

## Assignment 4

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

import os
os.chdir("./Data/")

abalone = pd.read_csv('abalone.csv')
```

abalone

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	\
0	M	0.455	0.365	0.095	0.5140	0.2245	
1	M	0.350	0.265	0.090	0.2255	0.0995	
2	F	0.530	0.420	0.135	0.6770	0.2565	
3	M	0.440	0.365	0.125	0.5160	0.2155	
4	I	0.330	0.255	0.080	0.2050	0.0895	
...	..	...	...	...	...	...	
4172	F	0.565	0.450	0.165	0.8870	0.3700	
4173	M	0.590	0.440	0.135	0.9660	0.4390	
4174	M	0.600	0.475	0.205	1.1760	0.5255	
4175	F	0.625	0.485	0.150	1.0945	0.5310	
4176	M	0.710	0.555	0.195	1.9485	0.9455	

	Viscera weight	Shell weight	Rings
0	0.1010	0.1500	15
1	0.0485	0.0700	7
2	0.1415	0.2100	9
3	0.1140	0.1550	10
4	0.0395	0.0550	7
...	...	...	...
4172	0.2390	0.2490	11
4173	0.2145	0.2605	10
4174	0.2875	0.3080	9
4175	0.2610	0.2960	10
4176	0.3765	0.4950	12

[4177 rows x 9 columns]

abalone.head()

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140

4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395
---	---	-------	-------	-------	--------	--------	--------

	Shell weight	Rings
0	0.150	15
1	0.070	7
2	0.210	9
3	0.155	10
4	0.055	7

abalone.tail()

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	\
4172	F	0.565	0.450	0.165	0.8870	0.3700	
4173	M	0.590	0.440	0.135	0.9660	0.4390	
4174	M	0.600	0.475	0.205	1.1760	0.5255	
4175	F	0.625	0.485	0.150	1.0945	0.5310	
4176	M	0.710	0.555	0.195	1.9485	0.9455	

	Viscera weight	Shell weight	Rings
4172	0.2390	0.2490	11
4173	0.2145	0.2605	10
4174	0.2875	0.3080	9
4175	0.2610	0.2960	10
4176	0.3765	0.4950	12

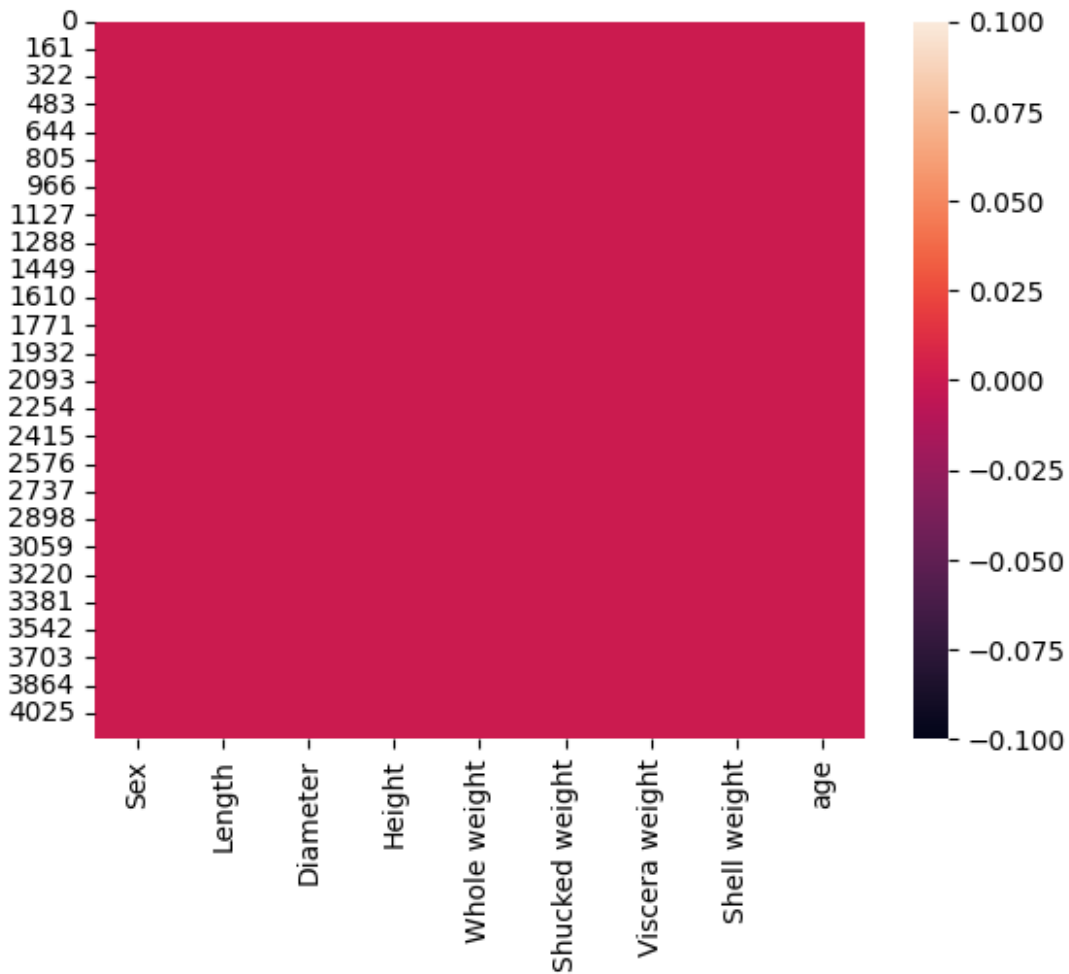
*#age can be calculated by using adding value 1.5 to Rings*

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

## Univariate Analysis

