Data Description

Predicting the age of abalone from physical measurements. The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope -- a boring and time-consuming task. Other measurements, which are easier to obtain, are used to predict the age. Further information, such as weather patterns and location (hence food availability) may be required to solve the problem.

From the original data examples with missing values were removed (the majority having the predicted value missing), and the ranges of the continuous values have been scaled for use with an ANN (by dividing by 200).

Attribute Information:

Given is the attribute name, attribute type, the measurement unit and a brief description. The number of rings is the value to predict: either as a continuous value or as a classification problem.

Name / Data Type / Measurement Unit / Description

Sex / nominal / -- / M, F, and I (infant)

Length / continuous / mm / Longest shell measurement

Diameter / continuous / mm / perpendicular to length

Height / continuous / mm / with meat in shell

Whole weight / continuous / grams / whole abalone

Shucked weight / continuous / grams / weight of meat

Viscera weight / continuous / grams / gut weight (after bleeding)

Shell weight / continuous / grams / after being dried

Rings / integer / -- / +1.5 gives the age in years

1. Download the dataset.

2. Load the dataset into the tool.

importing libraries

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

```
df = pd.read_csv( "/content/drive/MyDrive/Untitled folder/abalone (1).csv")
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import os
os.listdir()
```

```
['.config', 'drive', 'sample_data']
```

```
import pandas as pd
path = "/content/drive/MyDrive/Untitled folder/abalone (1).csv"
df = pd.read_csv(path)
df.describe
```

```
<bound method NDFrame.describe of      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]>
```

3.Perform Below visualizations:

*Univariate Analysis

```
df
```

	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	0.1500	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7
...	...	...	...	...	...	...	...	...	...
4172	F	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11
4173	M	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	10
4174	M	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9
4175	F	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10
4176	M	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12

4177 rows × 9 columns

```
df.head()
```

	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

```
df.describe()
```

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.(
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0.:
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.(
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.:
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.:

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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Sex             4177 non-null   object 
 1   Length          4177 non-null   float64
 2   Diameter        4177 non-null   float64
 3   Height          4177 non-null   float64
 4   Whole weight    4177 non-null   float64
 5   Shucked weight  4177 non-null   float64
 6   Viscera weight  4177 non-null   float64
 7   Shell weight    4177 non-null   float64
 8   age             4177 non-null   float64
dtypes: float64(8), object(1)
memory usage: 293.8+ KB
```

```
df.tail()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	age
4172	F	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	12.5
4173	M	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	11.5
4174	M	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	10.5

```
df.isnull().sum()
```

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

```
df.shape
```

```
(4177, 9)
```

```
df_sex = df['Sex'].value_counts()
print(df_sex.head())
```

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

```
df.columns
```

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

```
sns.heatmap(df.isnull())
```

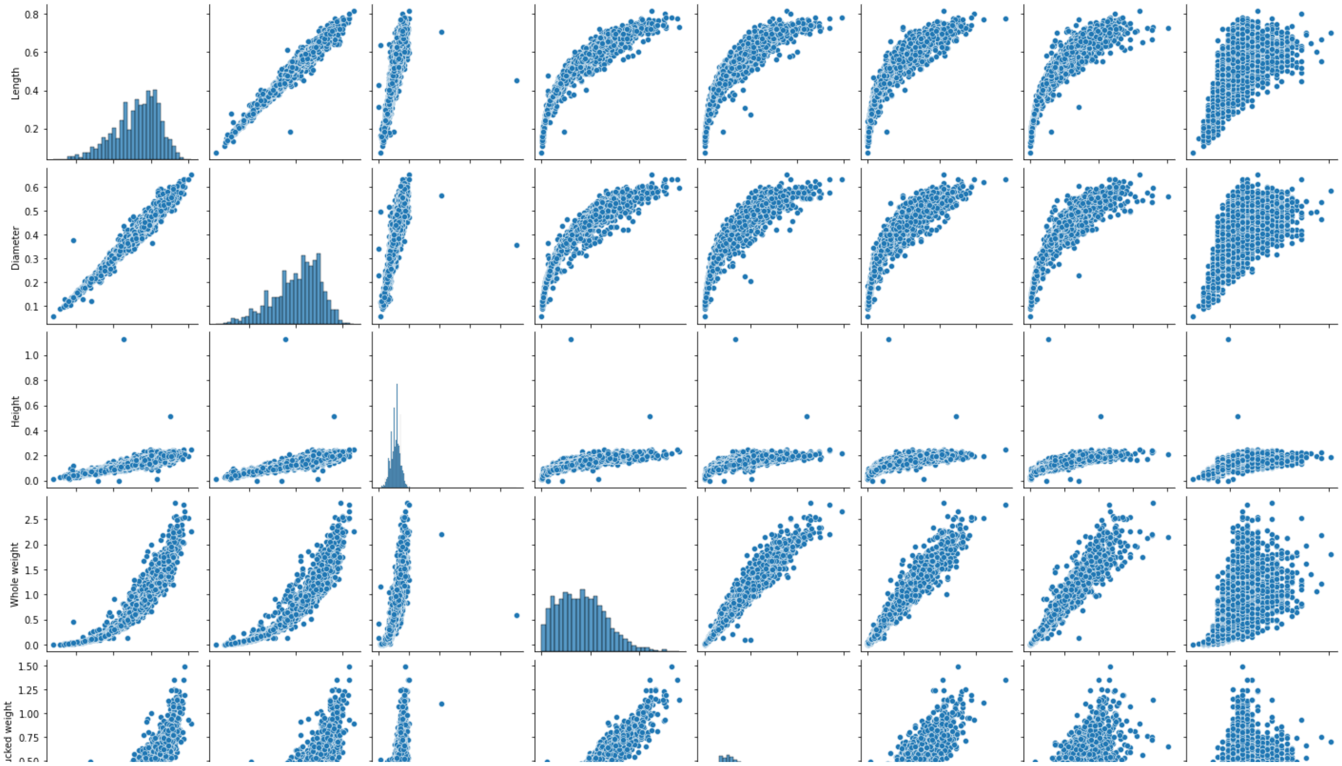
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f69bc41c690>
```

*Bivariate analysis

```
import seaborn as sns
```

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

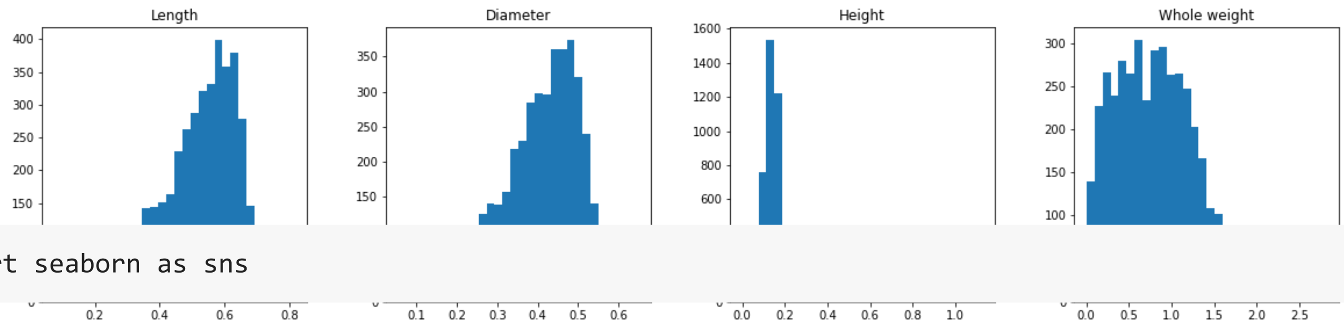
<seaborn.axisgrid.PairGrid at 0x7f69c4c29610>



*Multivariate analysis

