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

In [3]:
```
import os
os.chdir("C:/Users/cselab/Desktop/DATASETS")
```

In [4]:
```
df = pd.read_csv('abalone.csv')
```

In [5]:
```
df
```

Out[5]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	V
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	

4177 rows × 9 columns

In [6]:
```
df.head()
```

Out[6]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Visce weig
0	M	0.455	0.365	0.095	0.5140	0.2245	0.10
1	M	0.350	0.265	0.090	0.2255	0.0995	0.04
2	F	0.530	0.420	0.135	0.6770	0.2565	0.14
3	M	0.440	0.365	0.125	0.5160	0.2155	0.11
4	I	0.330	0.255	0.080	0.2050	0.0895	0.03

In [7]:
```
df.tail()
```

Out[7]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	V
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	

In [12]:
```
#Abalone age prediction
```

In [48]:
```
df['age'] = df['Rings']+1.5
df = df.drop('Rings', axis = 1)
```

```
---------------------------------------------------------------------------
KeyError                                  Tracebac
k (most recent call last)
File ~\anaconda3\lib\site-packages\pandas\core\ind
exes\base.py:3621, in Index.get_loc(self, key, met
hod, tolerance)
   3620 try:
-> 3621     return self._engine.get_loc(casted_key
)
   3622 except KeyError as err:

File ~\anaconda3\lib\site-packages\pandas\_libs\in
dex.pyx:136, in pandas._libs.index.IndexEngine.get
_loc()

File ~\anaconda3\lib\site-packages\pandas\_libs\in
dex.pyx:163, in pandas._libs.index.IndexEngine.get
_loc()

File pandas\_libs\hashtable_class_helper.pxi:5198,
in pandas._libs.hashtable.PyObjectHashTable.get_it
em()

File pandas\_libs\hashtable_class_helper.pxi:5206,
in pandas._libs.hashtable.PyObjectHashTable.get_it
em()

KeyError: 'Rings'

The above exception was the direct cause of the fo
llowing exception:

KeyError                                  Tracebac
```

```
k (most recent call last)
Input In [48], in ()
----> 1 df['age'] = df['Rings']+1.5
      2 df = df.drop('Rings', axis = 1)

File ~\anaconda3\lib\site-packages\pandas\core\fra
me.py:3505, in DataFrame.__getitem__(self, key)
   3503 if self.columns.nlevels > 1:
   3504     return self._getitem_multilevel(key)
-> 3505 indexer = self.columns.get_loc(key)
   3506 if is_integer(indexer):
   3507     indexer = [indexer]

File ~\anaconda3\lib\site-packages\pandas\core\ind
exes\base.py:3623, in Index.get_loc(self, key, met
hod, tolerance)
   3621     return self._engine.get_loc(casted_ke
y)
   3622 except KeyError as err:
-> 3623     raise KeyError(key) from err
   3624 except TypeError:
   3625     # If we have a listlike key, _check_in
dexing_error will raise
   3626     #  InvalidIndexError. Otherwise we fal
l through and re-raise
   3627     #  the TypeError.
   3628     self._check_indexing_error(key)

KeyError: 'Rings'
```

In [49]:

```
df
```

Out[49]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	V
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	