```
sns.heatmap(abalone.isnull())
```

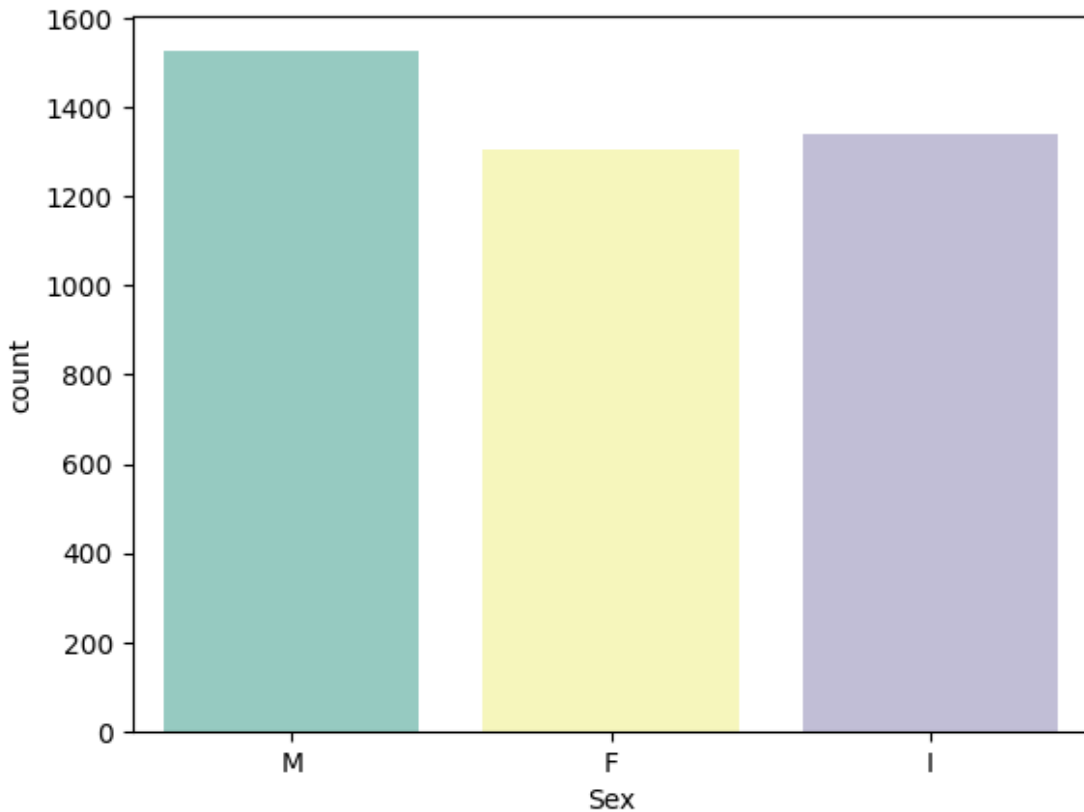
<AxesSubplot: >



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

sns.countplot(x = 'Sex', data = abalone, palette = 'Set3')

<AxesSubplot: xlabel='Sex', ylabel='count'>
```



## Bivariate Analysis

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

```
numerical_features
```

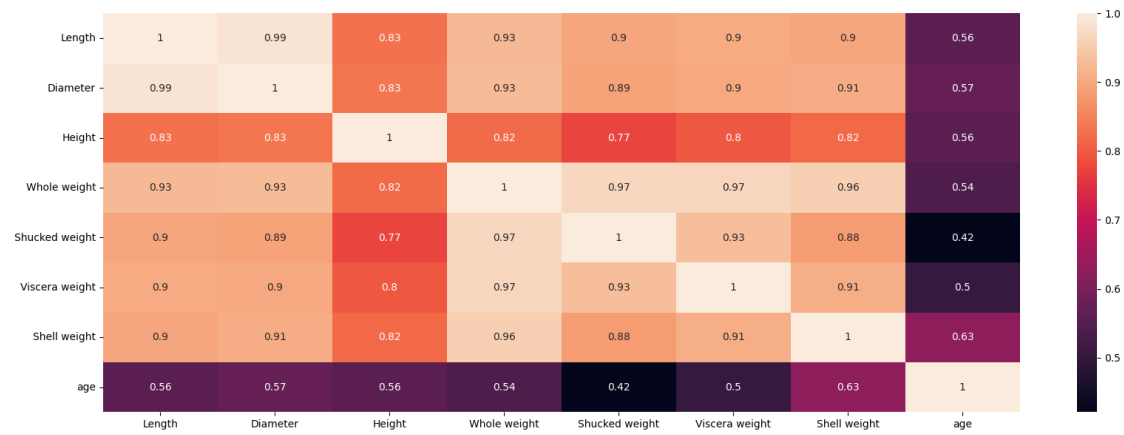
```
Index(['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',  
      'Viscera weight', 'Shell weight', 'age'],  
      dtype='object')
```

```
categorical_features
```