```
df.hist(figsize=(20,10),grid=False, layout=(2, 4), bins = 30)
```

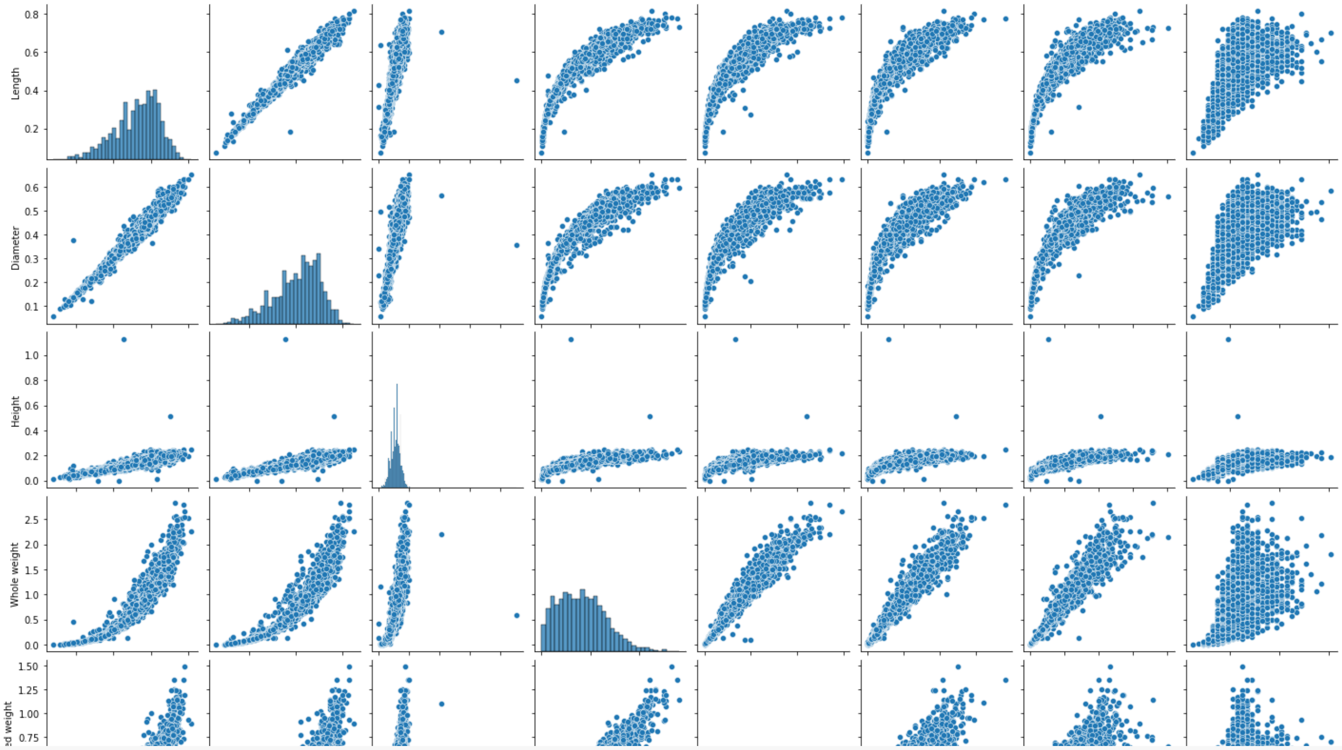
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f69bc69f650>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f69bc621710>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f69bc57e790>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f69bc533d90>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f69bc4f83d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f69bc4ae9d0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f69bc4dab10>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f69bc4295d0>]],
      dtype=object)
```




```
import seaborn as sns
```

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

<seaborn.axisgrid.PairGrid at 0x7f69c0b89590>



```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Sex             4177 non-null   object 
 1   Length          4177 non-null   float64
 2   Diameter        4177 non-null   float64
 3   Height          4177 non-null   float64
 4   Whole weight    4177 non-null   float64
 5   Shucked weight  4177 non-null   float64
 6   Viscera weight  4177 non-null   float64
 7   Shell weight    4177 non-null   float64
 8   age             4177 non-null   float64
dtypes: float64(8), object(1)
memory usage: 293.8+ KB
```

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

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: DeprecationWarning: `np
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/relea
```

```
numerical_featuresa
```

```
Index(['Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight',
```

```
       'Viscera weight', 'Shell weight', 'age'],
      dtype='object')
```

```
categorical_features
```

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

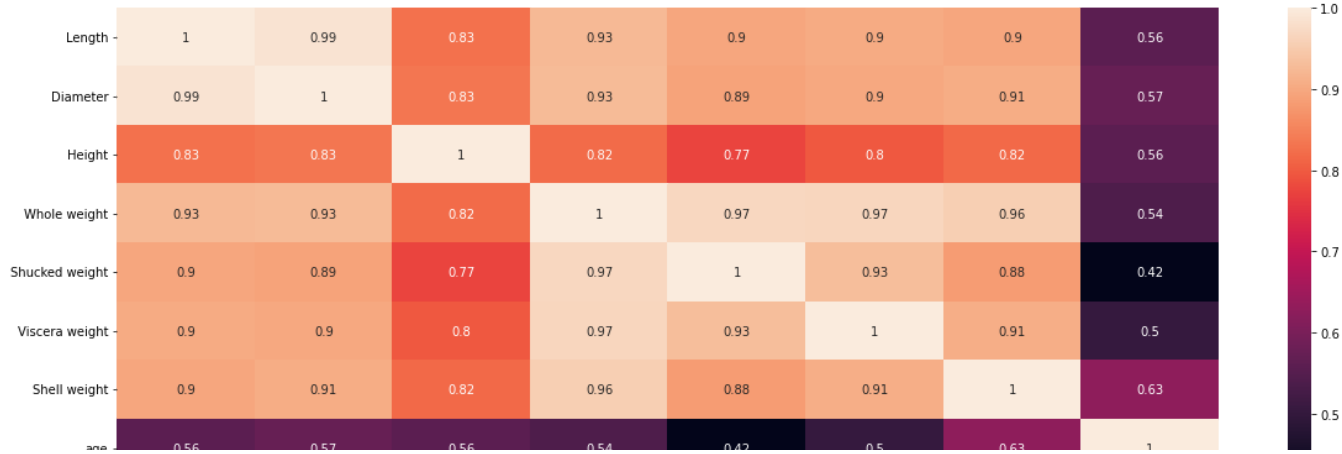
```
missing_values = df.isnull().sum().sort_values(ascending = False)
percentage_missing_values = (missing_values/len(df))*100
pd.concat([missing_values, percentage_missing_values], axis = 1, keys= ['Missing values','%mi
```

	Missing values	%missing_values
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

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

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f69bcd88f50>
```




```
Whole Weight is almost linearly varying with all other features except age
Heigh has least linearity with remaining features
Age is most linearly proprtional with Shell Weight followed by Diameter and length
Age is least correlated with Shucked Weight
```

