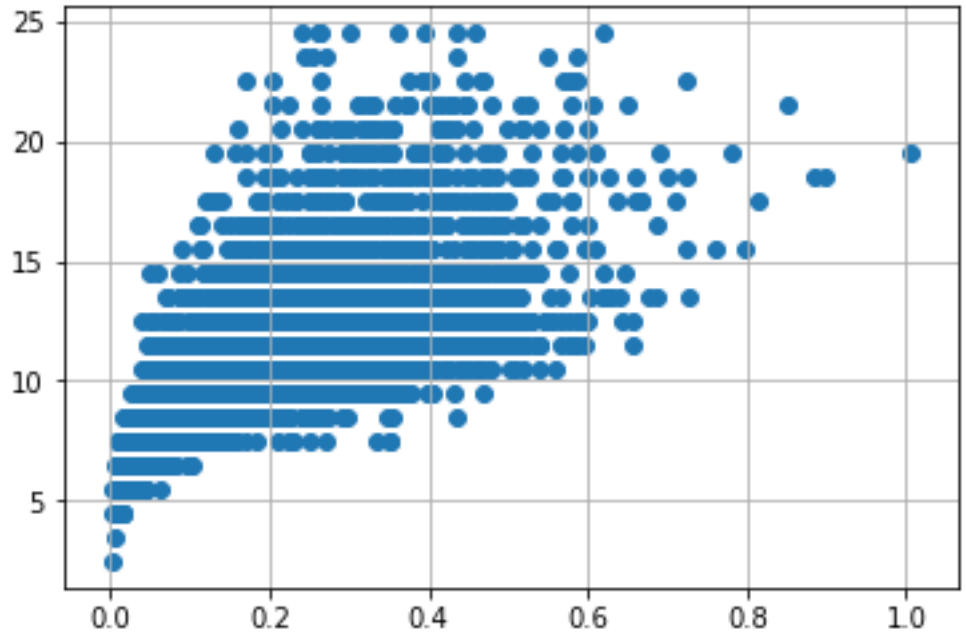


In [ ]:

```
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 [ ]:

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

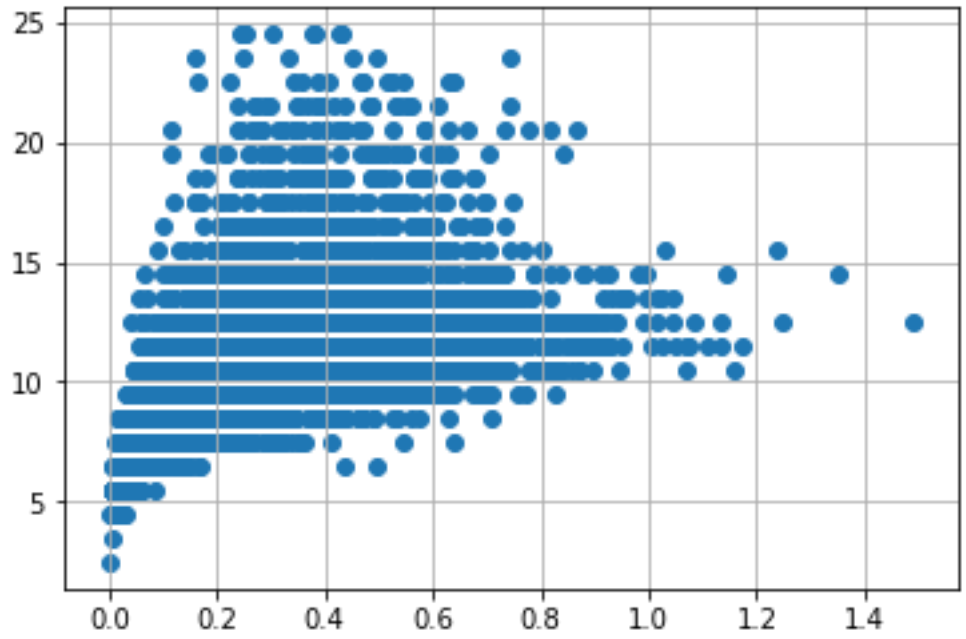


In [ ]:

```
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 [ ]:

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

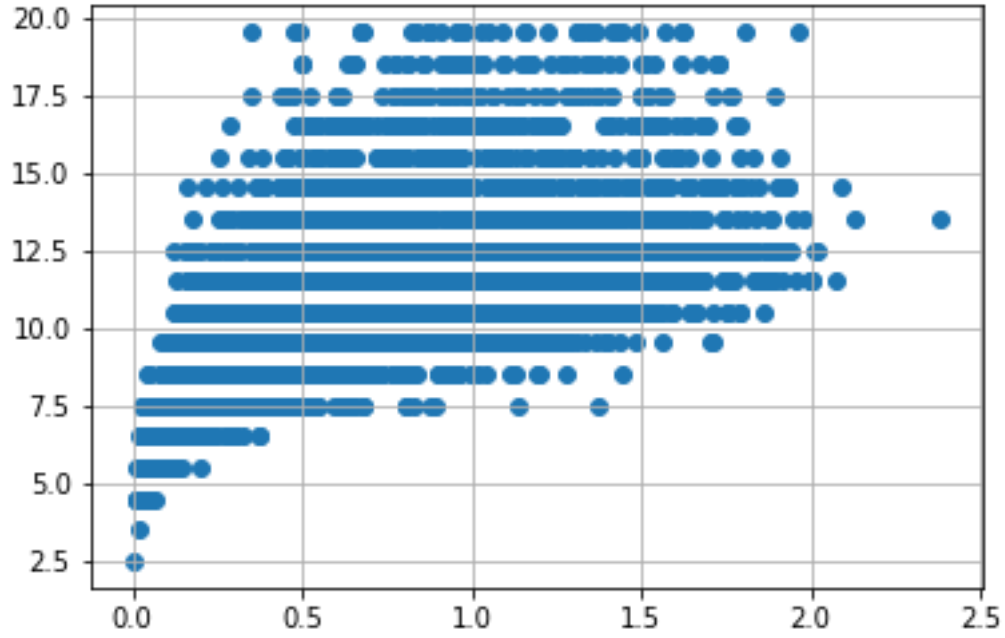


In [ ]:

```
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 [ ]:

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

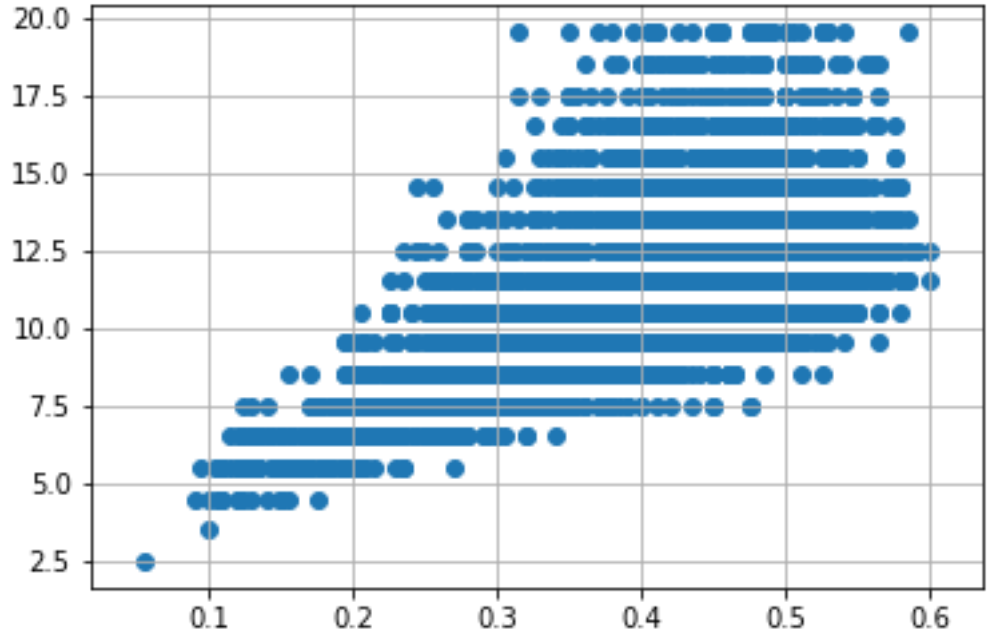


In [ ]:

```
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 [ ]:

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

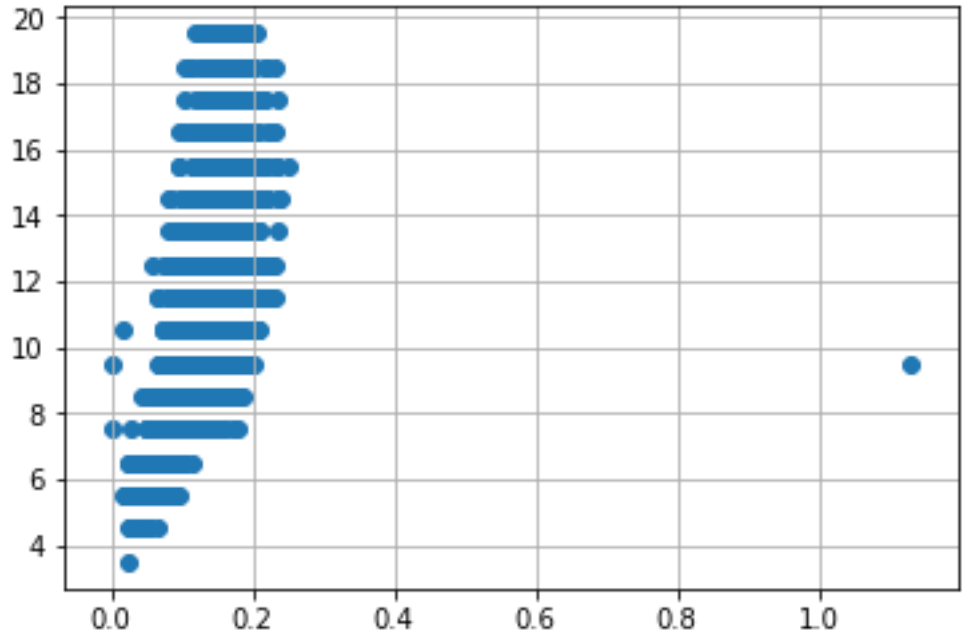


In [ ]:

```
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 [ ]:

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

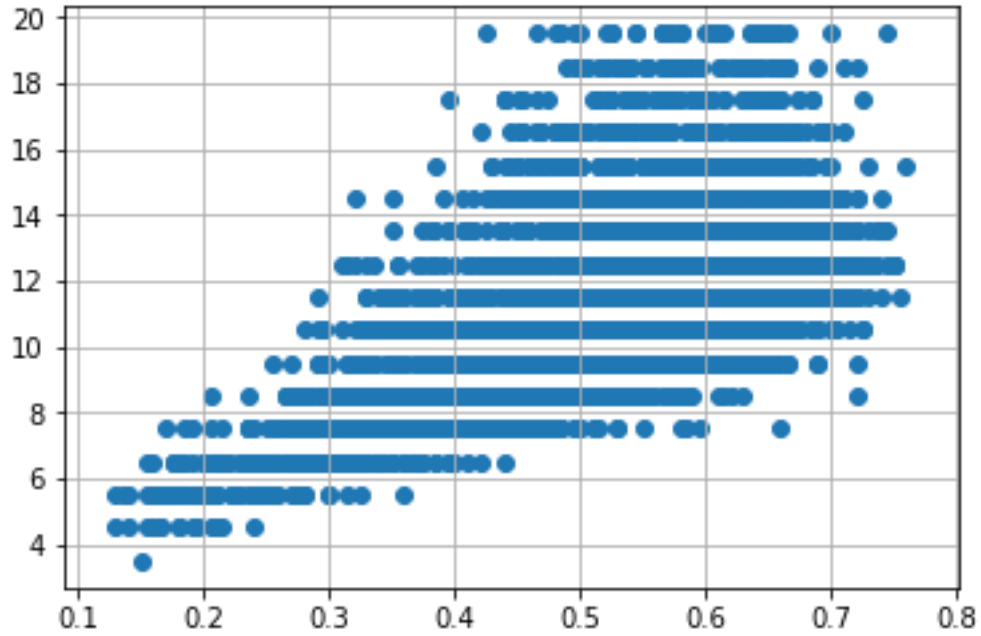


In [ ]:

```
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 [ ]:

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



In [ ]:

```
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 [ ]:

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

In [ ]:

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.feature_selection import SelectKBest
```

In [ ]:

```
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: D
ataConversionWarning: Data with input dtype uint8, float64 were all convert
ed to float64 by StandardScaler.
  return self.partial_fit(X, y)
/opt/conda/lib/python3.6/site-packages/sklearn/base.py:464: DataConversionW
arning: Data with input dtype uint8, float64 were all converted to float64
by StandardScaler.
```

```
  return self.fit(X, **fit_params).transform(X)
```

Linear regression

In [ ]:

```
from sklearn.linear_model import LinearRegression
```

In [ ]:

```
lm = LinearRegression()
lm.fit(X_train, y_train)
```

Out[ ]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [ ]:

```
y_train_pred = lm.predict(X_train)
y_test_pred = lm.predict(X_test)
```

In [ ]:

```
 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 :3.599719
Mean Squared error of testing set :3.466830
```

In [ ]:

```
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)
```

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