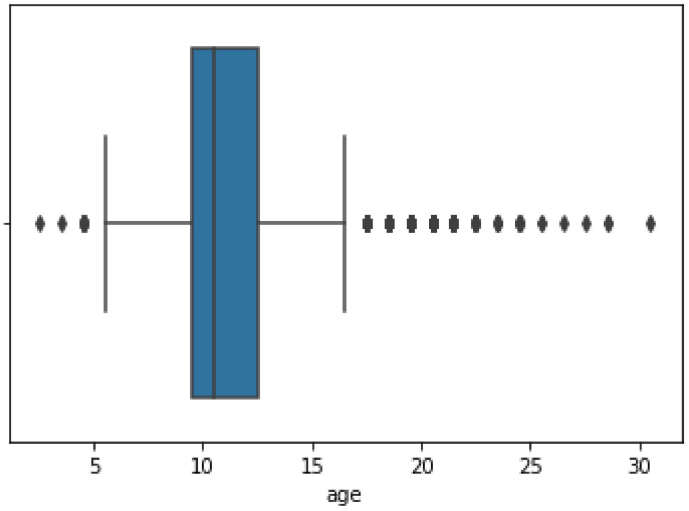
4177 rows × 9 columns

In [19]:
```
#Perform visualisations
#Univariate analysis
```

In [37]:
```
sns.boxplot(df.age)
```

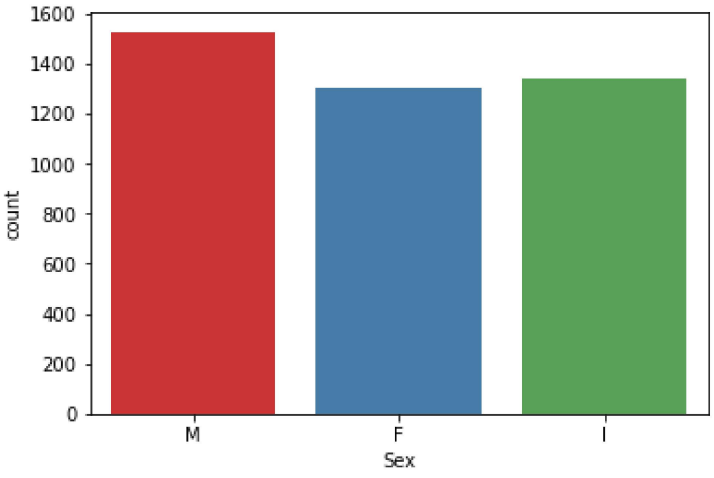
```
C:\Users\cselab\anaconda3\lib\site-packages\seabor
n\_decorators.py:36: FutureWarning: Pass the follo
wing variable as a keyword arg: x. From version 0.
12, the only valid positional argument will be `da
ta`, and passing other arguments without an explic
it keyword will result in an error or misinterpret
ation.
  warnings.warn(
```

Out[37]:



In [38]:
```
sns.countplot(x = 'Sex', data = df, palette = 'Se
```

Out[38]:



In [51]:
```
sns.heatmap(df , linewidth= 1, cmap = 'coolwarm')
```

```
---------------------------------------------
-------------------------
ValueError                                Tracebac
```

```
k (most recent call last)
Input In [51], in ()
----> 1 sns.heatmap(df , linewidth=1, cmap = 'cool
warm')

File ~\anaconda3\lib\site-packages\seaborn\_decora
tors.py:46, in _deprecate_positional_args..inner_f
(*args, **kwargs)
     36     warnings.warn(
     37         "Pass the following variable{} as
{}keyword arg{}: {}. "
     38         "From version 0.12, the only valid
positional argument "
   (...)
     43         FutureWarning
     44     )
     45 kwargs.update({k: arg for k, arg in zip(si
g.parameters, args)})
---> 46 return f(**kwargs)

File ~\anaconda3\lib\site-packages\seaborn\matrix.
py:540, in heatmap(data, vmin, vmax, cmap, center,
robust, annot, fmt, annot_kws, linewidths, linecol
or, cbar, cbar_kws, cbar_ax, square, xticklabels,
 yticklabels, mask, ax, **kwargs)
    362 """Plot rectangular data as a color-encode
d matrix.
    363
    364 This is an Axes-level function and will dr
aw the heatmap into the
   (...)
    537     ...     ax = sns.heatmap(corr, mask=ma
sk, vmax=.3, square=True)
    538 """
    539 # Initialize the plotter object
--> 540 plotter =  HeatMapper(data, vmin, vmax, cm
ap, center, robust, annot, fmt,
    541                       annot_kws, cbar, cba
r_kws, xticklabels,
    542                       yticklabels, mask)
    544 # Add the pcolormesh kwargs here
    545 kwargs["linewidths"] = linewidths

File ~\anaconda3\lib\site-packages\seaborn\matrix.
py:159, in _HeatMapper.__init__(self, data, vmin,
 vmax, cmap, center, robust, annot, fmt, annot_kw
s, cbar, cbar_kws, xticklabels, yticklabels, mask)
    156 self.ylabel = ylabel if ylabel is not None
else ""
    158 # Determine good default values for the co
lormapping
--> 159 self._determine_cmap_params(plot_data, vmi
n, vmax,
    160                             cmap, center,
robust)
    162 # Sort out the annotations
    163 if annot is None or annot is False:

File ~\anaconda3\lib\site-packages\seaborn\matrix.
py:193, in _HeatMapper._determine_cmap_params(sel
```