- All numerical features but 'sex'

```
- Though features are not normaly distributed, are close to normality
- None of the features have minimum = 0 except Height (requires re-check)
- Each feature has difference scale range
```

4.Perform descriptive Statitics on the dataset

```
import math
import statistics
import numpy as np
import scipy.stats
import pandas as pd
import seaborn as sns
```

Double-click (or enter) to edit

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

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f69bcb798d0>
```


```
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
```

```
df.head()
```

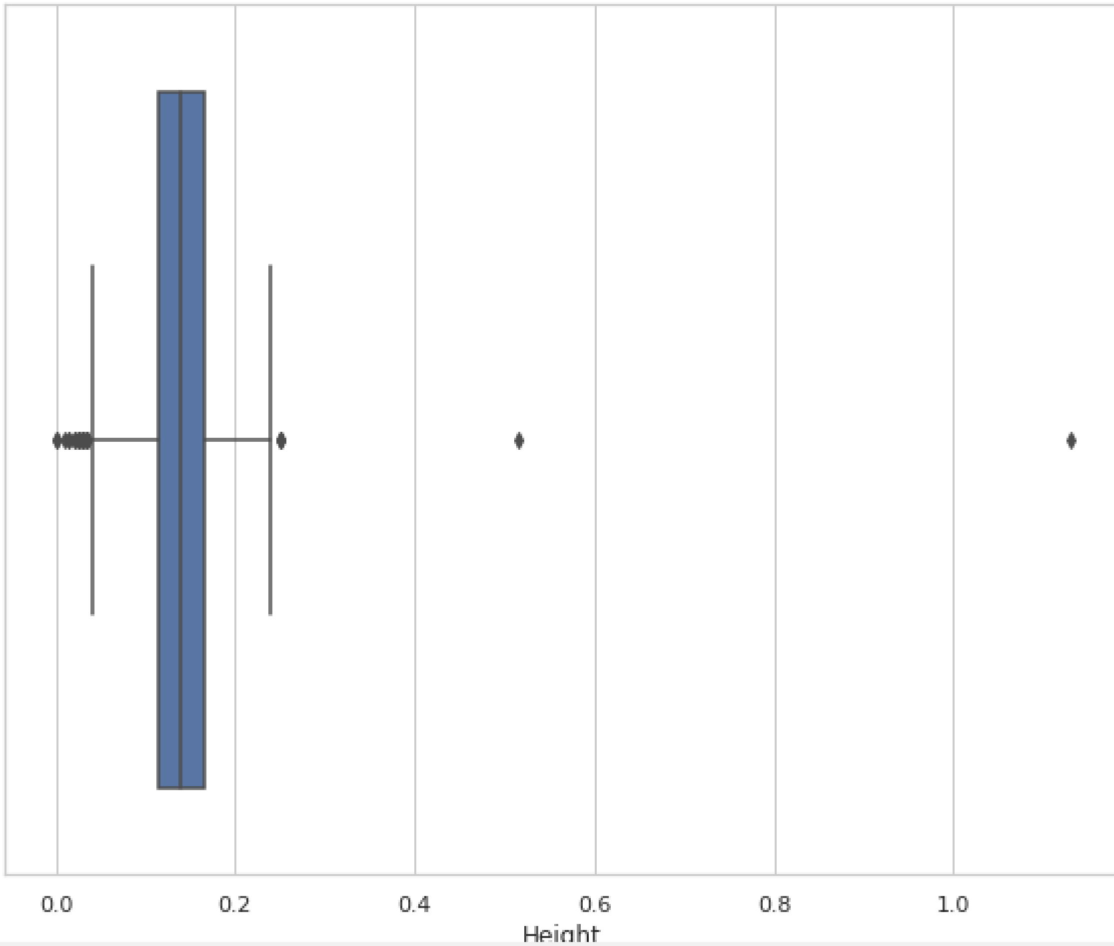
	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

```
import pandas as pd
path = "/content/drive/MyDrive/Untitled folder/abalone (1).csv"
df = pd.read_csv(path)
df['Rings'].mode()
```

```
0    9
dtype: int64
```

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set(style="whitegrid")
plt.figure(figsize=(10,8))
ax=sns.boxplot(x='Height',data=df,orient="v")
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_core.py:1326: UserWarning: Vertical orie
  warnings.warn(single_var_warning.format("Vertical", "x"))
```



Double-click (or enter) to edit

5. Check for Missing values and deal with them

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Sex             4177 non-null   object
 1   Length          4177 non-null   float64
 2   Diameter        4177 non-null   float64
 3   Height          4177 non-null   float64
 4   Whole weight    4177 non-null   float64
```

```
 5   Shucked weight  4177 non-null   float64
 6   Viscera weight  4177 non-null   float64
 7   Shell weight    4177 non-null   float64
 8   Rings           4177 non-null   int64
dtypes: float64(7), int64(1), object(1)
memory usage: 293.8+ KB
```

```
print(df.isnull().sum())
```

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

```
updated_df = df.dropna(axis=1)
```

```
updated_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Sex             4177 non-null   object
 1   Length          4177 non-null   float64
 2   Diameter        4177 non-null   float64
 3   Height          4177 non-null   float64
 4   Whole weight    4177 non-null   float64
 5   Shucked weight  4177 non-null   float64
 6   Viscera weight  4177 non-null   float64
 7   Shell weight    4177 non-null   float64
 8   Rings           4177 non-null   int64
dtypes: float64(7), int64(1), object(1)
memory usage: 293.8+ KB
```

6. Find the outliers and replace them outliers

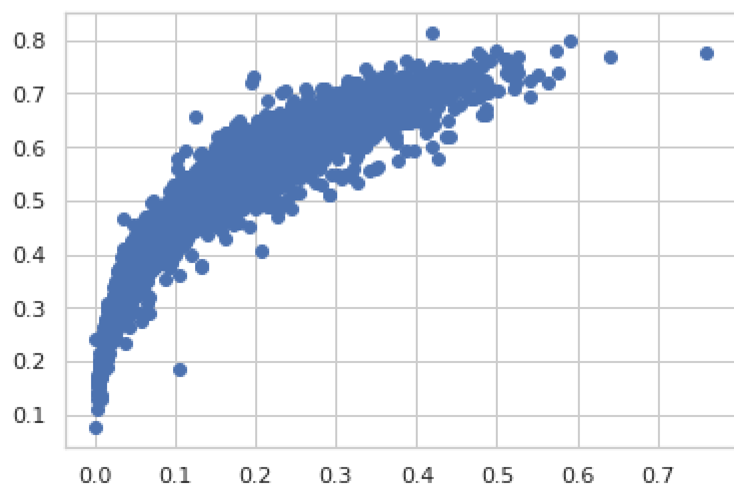
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
```

```
 ---  ------          --------------  -----
  0   Sex             4177 non-null   object
  1   Length          4177 non-null   float64
  2   Diameter        4177 non-null   float64
  3   Height          4177 non-null   float64
  4   Whole weight    4177 non-null   float64
  5   Shucked weight  4177 non-null   float64
  6   Viscera weight  4177 non-null   float64
  7   Shell weight    4177 non-null   float64
  8   Rings           4177 non-null   int64
dtypes: float64(7), int64(1), object(1)
memory usage: 293.8+ KB
```