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

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

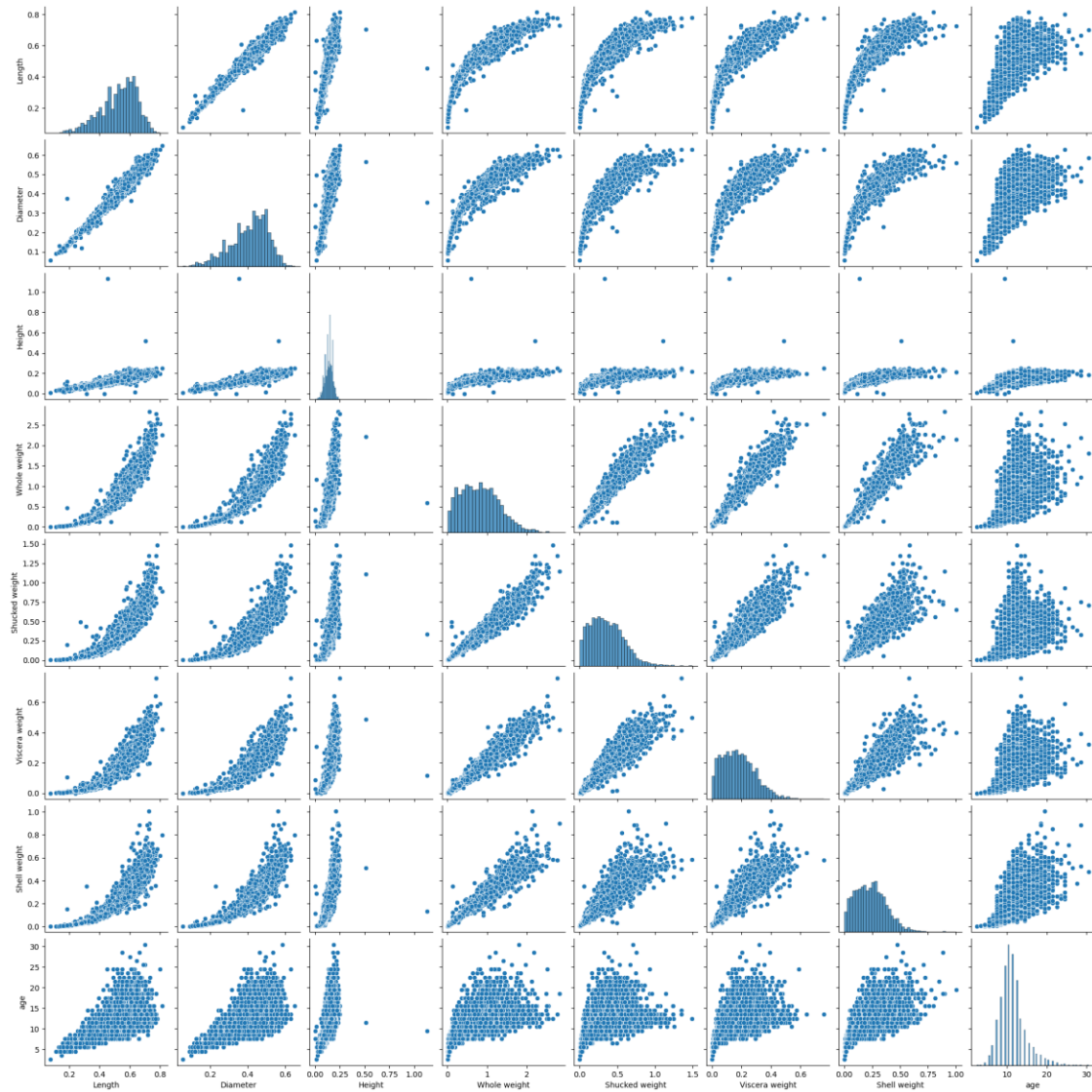
```
<AxesSubplot: >
```



## Multivariate Analysis

```
sns.pairplot(abalone)
```

```
<seaborn.axisgrid.PairGrid at 0x24a6b3f5ae0>
```



## Descriptive Statistics

*#continuous variables*

```
abalone['Length'].describe()
```

```
count    4177.000000
mean      0.523992
std       0.120093
min       0.075000
25%      0.450000
50%      0.545000
75%      0.615000
max       0.815000
Name: Length, dtype: float64
```

```
abalone['Shucked weight'].describe()
```

```
count      4177.000000
mean        0.359367
std         0.221963
min         0.001000
25%         0.186000
50%         0.336000
75%         0.502000
max         1.488000
Name: Shucked weight, dtype: float64
```

```
abalone['Shell weight'].describe()
```

```
count      4177.000000
mean        0.238831
std         0.139203
min         0.001500
25%         0.130000
50%         0.234000
75%         0.329000
max         1.005000
Name: Shell weight, dtype: float64
```

```
abalone['Height'].describe()
```

```
count      4177.000000
mean        0.139516
std         0.041827
min         0.000000
25%         0.115000
50%         0.140000
75%         0.165000
max         1.130000
Name: Height, dtype: float64
```

*# Categorical variable*

```
abalone['Sex'].describe()
```

```
count      4177
unique       3
top         M
freq       1528
Name: Sex, dtype: object
```

```
abalone['Sex'].value_counts()
```

```
M    1528
I    1342
F    1307
Name: Sex, dtype: int64
```

```
#Distribution measures
```

```
abalone['Length'].kurtosis()
```

```
0.06462097389494126
```

```
abalone['Length'].skew()
```

```
-0.639873268981801
```

```
abalone['Shucked weight'].kurtosis()
```

```
0.5951236783694207
```

```
abalone['Shucked weight'].skew()
```

```
0.7190979217612694
```

## Missing values

```
missing_values = abalone.isnull().sum()
```

```
missing_values
```

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

```
missing_values = abalone.isnull().sum().sort_values(ascending = False)
```

```
percentage_missing_values = (missing_values/len(abalone))*100
```

```
pd.concat([missing_values, percentage_missing_values], axis = 1, keys=
['Missing values', '% Missing'])
```