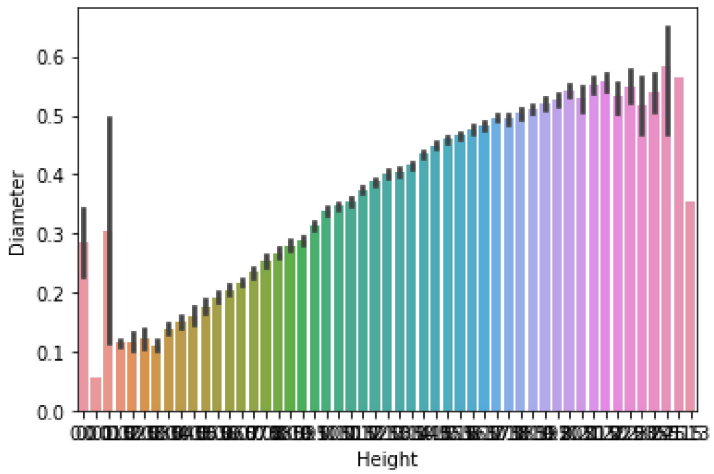
```
f, plot_data, vmin, vmax, cmap, center, robust)
    190 """Use some heuristics to set good default
s for colorbar and range."""
    192 # plot_data is a np.ma.array instance
--> 193 calc_data = plot_data.astype(float).filled
(np.nan)
    194 if vmin is None:
    195     if robust:

ValueError: could not convert string to float: 'M'
```

In [50]:
```
#Bivariate analysis
```

In [52]:
```
sns.barplot(x=df.Height,y=df.Diameter)
```

Out[52]:



In [53]:
```
numerical_features = df.select_dtypes(include = [
categorical_features = df.select_dtypes(include =
```

```
C:\Users\cselab\AppData\Local\Temp\ipykernel_3408
\1755490424.py:2: DeprecationWarning: `np.object`
is a deprecated alias for the builtin `object`. To
silence this warning, use `object` by itself. Doin
g this will not modify any behavior and is safe.
Deprecated in NumPy 1.20; for more details and gui
dance: https://numpy.org/devdocs/release/1.20.0-no
tes.html#deprecations
  categorical_features = df.select_dtypes(include
= [np.object]).columns
```

In [55]:
```
numerical_features
```

Out[55]:
```
Index(['Length', 'Diameter', 'Height', 'Whole weig
ht', 'Shucked weight',
       'Viscera weight', 'Shell weight', 'age'],
      dtype='object')
```