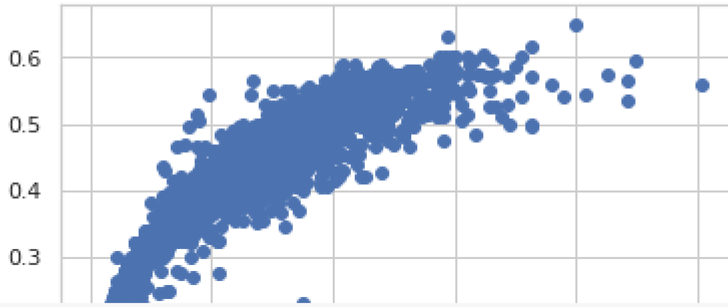
```
# outlier handling
df = pd.get_dummies(df)
dummy_df = df
```

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



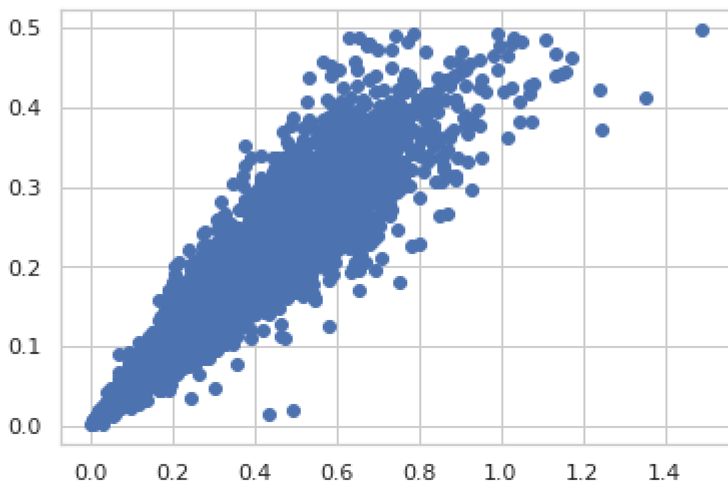

```
df.drop(df[(df['Viscera weight'] > 0.5) &
          (df['Whole weight'] < 20)].index, inplace = True)
df.drop(df[(df['Viscera weight']<0.5) & (
df['Whole weight'] > 25)].index, inplace = True)
```

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



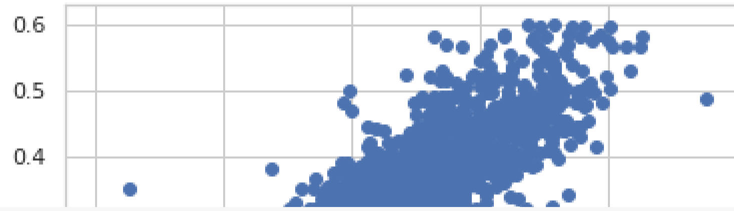

```
df.drop(df[(df['Shell weight'] > 0.6) &
          (df['Viscera weight'] < 25)].index, inplace = True)
df.drop(df[(df['Shell weight']<0.8) & (
df['Viscera weight'] > 25)].index, inplace = True)
```

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



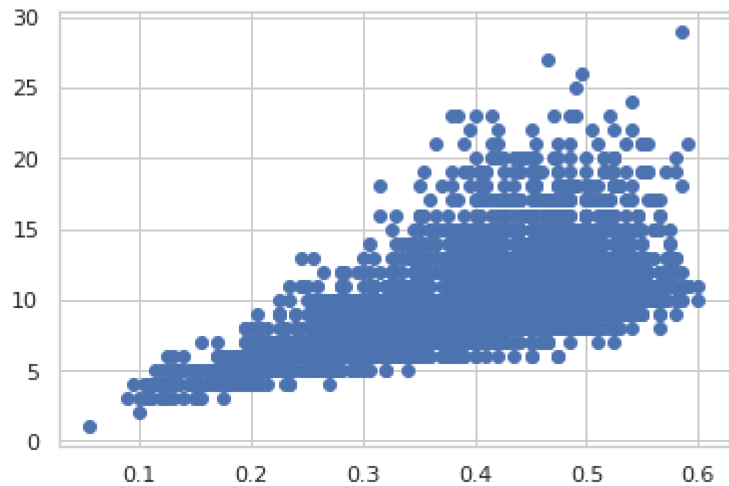

```
df.drop(df[(df['Shucked weight'] >= 1) &
          (df['Height'] < 20)].index, inplace = True)
df.drop(df[(df['Viscera weight']<1) & (
df['Height'] > 20)].index, inplace = True)
```

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



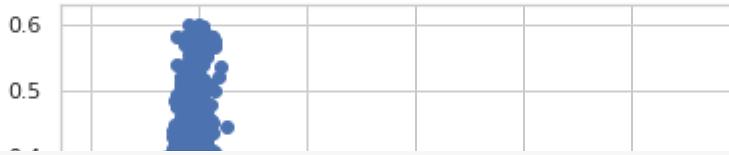

```
df.drop(df[(df['Whole weight'] >= 2.5) &
          (df['Height'] < 25)].index, inplace = True)
df.drop(df[(df['Whole weight']<2.5) & (
df['Height'] > 25)].index, inplace = True)
```

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



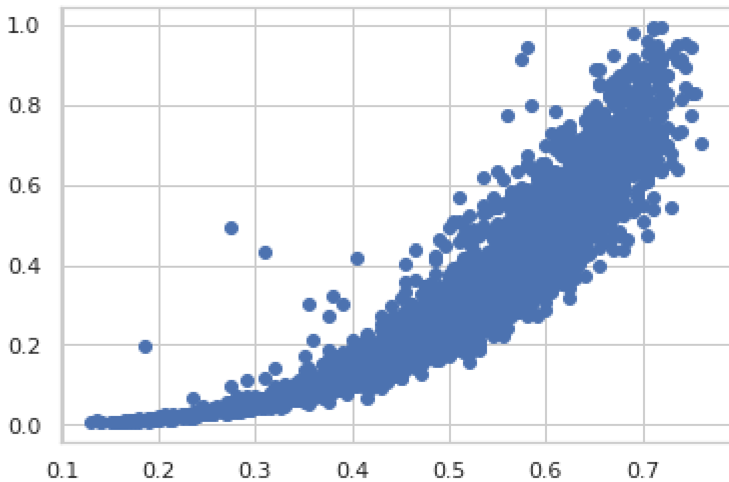

```
df.drop(df[(df['Diameter'] <0.1) &
          (df['Whole weight'] < 5)].index, inplace = True)
df.drop(df[(df['Diameter']<0.6) & (
df['Whole weight'] > 25)].index, inplace = True)
```