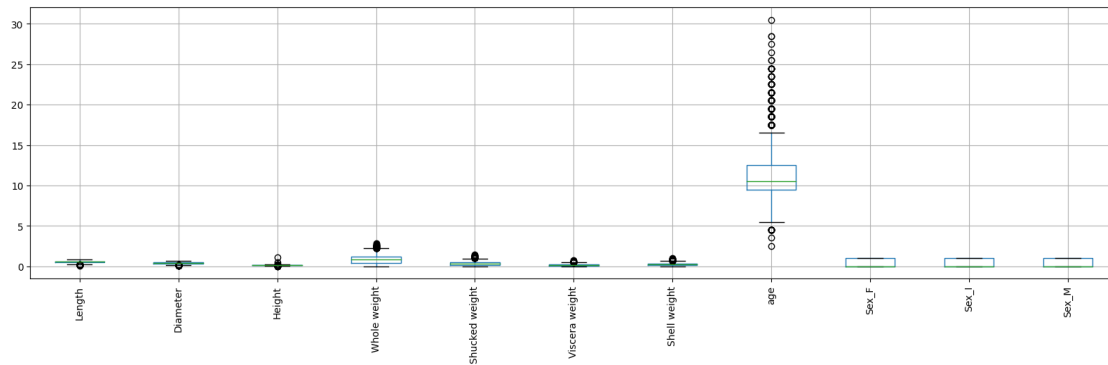
	Missing values	% Missing
Sex	0	0.0
Length	0	0.0
Diameter	0	0.0
Height	0	0.0
Whole weight	0	0.0
Shucked weight	0	0.0
Viscera weight	0	0.0
Shell weight	0	0.0
age	0	0.0

## Outliers

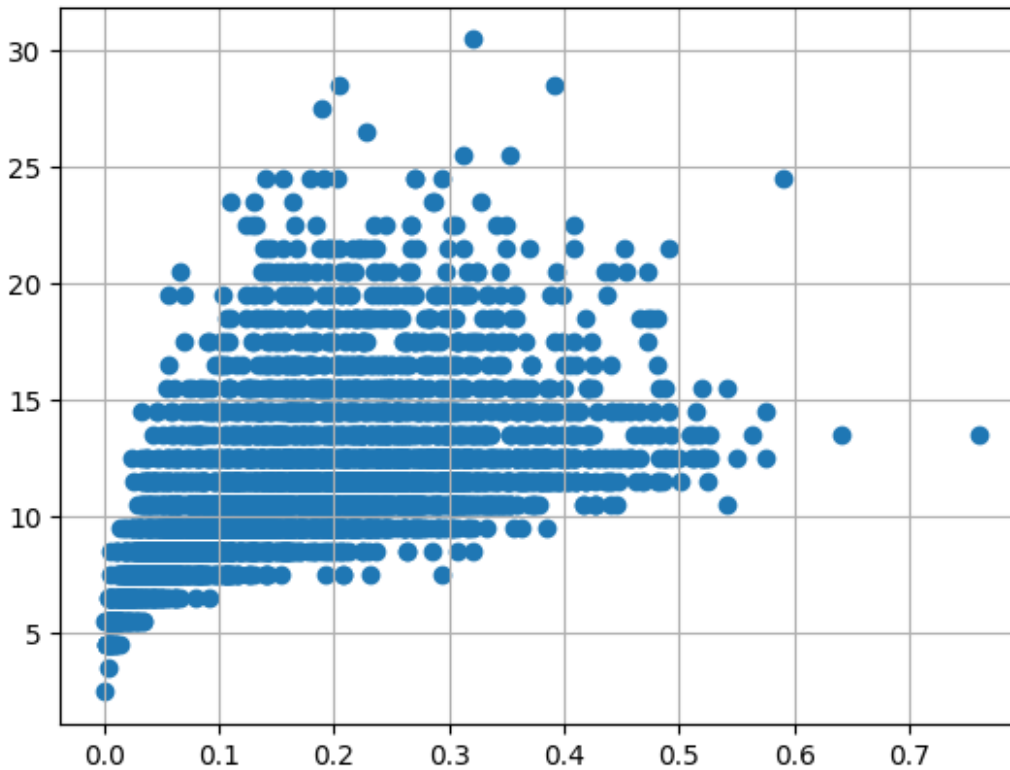
```
abalone = pd.get_dummies(abalone)
dummy_df = abalone
```

```
abalone.boxplot( rot = 90, figsize=(20,5))
```