In [56]:
```
categorical_features
```

Out[56]:
```
Index(['Sex'], dtype='object')
```

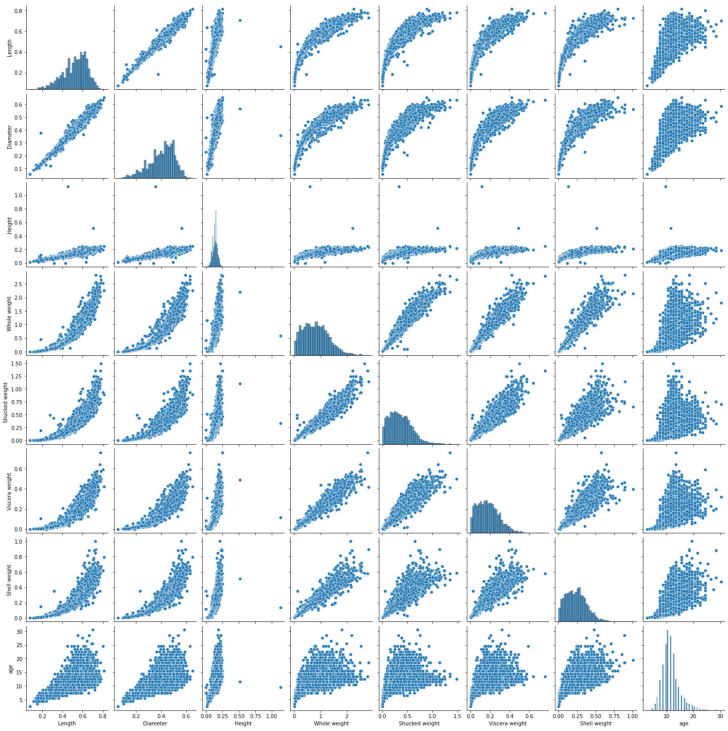
In [57]:
```
plt.figure(figsize = (20,7))
sns.heatmap(df[numerical_features].corr(),annot =
```

Out[57]:

In [58]:
```
#Multivariate Analysis
```

In [59]:
```
sns.pairplot(df)
```

Out[59]:



In [60]:
```
#Perform descriptive model on the dataset
```

In [61]:
```
df['Height'].describe()
```

Out[61]:
```
count    4177.000000
mean        0.139516
std         0.041827
min         0.000000
25%         0.115000
```

```
25%         0.115000
50%         0.140000
75%         0.165000
max         1.130000
Name: Height, dtype: float64
```

In [62]:
```
df['Height'].mean()
```

Out[62]:
```
0.1395163993296614
```

In [63]:
```
df.max()
```

Out[63]:
```
Sex                   M
Length            0.815
Diameter           0.65
Height             1.13
Whole weight     2.8255
Shucked weight    1.488
Viscera weight     0.76
Shell weight      1.005
age                30.5
dtype: object
```

In [64]:
```
df['Sex'].value_counts()
```

Out[64]:
```
M    1528
I    1342
F    1307
Name: Sex, dtype: int64
```

In [65]:
```
df[df.Height == 0]
```

Out[65]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	V
1257	I	0.430	0.34	0.0	0.428	0.2065	
3996	I	0.315	0.23	0.0	0.134	0.0575	

In [66]:
```
df['Shucked weight'].kurtosis()
```

Out[66]:
```
0.5951236783694207
```

In [67]:
```
df['Diameter'].median()
```

Out[67]:
```
0.425
```

In [68]:
```
df['Shucked weight'].skew()
```

Out[68]:
```
0.7190979217612694
```

In [69]:
```
#Missing values
```

In [70]:
```
df.isnull().sum()
```

Out[70]:
```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
age               0
dtype: int64
```

In [71]:
```
df.isna().sum()
```

Out[71]:
```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
age               0
dtype: int64
```

In [72]:
```
df.isna().any()
```

Out[72]:
```
Sex               False
Length            False
Diameter          False
Height            False
Whole weight      False
Shucked weight    False
Viscera weight    False
Shell weight      False
age               False
dtype: bool
```

In [73]:
```
#Missing values
```

In [74]:
```
missing_values = df.isnull().sum().sort_values(as
percentage_missing_values = (missing_values/len(d
pd.concat([missing_values, percentage_missing_val
```

Out[74]:

	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

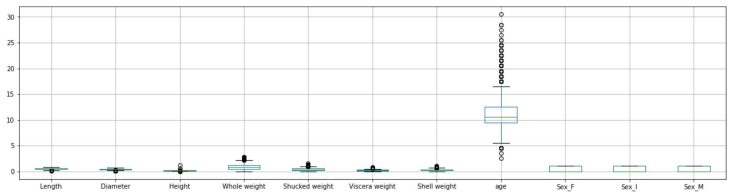
In [75]:
```
len(df)
```

Out[75]: 4177

In [76]:
```
#Find the outliers
```

In [85]:
```
df = pd.get_dummies(df)
dummy_df = df
df.boxplot( rot = 0, figsize=(20,5))
```

Out[85]:



In [86]:
```
df.drop(df[(df['Viscera weight']> 0.5) & (df['age
df.drop(df[(df['Viscera weight']<0.5) & (df['age'
```

In [87]:
```
df
```

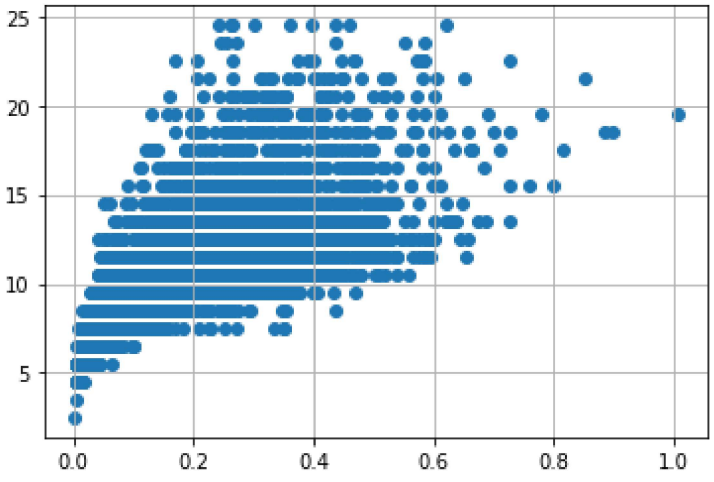
Out[87]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weigh
0	0.455	0.365	0.095	0.5140	0.2245	0.101(
1	0.350	0.265	0.090	0.2255	0.0995	0.048!
2	0.530	0.420	0.135	0.6770	0.2565	0.141!
3	0.440	0.365	0.125	0.5160	0.2155	0.114(
4	0.330	0.255	0.080	0.2050	0.0895	0.039!
...	...	...	...	...	...	..
4172	0.565	0.450	0.165	0.8870	0.3700	0.239(
4173	0.590	0.440	0.135	0.9660	0.4390	0.214!
4174	0.600	0.475	0.205	1.1760	0.5255	0.287!

4175	0.625	0.485	0.150	1.0945	0.5310	0.261(
4176	0.710	0.555	0.195	1.9485	0.9455	0.376[

4147 rows × 11 columns

In [88]:
```
var = 'Shell weight'
plt.scatter(x = df[var], y = df['age'])
plt.grid(True)
```



In [89]:
```
#Dependent and Independent Variables
```

In [101…
```
x = df.iloc[:, 0:1].values
```

In [102…
```
x
```

Out[102…
```
array([[0.455],
       [0.35 ],
       [0.53 ],
       ...,
       [0.6  ],
       [0.625],
       [0.71 ]])
```

In [92]:
```
y = df.iloc[:, 1]
```

In [93]:
```
y
```

Out[93]:
```
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: 4147, dtype: float64
```

In [98]:

```
#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: 4147, dtype: float64
```

In [103…

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

```
VALUES AFTER MIN MAX SCALING:

[[0.51351351]
[0.37162162]
[0.61486486]
...
[0.70945946]
[0.74324324]
[0.85810811]]
```

In [108…

```
#Split the data into Training and Testing
X = df.drop('age', axis = 1)
y = df['age']
```