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




```
df.drop(df[(df['Height'] > 0.4) &
          (df['Length'] < 15)].index, inplace = True)
df.drop(df[(df['Height']<0.4) & (
df['Length'] > 25)].index, inplace = True)
```

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




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

7. Check for Categorical columns and perform encoding.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   Sex             4177 non-null   object 
 1   Length          4177 non-null   float64
 2   Diameter        4177 non-null   float64
 3   Height          4177 non-null   float64
```

```
 4   Whole weight    4177 non-null   float64
 5   Shucked weight  4177 non-null   float64
 6   Viscera weight  4177 non-null   float64
 7   Shell weight    4177 non-null   float64
 8   Rings           4177 non-null   int64
dtypes: float64(7), int64(1), object(1)
memory usage: 293.8+ KB
```

```
train_data=data.copy()
```

```
import numpy as np
import pandas as pd
path = "/content/drive/MyDrive/Untitled folder/abalone (1).csv"
df = pd.read_csv(path)
df.describe
print(data.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
```

```
print(data['Sex'].unique())
print(data['Length'].unique())
print(data['Diameter'].unique())
print(data['Height'].unique())
print(data['Whole weight'].unique())
print(data['Shucked weight'].unique())
print(data['Viscera weight'].unique())
print(data['Shell weight'].unique())
print(data['Rings'].unique())
```

```
['M' 'F' 'I']
[0.455 0.35  0.53  0.44  0.33  0.425 0.545 0.475 0.55  0.525 0.43  0.49
 0.535 0.47  0.5   0.355 0.365 0.45  0.38  0.565 0.615 0.56  0.58  0.59
 0.605 0.575 0.68  0.665 0.705 0.465 0.54  0.24  0.205 0.21  0.39  0.46
 0.325 0.52  0.4   0.485 0.405 0.445 0.245 0.505 0.595 0.31  0.555 0.57
 0.6   0.62  0.625 0.695 0.36  0.51  0.435 0.495 0.385 0.515 0.37  0.27
 0.375 0.7   0.71  0.265 0.305 0.345 0.65  0.28  0.175 0.17  0.635 0.645
 0.61  0.725 0.235 0.315 0.225 0.64  0.63  0.585 0.42  0.335 0.415 0.275
 0.295 0.075 0.13  0.11  0.16  0.23  0.3   0.32  0.655 0.66  0.2   0.165
 0.19  0.74  0.34  0.675 0.745 0.685 0.69  0.67  0.29  0.26  0.395 0.41
```

```
 0.22  0.255 0.735 0.155 0.48  0.195 0.25  0.18  0.15  0.215 0.73  0.715
 0.765 0.185 0.285 0.72  0.75  0.755 0.78  0.815 0.14  0.77  0.775 0.76
 0.135 0.8  ]
[0.365 0.265 0.42  0.255 0.3   0.415 0.425 0.37  0.44  0.38  0.35  0.405
 0.355 0.4   0.28  0.34  0.295 0.32  0.275 0.48  0.45  0.445 0.475 0.47
 0.56  0.525 0.55  0.29  0.335 0.175 0.15  0.375 0.245 0.41  0.36  0.31
 0.385 0.19  0.345 0.325 0.495 0.39  0.235 0.51  0.465 0.535 0.435 0.43
 0.395 0.305 0.195 0.54  0.26  0.2   0.33  0.23  0.285 0.52  0.455 0.205
 0.13  0.5   0.515 0.485 0.46  0.545 0.57  0.575 0.16  0.21  0.49  0.25
 0.27  0.505 0.215 0.225 0.055 0.1   0.09  0.12  0.53  0.145 0.22  0.6
 0.58  0.585 0.565 0.555 0.185 0.165 0.125 0.59  0.14  0.11  0.155 0.315
 0.24  0.17  0.18  0.105 0.595 0.135 0.625 0.63  0.61  0.65  0.62  0.605
 0.095 0.115 0.615]
[0.095 0.09  0.135 0.125 0.08  0.15  0.14  0.11  0.145 0.1   0.13  0.085
 0.155 0.165 0.185 0.18  0.175 0.2   0.105 0.045 0.055 0.05  0.12  0.07
 0.16  0.06  0.17  0.195 0.19  0.115 0.075 0.065 0.215 0.21  0.23  0.205
 0.22  0.04  0.01  0.03  0.035 0.225 0.24  0.235 0.02  0.025 0.015 0.
 0.515 0.25  1.13 ]
[0.514  0.2255 0.677  ... 1.176  1.0945 1.9485]
[0.2245 0.0995 0.2565 ... 0.727  0.137  0.9455]
[1.010e-01 4.850e-02 1.415e-01 1.140e-01 3.950e-02 7.750e-02 1.495e-01
 1.125e-01 1.510e-01 1.475e-01 8.100e-02 9.500e-02 1.710e-01 8.050e-02
 1.330e-01 8.700e-02 4.300e-02 7.500e-02 6.200e-02 4.900e-02 2.140e-01
 2.100e-01 3.010e-01 1.880e-01 2.720e-01 2.340e-01 2.190e-01 2.270e-01
 2.420e-01 2.805e-01 3.575e-01 3.925e-01 4.115e-01 1.240e-01 3.075e-01
 1.165e-01 2.035e-01 8.600e-02 9.100e-02 1.960e-01 2.350e-02 1.500e-02
 1.250e-02 4.500e-02 1.100e-01 2.550e-02 2.135e-01 1.110e-01 6.000e-02
 9.600e-02 1.055e-01 9.150e-02 1.755e-01 9.550e-02 1.200e-01 1.400e-02
 1.300e-01 1.600e-01 1.935e-01 8.000e-02 1.315e-01 1.015e-01 2.240e-01
 1.155e-01 4.050e-02 1.680e-01 9.850e-02 2.250e-01 2.610e-01 2.895e-01
 2.210e-01 1.890e-01 1.940e-01 1.595e-01 2.355e-01 2.520e-01 1.920e-01
 2.160e-01 2.225e-01 2.050e-01 2.075e-01 2.310e-01 1.675e-01 1.525e-01
 2.540e-01 3.180e-01 3.425e-01 3.880e-01 1.385e-01 1.320e-01 8.500e-02
 4.600e-02 1.915e-01 1.640e-01 2.405e-01 1.835e-01 1.830e-01 1.790e-01
 9.800e-02 1.345e-01 1.070e-01 6.350e-02 6.300e-02 1.460e-01 1.585e-01
 1.020e-01 1.720e-01 5.950e-02 1.035e-01 5.750e-02 1.725e-01 5.100e-02
 5.050e-02 5.450e-02 6.100e-02 2.635e-01 2.830e-01 3.600e-02 5.600e-02
 2.050e-02 3.450e-02 2.950e-02 2.055e-01 1.435e-01 3.060e-01 1.900e-01
 1.075e-01 1.080e-01 3.900e-02 6.500e-03 8.000e-03 2.620e-01 2.585e-01
 3.050e-01 1.310e-01 2.425e-01 2.615e-01 1.785e-01 2.475e-01 2.575e-01
 2.970e-01 3.980e-01 4.830e-01 4.515e-01 4.080e-01 3.085e-01 4.090e-01
 5.410e-01 1.690e-01 2.330e-01 1.845e-01 1.800e-02 3.700e-02 3.750e-02
 4.750e-02 2.165e-01 2.000e-01 2.660e-01 2.680e-01 3.670e-01 2.965e-01
 3.360e-01 3.315e-01 3.350e-01 2.710e-01 1.815e-01 1.965e-01 1.535e-01
 1.450e-01 1.780e-01 2.130e-01 1.465e-01 1.550e-01 7.100e-02 1.085e-01
 6.700e-02 1.685e-01 5.200e-02 1.625e-01 4.150e-02 7.850e-02 1.355e-01
 1.280e-01 1.635e-01 7.000e-02 1.150e-01 7.550e-02 1.235e-01 1.440e-01
 1.250e-01 8.200e-02 2.095e-01 1.580e-01 1.930e-01 2.870e-01 2.900e-02
```