<AxesSubplot: >



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



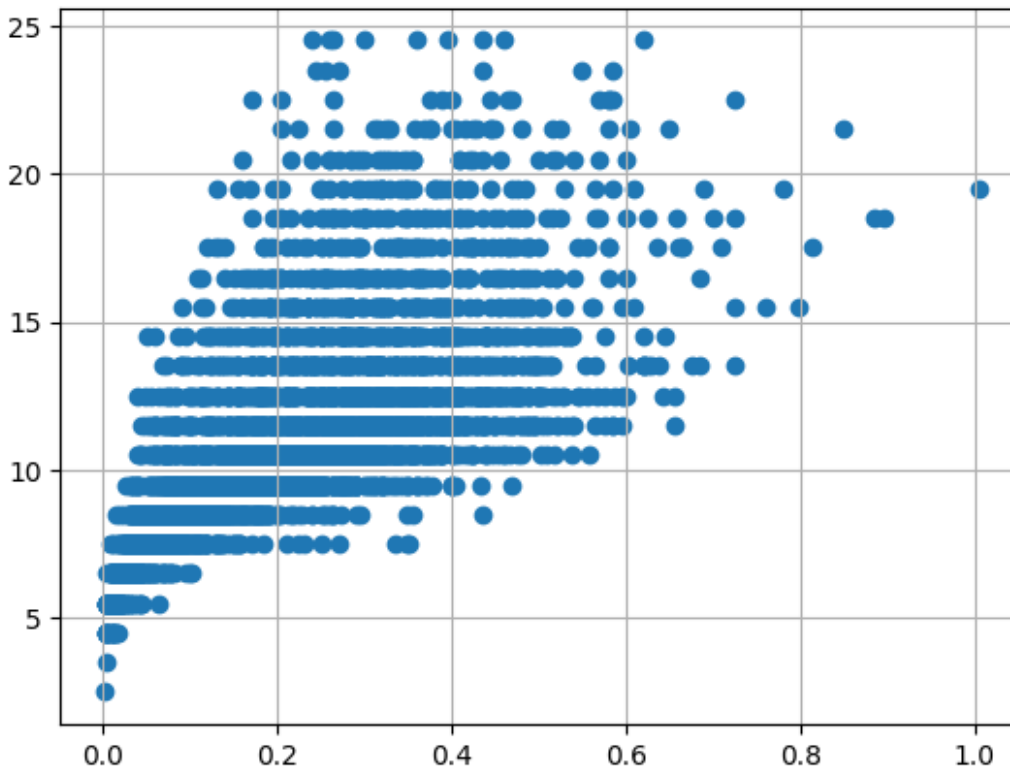
```
abalone.drop(abalone[(abalone['Viscera weight'] > 0.5) & (abalone['age'] < 20)].index, inplace=True)
```

```

abalone.drop(abalone[(abalone['Viscera weight']<0.5) & (abalone['age'] >
25)].index, inplace=True)

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

```

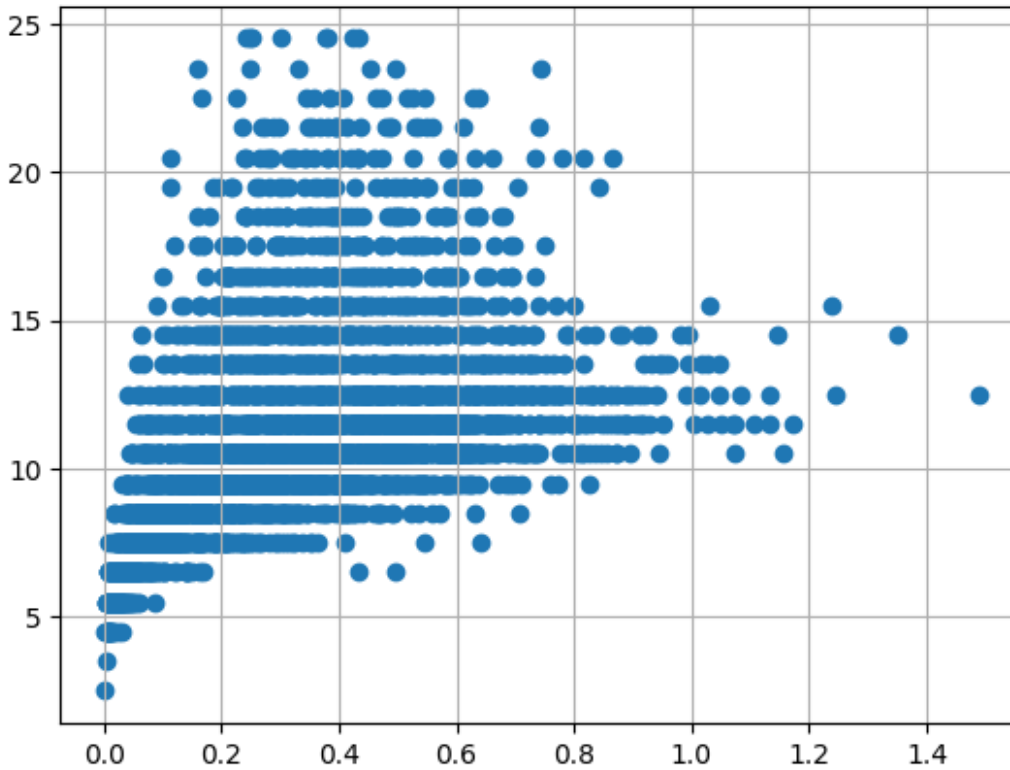


```

abalone.drop(abalone[(abalone['Shell weight']> 0.6) & (abalone['age'] <
25)].index, inplace=True)
abalone.drop(abalone[(abalone['Shell weight']<0.8) & (abalone['age'] >
25)].index, inplace=True)

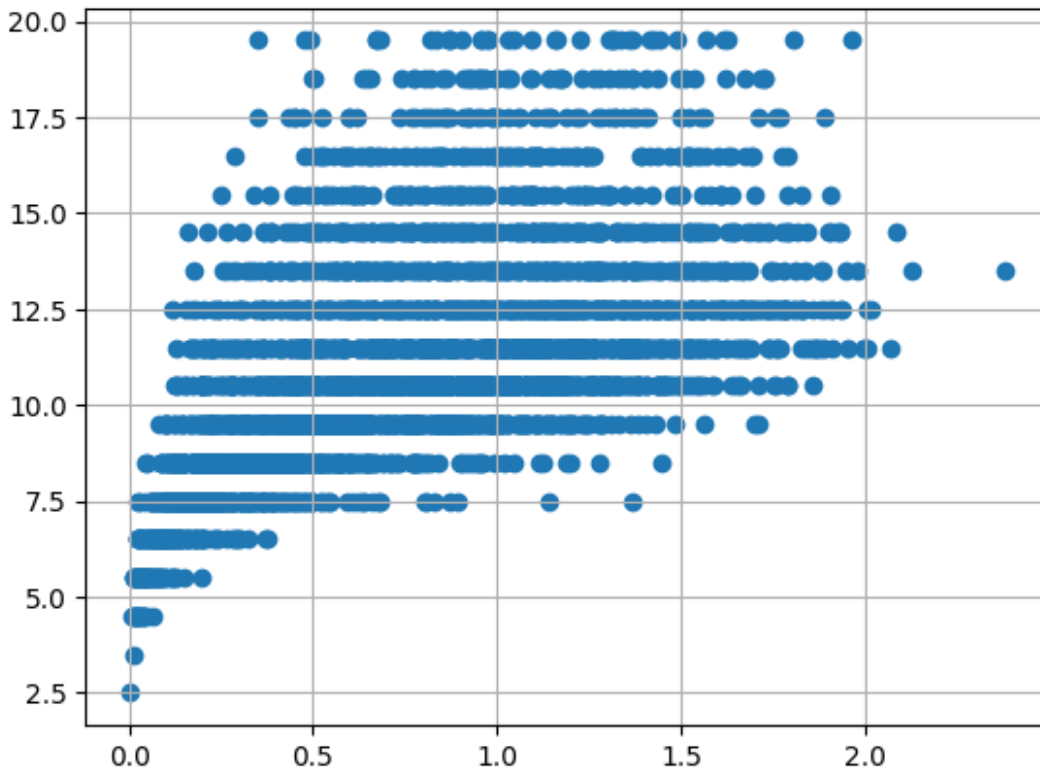
var = 'Shucked weight'
plt.scatter(x = abalone[var], y = abalone['age'])
plt.grid(True)

