657a-as1-ques1

February 9, 2023

[124]:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
df = pd.read_csv("D:/UWaterloo/Data Knowledge and Modelling/Assignment 1/
 ↪abalone.csv", names = ['Sex', 'Length', 'Diameter', 'Height', 'Whole_weight',
                         'Sucked_weight', 'Viscera_weight', 'Shell_weight',␣
 ↪'Rings'], sep = ',')
warnings.filterwarnings('ignore')
```

[125]:

```
df.head()
```

[125]:

```
  Sex  Length  Diameter  Height  Whole_weight  Sucked_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
```

[126]:

```
df.isna().any()
```

[126]:

```
Sex               False
Length            False
Diameter          False
Height            False
Whole_weight      False
Sucked_weight     False
Viscera_weight    False
Shell_weight      False
Rings             False
dtype: bool
```

There is no missing data in any of the feature columns. Sex is the only categorical data, with the rest of the features being numerical.

[127]:

```
len(df)
```

[127]:

```
4177
```

[128]:

```
df.columns
```

[128]:

```
Index(['Sex', 'Length', 'Diameter', 'Height', 'Whole_weight', 'Sucked_weight',
       'Viscera_weight', 'Shell_weight', 'Rings'],
      dtype='object')
```

[129]:

```
df.describe()
```

[129]:

```
            Length     Diameter       Height  Whole_weight  Sucked_weight  \
count  4177.000000  4177.000000  4177.000000   4177.000000    4177.000000
mean      0.523992     0.407881     0.139516      0.828742       0.359367
std       0.120093     0.099240     0.041827      0.490389       0.221963
min       0.075000     0.055000     0.000000      0.002000       0.001000
25%       0.450000     0.350000     0.115000      0.441500       0.186000
50%       0.545000     0.425000     0.140000      0.799500       0.336000
75%       0.615000     0.480000     0.165000      1.153000       0.502000
max       0.815000     0.650000     1.130000      2.825500       1.488000

       Viscera_weight  Shell_weight        Rings
count     4177.000000   4177.000000  4177.000000
mean         0.180594      0.238831     9.933684
std          0.109614      0.139203     3.224169
min          0.000500      0.001500     1.000000
25%          0.093500      0.130000     8.000000
50%          0.171000      0.234000     9.000000
75%          0.253000      0.329000    11.000000
max          0.760000      1.005000    29.000000
```

[130]:

```
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   Sucked_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
```

```
[131]: df['Sex'].describe()
```

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

```
[132]: df['Sex'].value_counts()
```

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

```
[133]: df.median()
```

```
[133]: Length            0.5450
       Diameter          0.4250
       Height            0.1400
       Whole_weight      0.7995
       Sucked_weight     0.3360
       Viscera_weight    0.1710
       Shell_weight      0.2340
       Rings             9.0000
       dtype: float64
```

```
[134]: df.var()
```

```
[134]: Length             0.014422
       Diameter           0.009849
       Height             0.001750
       Whole_weight       0.240481
       Sucked_weight      0.049268
       Viscera_weight     0.012015
       Shell_weight       0.019377
       Rings             10.395266
       dtype: float64
```