```
data['Sex'].value_counts()
data['Length'].value_counts()
data['Diameter'].value_counts()
data['Height'].value_counts()
data['Whole weight'].value_counts()
data['Shucked weight'].value_counts()
```

```
data['Viscera weight'].value_counts()
data['Shell weight'].value_counts()
data['Rings'].value_counts()
```

```
9     689
10    634
8     568
11    487
7     391
12    267
6     259
13    203
14    126
5     115
15    103
16     67
17     58
4      57
18     42
19     32
20     26
3      15
21     14
23      9
22      6
27      2
24      2
1       1
26      1
29      1
2       1
25      1
Name: Rings, dtype: int64
```

```
one_hot_encoded_data = pd.get_dummies(data, columns = ['Sex', 'Height'])
print(one_hot_encoded_data)
```

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

      Shell weight  Rings  Sex_F  Sex_I  Sex_M  ...  Height_0.21  \
0           0.1500     15      0      0      1  ...            0
1           0.0700      7      0      0      1  ...            0
2           0.2100      9      1      0      0  ...            0
```

```
3         0.1550    10     0      0      1  ...          0
4         0.0550     7     0      1      0  ...          0
...          ...   ...   ...    ...    ...  ...        ...
4172      0.2490    11     1      0      0  ...          0
4173      0.2605    10     0      0      1  ...          0
4174      0.3080     9     0      0      1  ...          0
4175      0.2960    10     1      0      0  ...          0
4176      0.4950    12     0      0      1  ...          0

      Height_0.215  Height_0.22  Height_0.225  Height_0.23  Height_0.235  \
0                0            0             0            0             0
1                0            0             0            0             0
2                0            0             0            0             0
3                0            0             0            0             0
4                0            0             0            0             0
...            ...          ...           ...          ...           ...
4172             0            0             0            0             0
4173             0            0             0            0             0
4174             0            0             0            0             0
4175             0            0             0            0             0
4176             0            0             0            0             0

      Height_0.24  Height_0.25  Height_0.515  Height_1.13
0               0            0             0            0
1               0            0             0            0
2               0            0             0            0
3               0            0             0            0
4               0            0             0            0
...           ...          ...           ...          ...
4172            0            0             0            0
4173            0            0             0            0
4174            0            0             0            0
4175            0            0             0            0
4176            0            0             0            0

[4177 rows x 61 columns]
```

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

8. Split the data into dependent and independent variables

```
X = df.iloc[:, :-1].values
print(X)
```

```
[['M' 0.455 0.365 ... 0.2245 0.101 0.15]
 ['M' 0.35 0.265 ... 0.0995 0.0485 0.07]
 ['F' 0.53 0.42 ... 0.2565 0.1415 0.21]
 ...
```

```
 ['M' 0.6 0.475 ... 0.5255 0.2875 0.308]
 ['F' 0.625 0.485 ... 0.531 0.261 0.296]
 ['M' 0.71 0.555 ... 0.9455 0.3765 0.495]]
```

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

```
X= data.iloc[:,:-1].values
```

```
y= data.iloc[:,3].values
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
```

```
import pandas as pd
path = "/content/drive/MyDrive/Untitled folder/abalone (1).csv"
df = pd.read_csv(path)
df.describe
df.head()
```

	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

```
plt.scatter(df['Length'],df['Rings'])
```

```
<matplotlib.collections.PathCollection at 0x7f69b89d8b50>
```

```
X = df[['Length','Diameter']]
Y = df['Rings']
X.head(10)
```

	Length	Diameter
0	0.455	0.365
1	0.350	0.265
2	0.530	0.420
3	0.440	0.365
4	0.330	0.255
5	0.425	0.300
6	0.530	0.415
7	0.545	0.425
8	0.475	0.370
9	0.550	0.440

```
x_train, x_test,y_train,y_test = train_test_split(X,Y,test_size =0.2)
# print the data
x_train
```

```
from sklearn.linear_model import LinearRegression
clf = LinearRegression()
```