```



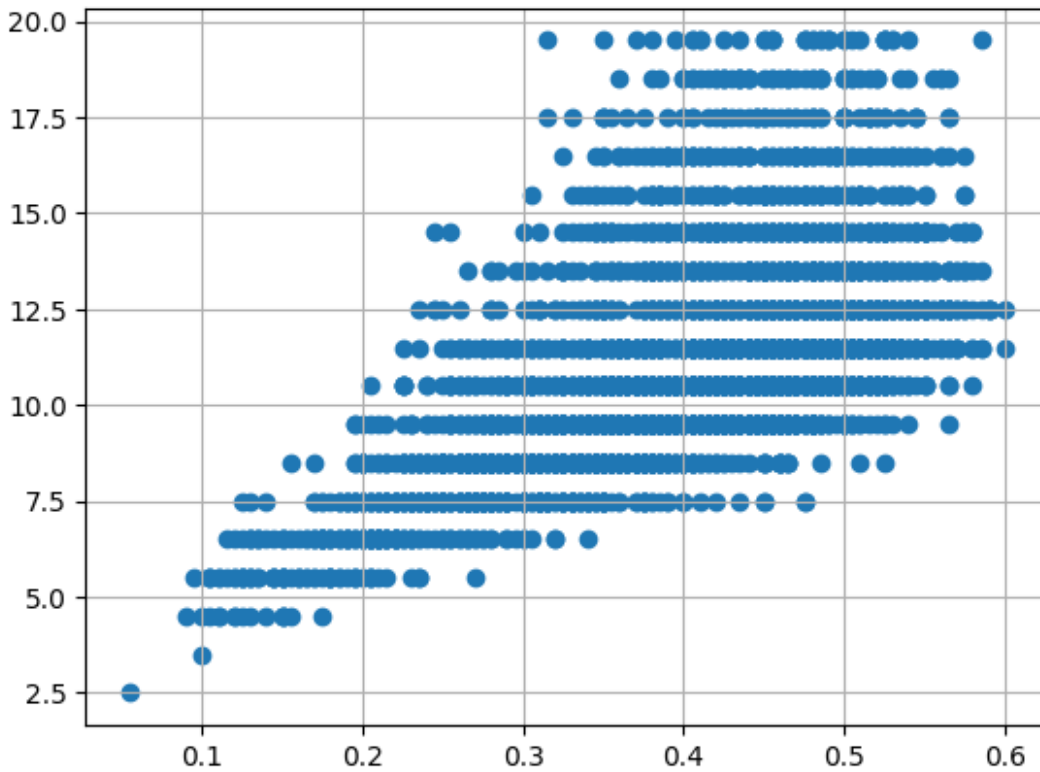
```
abalone.drop(abalone[(abalone['Shucked weight'] >= 1) & (abalone['age'] <
20)].index, inplace = True)
abalone.drop(abalone[(abalone['Viscera weight']<1) & (abalone['age'] >
20)].index, inplace = True)

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



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

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

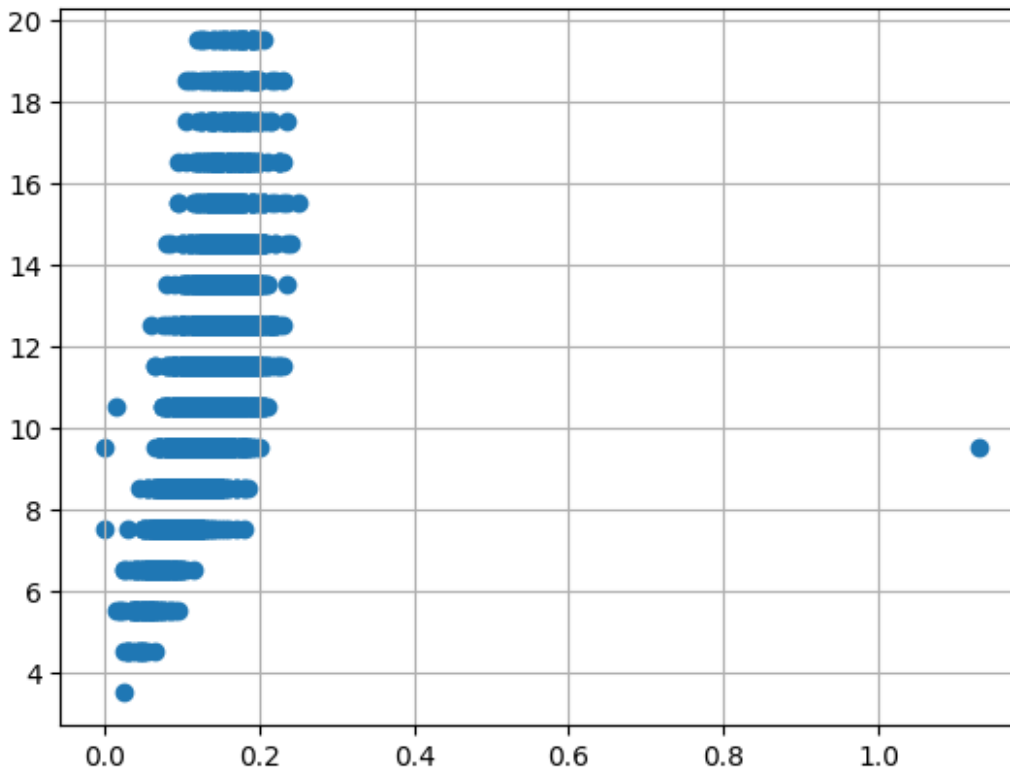