```
[135]: df.skew()
```

```
[135]: Length           -0.639873
       Diameter         -0.609198
```

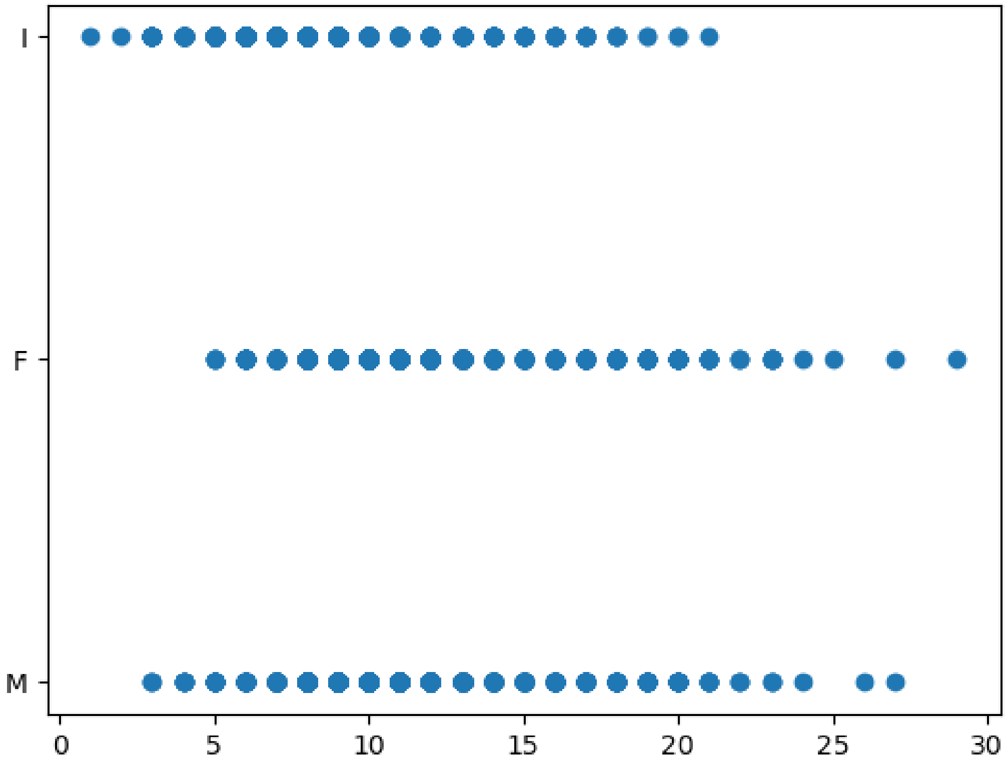
```
Height             3.128817
Whole_weight       0.530959
Sucked_weight      0.719098
Viscera_weight     0.591852
Shell_weight       0.620927
Rings              1.114102
dtype: float64
```

[136]:
```
df.kurtosis()
```

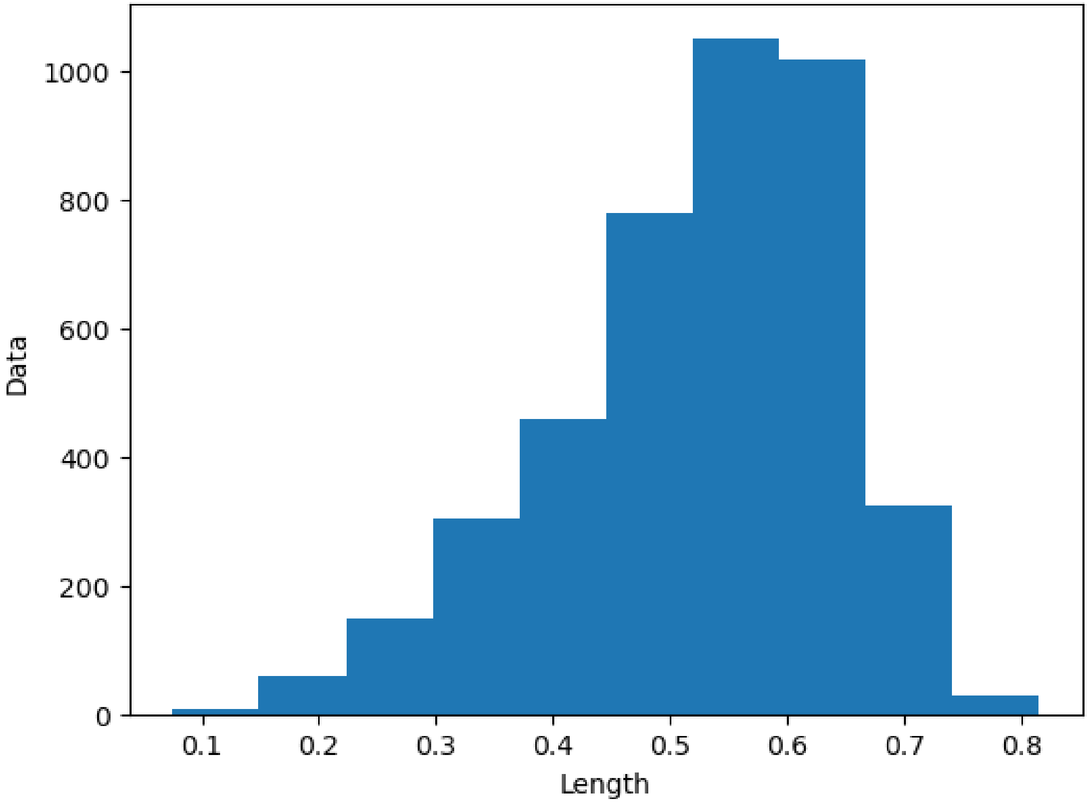
[136]:
```
Length             0.064621
Diameter          -0.045476
Height            76.025509
Whole_weight      -0.023644
Sucked_weight      0.595124
Viscera_weight     0.084012
Shell_weight       0.531926
Rings              2.330687
dtype: float64
```

[155]:
```
# Check any relationship between Sex and Rings
plt.scatter(y=df['Sex'], x=df['Rings'])
```

[155]:
```
<matplotlib.collections.PathCollection at 0x264ba2e7be0>
```



[137]:
```
plt.hist(df['Length'], bins=10)
plt.xlabel('Length')
plt.ylabel('Data')
plt.show()
df['Height'].describe()
df[df['Height'] == 0]
```



[137]:
```
      Sex  Length  Diameter  Height  Whole_weight  Sucked_weight  \
1257    I   0.430      0.34     0.0         0.428         0.2065
3996    I   0.315      0.23     0.0         0.134         0.0575