3938 0.350 0.275

```
clf.fit(x_train,y_train)
```

```
LinearRegression()
```

```
clf.predict(x_test)
```

```
array([10.73921289, 11.30378172,  8.35848475,  9.58492883,  8.49269097,
        8.52959078, 12.25593665, 10.38688701, 10.24092738,  9.87097139,
        7.32665351,  8.02542856, 10.09496775,  8.26117833, 11.03536927,
        7.33840692, 12.83225889, 10.81137291, 10.03456113,  6.25204576,
       10.08909104, 11.6368379 , 10.7937428 , 11.11504558, 11.64271461,
       11.51602467,  9.76191156,  9.53039892,  7.2721236 ,  9.11178972,
       12.40777298, 10.40451712, 10.04043783,  9.4641156 ,  8.85677026,
       12.75422216,  9.06901321,  8.91717688, 11.84908086,  8.74771043,
        9.72501176,  6.67065496,  8.6017508 , 11.21071242,  5.83931327,
        9.4699923 , 11.69136782, 11.03536927,  6.25204576,  7.99440546,
        6.71930817,  9.48174571, 11.88598066, 10.50770024,  8.6076275 ,
       10.50182354, 11.47324816, 11.04876226, 11.32141183,  9.56142202,
       10.18052076, 12.10410031,  8.54134418, 11.06051567,  7.57579626,
        8.87440037, 10.60500666, 10.61088337, 10.14949766,  9.63358204,
       11.31553513,  9.12354312,  8.96583009, 10.19227417, 10.71406649,
        9.12354312, 11.07814578, 12.0747168 , 10.70818979, 11.6250845 ,
       10.60500666, 11.46737146,  9.16631963, 11.79455094, 10.84827272,
        9.62770534, 11.52190137, 11.47912487,  6.26379917, 11.73414432,
       11.42459495,  6.50706522,  8.45579117, 10.05219124,  7.16894047,
       12.30458986,  9.83994829, 10.08321434, 12.18965333, 12.55373261,
       12.48744929, 10.70231308,  9.23260295,  9.62182863,  7.62444947,
       10.11259786, 11.6250845 , 11.48500157,  9.22084955, 10.29545729,
        9.49349911, 11.99504049, 10.51945365,  7.47848984, 10.91455604,
       11.43634836,  9.8827248 ,  9.73088846, 11.68549111, 14.39927544,
       11.5999381 ,  6.15473934,  7.74526271, 10.14949766, 11.35831163,
       11.33904194, 11.64859131,  8.73431744,  8.75358714, 11.07814578,
        6.25204576, 12.66279244,  8.75358714,  7.42395993, 11.78867424,
       11.83145075, 12.10997702, 11.51602467, 11.38181845, 11.39357185,
        6.81661459,  7.32665351,  9.94900812,  9.87684809,  9.02036   ,
       11.6309612 ,  8.75358714, 10.64190647, 11.01773916, 11.83732745,
       11.58230799,  8.40126126, 11.99504049, 11.52190137, 11.26688192,
       11.56467788, 11.15782209,  9.48174571,  6.8652678 ,  9.17219633,
        9.03962969,  7.54477316, 11.34491864,  6.07670261, 10.7627197 ,
       11.46149476, 12.66866914,  9.0572598 , 10.13774425, 11.06639237,
        9.10003631,  7.93399885, 10.24092738, 11.56467788, 12.80287537,
        7.32665351, 11.08402248, 11.78867424,  8.76534054, 10.36174061,
        5.30576754,  9.94900812, 12.74834545, 10.14362096,  6.09433273,
        7.47261314, 12.03781699,  9.0513831 , 11.67373771,  7.54477316,
        9.62770534, 12.19553003,  9.77954167, 11.74589773,  9.8827248 ,
        8.44403776,  8.86852367,  9.33578608, 11.74002103,  6.73106158,
        9.18394974,  9.16044293, 12.35324307, 11.58230799,  9.65121215,
       12.9236886 ,  7.83669243,  6.71930817,  9.73088846, 11.57643129,
       12.18965333,  9.82819488,  7.42983663, 10.56223016,  8.35848475,
```

```
       10.85414942,  5.93661969,  9.00272989, 10.7878661 ,  5.58429381,
        8.77121725,  6.91979772, 11.38181845, 11.54704777,  9.67048184,
        9.0513831 ,  9.49937582,  9.32403267, 10.22329727,  8.75358714,
       10.98083936, 10.61676007, 10.301334  , 10.04631454, 10.54460005,
       10.95145584,  9.17807304, 10.38101031,  9.99766133, 11.84320415,
       10.23505068, 12.71732235,  9.79717178,  4.77058222,  9.16044293,
        7.52714305, 12.28108305,  7.32077681, 11.64271461, 10.60500666,
       11.42459495,  8.66215742,  8.70493393, 10.86590283, 10.50770024,
        9.32403267,  6.70755477, 11.51602467, 10.04631454, 10.13774425,
        5.69335364,  7.01710414,  9.42133909,  9.83994829, 13.38507431,
       10.53872334, 10.21742057, 10.64778317, 11.01186246,  8.87440037,
       10.57398356, 11.77104413, 10.32060369, 12.11585372, 12.20140673,
       12.30458986, 11.5747917 ,  9.57317542,  8.24942492,  8.36436145,
       10.66541328, 11.79455094,  9.77954167,  9.21497284, 11.72239092,
```

```
clf.score(x_test,y_test)
```

```
0.3480503858042926
```

9. Scale the independent variables

```
from sklearn.preprocessing import StandardScaler
```

```
import pandas as pd
from sklearn.preprocessing import StandardScaler

# Read Data from CSV
path = "/content/drive/MyDrive/Untitled folder/abalone (1).csv"
df = pd.read_csv(path)
df.describe
df.head()

# Initialise the Scaler
scaler = StandardScaler()

# To scale data
scaler.fit
```

```
<bound method StandardScaler.fit of StandardScaler()>
```

10.Split the data into training and testing

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

```
# read the dataset

path = "/content/drive/MyDrive/Untitled folder/abalone (1).csv"
df = pd.read_csv(path)
df.describe



# get the locations
X = df.iloc[:, :-1]
y = df.iloc[:, -1]

# split the dataset
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.05, random_state=0)
```

```
X_train
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
678	F	0.450	0.380	0.165	0.8165	0.2500	0.1915	0.2650
3009	I	0.255	0.185	0.065	0.0740	0.0305	0.0165	0.0200
1906	I	0.575	0.450	0.135	0.8245	0.3375	0.2115	0.2390
768	F	0.550	0.430	0.155	0.7850	0.2890	0.2270	0.2330
2781	M	0.595	0.475	0.140	1.0305	0.4925	0.2170	0.2780
...	...	...	...	...	...	...	...	...
1033	M	0.650	0.525	0.185	1.6220	0.6645	0.3225	0.4770
3264	F	0.655	0.500	0.140	1.1705	0.5405	0.3175	0.2850
1653	M	0.595	0.450	0.145	0.9590	0.4630	0.2065	0.2535
2607	F	0.625	0.490	0.165	1.1270	0.4770	0.2365	0.3185
2732	I	0.410	0.325	0.110	0.3260	0.1325	0.0750	0.1010

3968 rows × 8 columns

```
X_test
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
668	M	0.550	0.425	0.155	0.9175	0.2775	0.2430	0.3350
1580	I	0.500	0.400	0.120	0.6160	0.2610	0.1430	0.1935
3784	M	0.620	0.480	0.155	1.2555	0.5270	0.3740	0.3175
463	I	0.220	0.165	0.055	0.0545	0.0215	0.0120	0.0200
2615	M	0.645	0.500	0.175	1.5105	0.6735	0.3755	0.3775
...	...	...	...	...	...	...	...	...
1670	F	0.610	0.485	0.150	1.2405	0.6025	0.2915	0.3085
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

```
y_train
```

```
678     23
3009     4
1906    11
768     11
2781    10
        ..
1033    10
3264    12
1653    10
2607     9
2732     8
Name: Rings, Length: 3968, dtype: int64
```

```
y_test
```

```
668     13
1580     8
3784    11
463      5
2615    12
        ..
1670    12
3055    11
3366     5
1410    10
4035    11
Name: Rings, Length: 209, dtype: int64
```