```

abalone.drop(abalone[(abalone['Diameter'] < 0.1) & (abalone['age'] <
5)].index, inplace = True)
abalone.drop(abalone[(abalone['Diameter'] < 0.6) & (abalone['age'] >
25)].index, inplace = True)
abalone.drop(abalone[(abalone['Diameter'] >= 0.6) & (abalone['age'] <
25)].index, inplace = True)

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

```

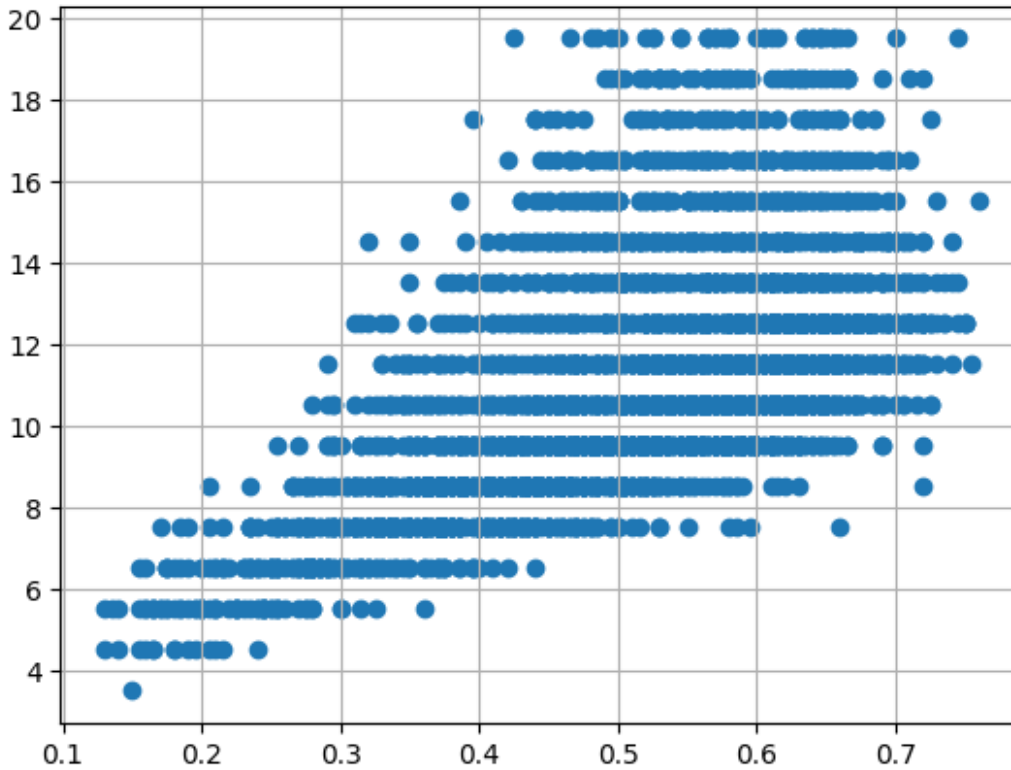


```

abalone.drop(abalone[(abalone['Height'] > 0.4) & (abalone['age'] <
15)].index, inplace = True)
abalone.drop(abalone[(abalone['Height']<0.4) & (abalone['age'] > 25)].index,
inplace = True)