      Viscera_weight  Shell_weight  Rings
1257          0.0860        0.1150      8
3996          0.0285        0.3505      6
```

[138]:
```
df[['Whole_weight', 'Sucked_weight', 'Viscera_weight', 'Shell_weight']].
 ↪describe()
```

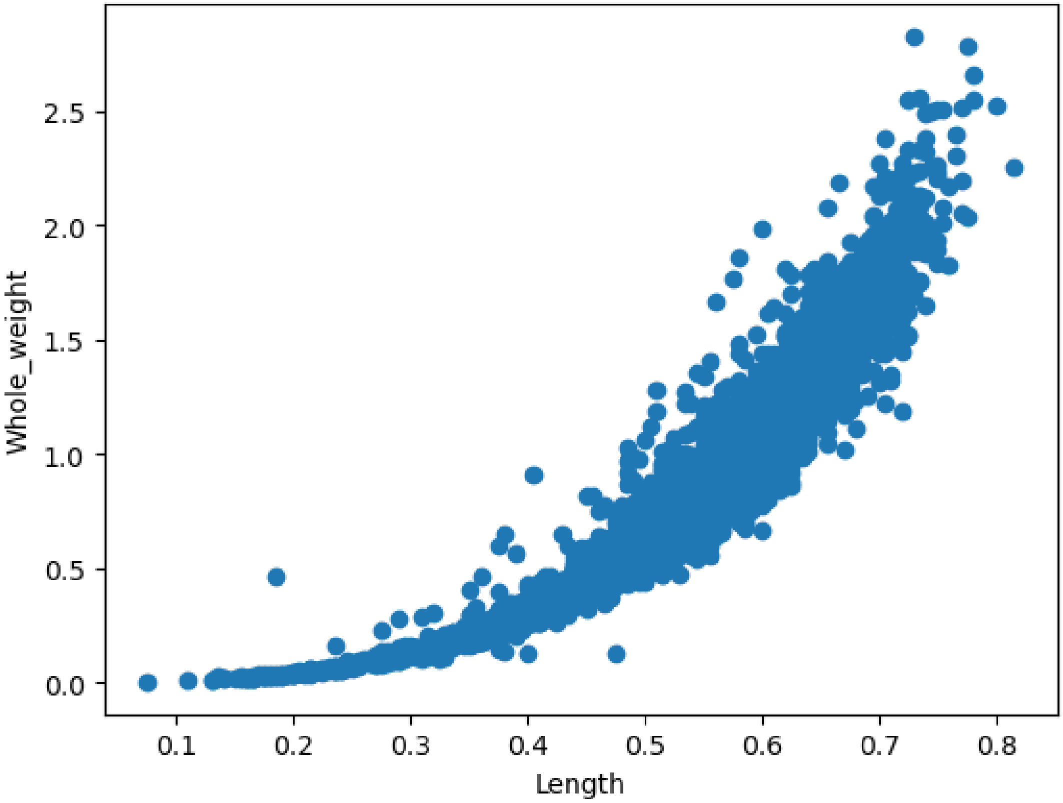
```
[138]:        Whole_weight  Sucked_weight  Viscera_weight  Shell_weight
       count   4177.000000    4177.000000     4177.000000   4177.000000
       mean       0.828742       0.359367        0.180594      0.238831
       std        0.490389       0.221963        0.109614      0.139203
       min        0.002000       0.001000        0.000500      0.001500
       25%        0.441500       0.186000        0.093500      0.130000
       50%        0.799500       0.336000        0.171000      0.234000
       75%        1.153000       0.502000        0.253000      0.329000
       max        2.825500       1.488000        0.760000      1.005000
```

1. The number of male fishes is higher than female and infants.
2. For the column of length, the mean is around 0.523 and median is 0.545 (denoted by the 50% value), which shows the distribution must be left skewed due to the presence of infants. The spread is less as SD is less.
3. The min of height feature is 0.000000, which can be an anomaly as other features have valid values. It can be a missing value that needs to be handled. SD is low so the normal distribution is not spread out a lot. It is close to the mean values.
4. Considering the data of all weights in the dataset, they are highly correlated and so we cannot use each weight against another to predict the age of the fish.

Scatter plot : Length vs Whole-weight