In [109…

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_sp
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_spl
X_train
```

Out[109…
```
array([[0.72 , 0.59 , 0.205, ..., 1.   , 0.   , 0.
],
       [0.6  , 0.475, 0.16 , ..., 0.   , 0.   , 1.
],
       [0.565, 0.445, 0.155, ..., 1.   , 0.   , 0.
],
       ...,
       [0.535, 0.42 , 0.13 , ..., 1.   , 0.   , 0.
],
       [0.385, 0.275, 0.115, ..., 0.   , 0.   , 1.
],
       [0.3  , 0.24 , 0.09 , ..., 0.   , 0.   , 1.
]])
```

In [110…
```
y_train
```

Out[110…
```
4106    12.5
3501    13.5
140     11.5
803      8.5
233      8.5
        ...
2291    11.5
1836     7.5
2065     9.5
772      9.5
3249     7.5
Name: age, Length: 3110, dtype: float64
```

In [111…
```
# Build the model
# Linear Regression
```

In [112…
```
from sklearn import linear_model as lm
from sklearn.linear_model import LinearRegression
model=lm.LinearRegression()
results=model.fit(X_train,y_train)
```

In [113…
```
accuracy = model.score(X_train, y_train)
print('Accuracy of the model:', accuracy)
```

```
Accuracy of the model: 0.5345240899781238
```

In [114…
```
#Training the model
lm = LinearRegression()
lm.fit(X_train, y_train)
y_train_pred = lm.predict(X_train)
y_train_pred
```

Out[114…
```
array([13.93218248, 12.77125849, 12.37911511, ...,
11.42060958
```

```
       9.92718831,  8.91330011])
```

In [115…
```
X_train
```

Out[115…
```
array([[0.72 , 0.59 , 0.205, ..., 1.   , 0.   , 0.
],
       [0.6  , 0.475, 0.16 , ..., 0.   , 0.   , 1.
],
       [0.565, 0.445, 0.155, ..., 1.   , 0.   , 0.
],
       ...,
       [0.535, 0.42 , 0.13 , ..., 1.   , 0.   , 0.
],
       [0.385, 0.275, 0.115, ..., 0.   , 0.   , 1.
],
       [0.3  , 0.24 , 0.09 , ..., 0.   , 0.   , 1.
]])
```

In [116…
```
y_train
```

Out[116…
```
4106    12.5
3501    13.5
140     11.5
803      8.5
233      8.5
        ...
2291    11.5
1836     7.5
2065     9.5
772      9.5
3249     7.5
Name: age, Length: 3110, dtype: float64
```

In [117…
```
from sklearn.metrics import mean_absolute_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 :4.559881
```

In [118…
```
#Testing the model
```

In [119…
```
y_train_pred = lm.predict(X_train)
y_test_pred = lm.predict(X_test)
```

In [120…
```
y_test_pred
```

Out[120…
```
array([14.09032428, 11.30893194,  8.21840963, ...,
8.35582227,
       13.91067006,  8.51141958])
```

In [121…
```
X_test
```

Out[121…
```
array([[0.57 , 0.48 , 0.175, ..., 0.   , 0.   , 1.
],
       [0.46 , 0.355, 0.14 , ..., 0.   , 0.   , 1.
],
       [0.375, 0.275, 0.085, ..., 0.   , 1.   , 0.
],
       ...,
       [0.355, 0.27 , 0.075, ..., 0.   , 1.   , 0.
],
       [0.57 , 0.45 , 0.165, ..., 1.   , 0.   , 0.
],
       [0.35 , 0.26 , 0.095, ..., 0.   , 1.   , 0.
]])
```

In [122…
```
y_test
```

Out[122…
```
73      12.5
3090    11.5
2004     8.5
399     12.5
1674    10.5
        ...
3202    18.5
474     11.5
1538     7.5
2234    15.5
133      8.5
Name: age, Length: 1037, dtype: float64
```

In [123…
```
p = mean_squared_error(y_test, y_test_pred)
print('Mean Squared error of testing set :%2f'%p)
```

```
Mean Squared error of testing set :5.077154
```

In [124…
```
#Measure the performance using metrices
```

In [125…
```
from sklearn.metrics import r2_score
```