11.Bulid the Model

```
pip install -U scikit-learn
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub]
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (1
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages (1
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (1
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: numpy>=1.14.6 in /usr/local/lib/python3.7/dist-packages (
```

```
pip install pandas
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub]
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (1.3.5)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (1
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from
```

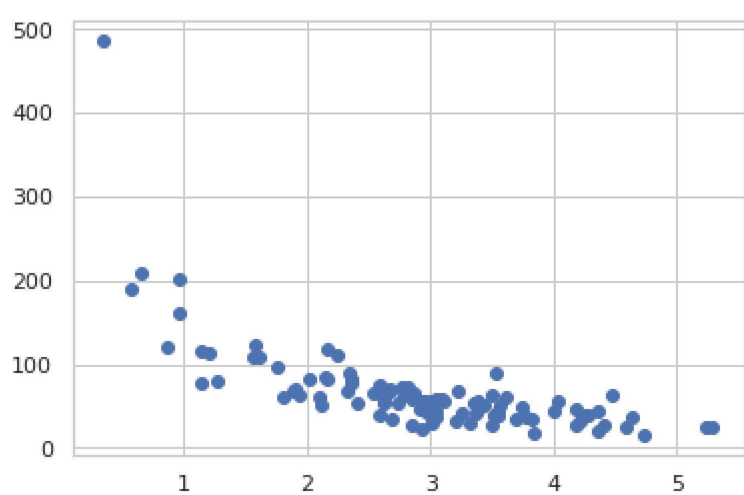
```
import pandas as pd

# reading csv file
path = "/content/drive/MyDrive/Untitled folder/abalone (1).csv"
data = pd.read_csv(path)
data.describe


# shape of dataset
print("Shape:", data.shape)

# column names
print("\nFeatures:", data.columns)

# storing the feature matrix (X) and response vector (y)
X = data[data.columns[:-1]]
y = data[data.columns[-1]]

# printing first 5 rows of feature matrix
print("\nFeature matrix:\n", X.head())

# printing first 5 values of response vector
print("\nResponse vector:\n", y.head())
```

```
Shape: (4177, 9)

Features: Index(['Sex', 'Length', 'Diameter', 'Height', 'Whole weight', 'Shucked weight
       'Viscera weight', 'Shell weight', 'Rings'],
      dtype='object')

Feature matrix:
   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
0         0.150
1         0.070
2         0.210
3         0.155
4         0.055

Response vector:
 0    15
1     7
2     9
3    10
4     7
Name: Rings, dtype: int64
```

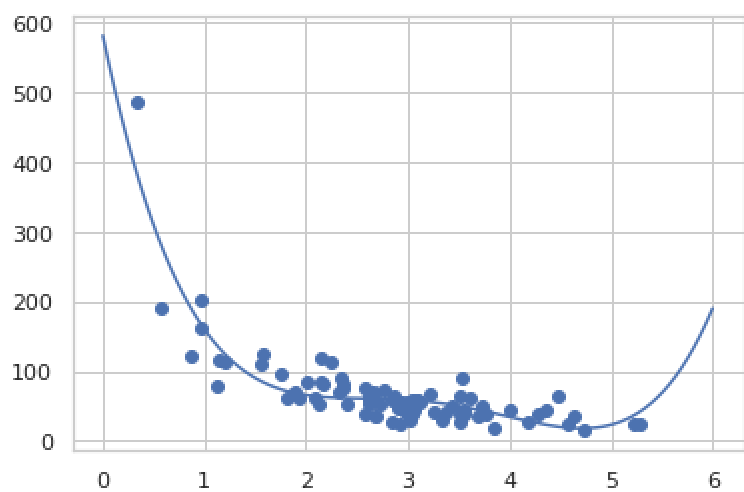
12.Train the Dataset

```
import numpy
import matplotlib.pyplot as plt
numpy.random.seed(2)

x = numpy.random.normal(3, 1, 100)
y = numpy.random.normal(150, 40, 100) / x

plt.scatter(x, y)
plt.show()
```




```
import numpy
import matplotlib.pyplot as plt
numpy.random.seed(2)
```

```
x = numpy.random.normal(3, 1, 100)
y = numpy.random.normal(150, 40, 100) / x

train_x = x[:80]
train_y = y[:80]

test_x = x[80:]
test_y = y[80:]

mymodel = numpy.poly1d(numpy.polyfit(train_x, train_y, 4))

myline = numpy.linspace(0, 6, 100)

plt.scatter(train_x, train_y)
plt.plot(myline, mymodel(myline))
plt.show()
```



13.Test the model

```
import numpy
from sklearn.metrics import r2_score
numpy.random.seed(2)

x = numpy.random.normal(3, 1, 100)
y = numpy.random.normal(150, 40, 100) / x

train_x = x[:80]
train_y = y[:80]

test_x = x[80:]
test_y = y[80:]
```

```
mymodel = numpy.poly1d(numpy.polyfit(train_x, train_y, 4))

r2 = r2_score(test_y, mymodel(test_x))

print(r2)
```

```
0.8086921460343566
```

```
print(mymodel(5))
```

```
22.8796259181172
```

14.Measure the performance using metrics

```
import tracemalloc
import pandas as pd
import dask.dataframe as dd
import time
```

```
def tracing_start():
    tracemalloc.stop()
    print("nTracing Status : ", tracemalloc.is_tracing())
    tracemalloc.start()
    print("Tracing Status : ", tracemalloc.is_tracing())
def tracing_mem():
    first_size, first_peak = tracemalloc.get_traced_memory()
    peak = first_peak/(1024*1024)
    print("Peak Size in MB - ", peak)
```

```
tracing_start()
start = time.time()
sq_list1 = [elem + elem**2 for elem in range(1,1000)]
#print(sq_list1)
end = time.time()
print("time elapsed {} milli seconds".format((end-start)*1000))
tracing_mem()
```

```
nTracing Status :  False
Tracing Status :  True
time elapsed 1.9772052764892578 milli seconds
Peak Size in MB -  0.047112464904785156
```

```
tracing_start()
start = time.time()
list_word = ["Quantify","performance","improvements","in","Python"]
```

```
s = ""
for substring in list_word:
    s += substring + " "
print(s)
end = time.time()
print("time elapsed {} milli seconds".format((end-start)*1000))
tracing_mem()
```

```
nTracing Status :  False
Tracing Status :  True
Quantify performance improvements in Python
time elapsed 0.45180320739746094 milli seconds
Peak Size in MB -  0.03258228302001953
```

```
tracing_start()
start = time.time()
a = [2,3,3,2,5,4,4,6,5,7,7,3,3,4,7,2,5,2,5]
b = []
for i in a:
    if i not in b:
        b.append(i)
print(b)
end = time.time()
print("time elapsed {} milli seconds".format((end-start)*1000))
tracing_mem()
```

```
nTracing Status :  False
Tracing Status :  True
[2, 3, 5, 4, 6, 7]
time elapsed 1.0797977447509766 milli seconds
Peak Size in MB -  0.011893272399902344
```