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

```



```

abalone.drop(abalone[(abalone['Length'] < 0.1) & (abalone['age'] < 5)].index,
inplace = True)
abalone.drop(abalone[(abalone['Length'] < 0.8) & (abalone['age'] > 25)].index,
inplace = True)
abalone.drop(abalone[(abalone['Length'] >= 0.8) & (abalone['age'] < 25)].index,
inplace = True)

```

abalone

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight
0	0.455	0.365	0.095	0.5140	0.2245	0.1010
1	0.350	0.265	0.090	0.2255	0.0995	0.0485
2	0.530	0.420	0.135	0.6770	0.2565	0.1415
3	0.440	0.365	0.125	0.5160	0.2155	0.1140
4	0.330	0.255	0.080	0.2050	0.0895	0.0395
...	...	...	...	...	...	...
4172	0.565	0.450	0.165	0.8870	0.3700	0.2390
4173	0.590	0.440	0.135	0.9660	0.4390	0.2145
4174	0.600	0.475	0.205	1.1760	0.5255	0.2875
4175	0.625	0.485	0.150	1.0945	0.5310	0.2610
4176	0.710	0.555	0.195	1.9485	0.9455	0.3765

	Shell weight	age	Sex_F	Sex_I	Sex_M
0	0.1500	16.5	0	0	1
1	0.0700	8.5	0	0	1

2	0.2100	10.5	1	0	0
3	0.1550	11.5	0	0	1
4	0.0550	8.5	0	1	0
...	...	...	...	...	...
4172	0.2490	12.5	1	0	0
4173	0.2605	11.5	0	0	1
4174	0.3080	10.5	0	0	1
4175	0.2960	11.5	1	0	0
4176	0.4950	13.5	0	0	1

[3995 rows x 11 columns]

## Categorical columns

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

```
numerical_features
```

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

```
categorical_features
```

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

```
abalone_numeric = abalone[['Length', 'Diameter', 'Height', 'Whole weight',
'Shucked weight', 'Viscera weight', 'Shell weight', 'age', 'Sex_F', 'Sex_I',
'Sex_M']]
```

```
abalone_numeric.head()
```

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	\
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	

	Shell weight	age	Sex_F	Sex_I	Sex_M
0	0.150	16.5	0	0	1
1	0.070	8.5	0	0	1
2	0.210	10.5	1	0	0
3	0.155	11.5	0	0	1
4	0.055	8.5	0	1	0

## Dependent and Independent Variables

```
x = abalone.iloc[:, 0:1].values
```

```
y = abalone.iloc[:, 1]
```

x

```
array([[0.455],
       [0.35 ],
       [0.53 ],
       ...,
       [0.6   ],
       [0.625],
       [0.71 ]])
```

y

```
0      0.365
1      0.265
2      0.420
3      0.365
4      0.255
```

```
...
4172   0.450
4173   0.440
4174   0.475
4175   0.485
4176   0.555
```

Name: Diameter, Length: 3995, dtype: float64

## Scaling the Independent Variables

```
print ("\n ORIGINAL VALUES: \n\n", x,y)
```

ORIGINAL VALUES:

```
[[0.455]
 [0.35 ]
 [0.53 ]
 ...
 [0.6   ]
 [0.625]
 [0.71 ]] 0      0.365
1      0.265
2      0.420
3      0.365
4      0.255
...
4172   0.450
4173   0.440
4174   0.475
4175   0.485
```

```
4176      0.555
Name: Diameter, Length: 3995, dtype: float64
```

```
from sklearn import preprocessing
min_max_scaler = preprocessing.MinMaxScaler(feature_range =(0, 1))
new_y= min_max_scaler.fit_transform(x,y)
print ("\n VALUES AFTER MIN MAX SCALING: \n\n", new_y)
```

VALUES AFTER MIN MAX SCALING:

```
[[0.51587302]
 [0.34920635]
 [0.63492063]
 ...
 [0.74603175]
 [0.78571429]
 [0.92063492]]
```

## Split the data into Training and Testing

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

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

```
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)
```

## Build the model

### Linear Regression

```
from sklearn import linear_model as lm
from sklearn.linear_model import LinearRegression
model=lm.LinearRegression()
results=model.fit(X_train,y_train)
```

```
accuracy = model.score(X_train, y_train)
print('Accuracy of the model:', accuracy)
```

Accuracy of the model: 0.523907384638246

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

```
LinearRegression()
```

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

## Training the model

X\_train

```
array([[0.61 , 0.46 , 0.145, ..., 1.    , 0.    , 0.    ],
       [0.525, 0.415, 0.15 , ..., 1.    , 0.    , 0.    ],
       [0.45 , 0.33 , 0.105, ..., 0.    , 1.    , 0.    ],
       ...,
       [0.4   , 0.32 , 0.095, ..., 0.    , 0.    , 1.    ],
       [0.37 , 0.275, 0.1   , ..., 0.    , 1.    , 0.    ],
       [0.72 , 0.55 , 0.2   , ..., 1.    , 0.    , 0.    ]])
```

y\_train

```
1923    11.5
2755    11.5
1089     8.5
1710    11.5
1544     7.5
...
3873     6.5
3201     7.5
51       8.5
680      8.5
1200    11.5
```

Name: age, Length: 2996, dtype: float64

```
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)
```

Mean Squared error of training set :3.682301

## Testing the model

X\_test

```
array([[0.445, 0.34 , 0.12 , ..., 0.    , 0.    , 1.    ],
       [0.33 , 0.265, 0.085, ..., 0.    , 1.    , 0.    ],
       [0.62 , 0.525, 0.155, ..., 0.    , 0.    , 1.    ],
       ...,
       ...])
```

```
[0.61 , 0.495, 0.19 , ..., 1.   , 0.   , 0.   ],
[0.615, 0.465, 0.15 , ..., 1.   , 0.   , 0.   ],
[0.6   , 0.465, 0.165, ..., 0.   , 0.   , 1.   ]])
```

y\_test

```
2185    10.5
907      7.5
3683    11.5
380      16.5
3838      9.5
...
823      7.5
2552      7.5
3247    16.5
1165    10.5
2922    12.5
```

Name: age, Length: 999, dtype: float64

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

Mean Squared error of testing set :3.215477

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

R2 Score of training set:0.52

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

R2 Score of testing set:0.56