```
[139]: plt.scatter(x=df['Length'], y=df['Whole_weight'])
       plt.xlabel('Length')
       plt.ylabel('Whole_weight')
```

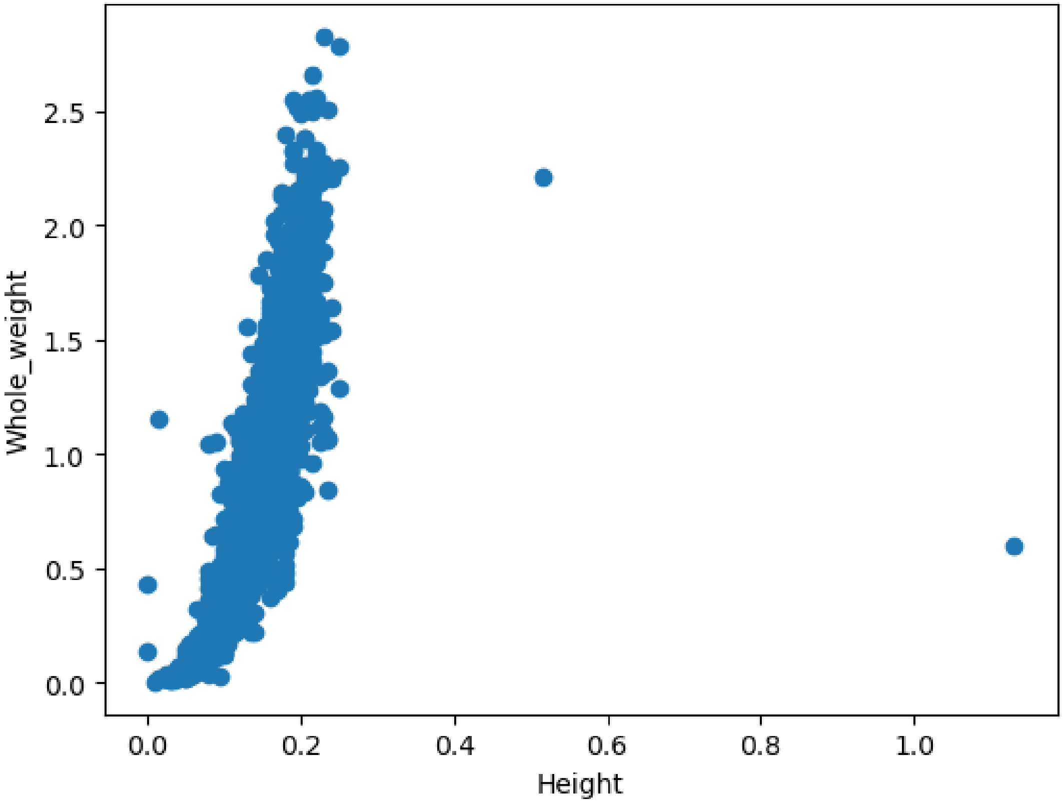
```
[139]: Text(0, 0.5, 'Whole_weight')
```



Height vs Whole weight Analysis: -> There are outliers when height is around 0.5 and height> 1 -> There is more or less a constant relationship between whole_weight and height when height is less than 0.2

[140]:
```
plt.scatter(x=df['Height'], y=df['Whole_weight'])
plt.xlabel('Height')
plt.ylabel('Whole_weight')
```

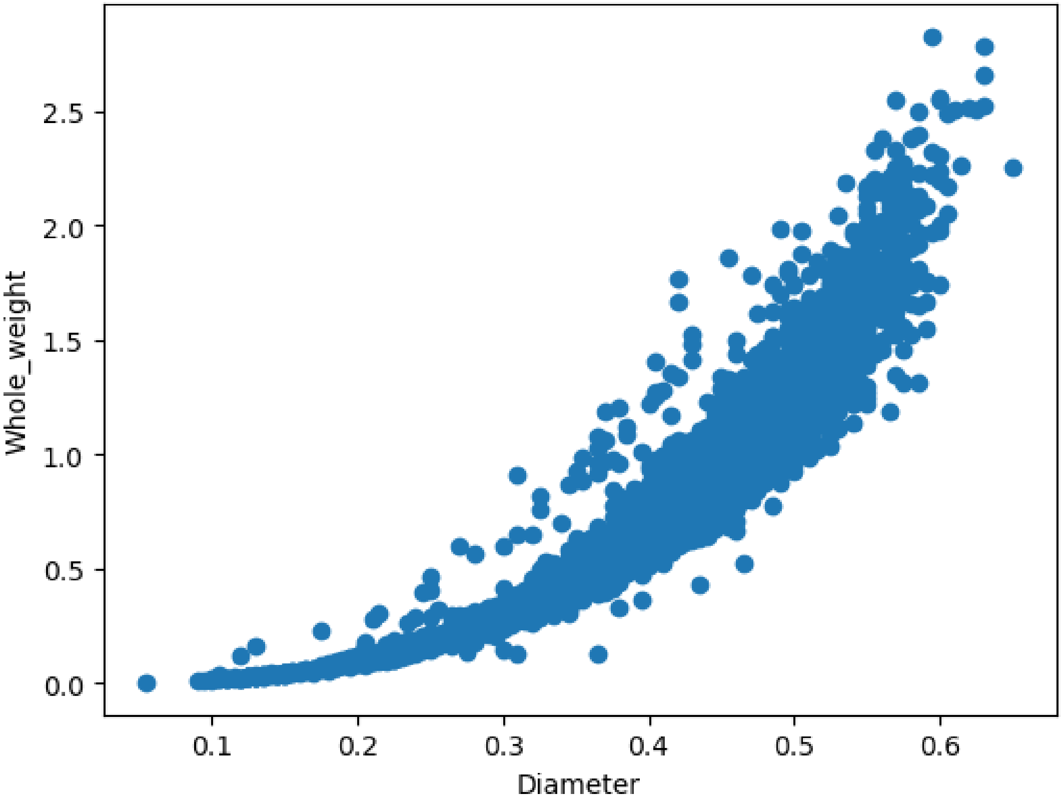
[140]:
```
Text(0, 0.5, 'Whole_weight')
```



Diameter vs Whole weight Analysis: -> There are not a lot of outliers and hence this feature can be used to derive a relationship.

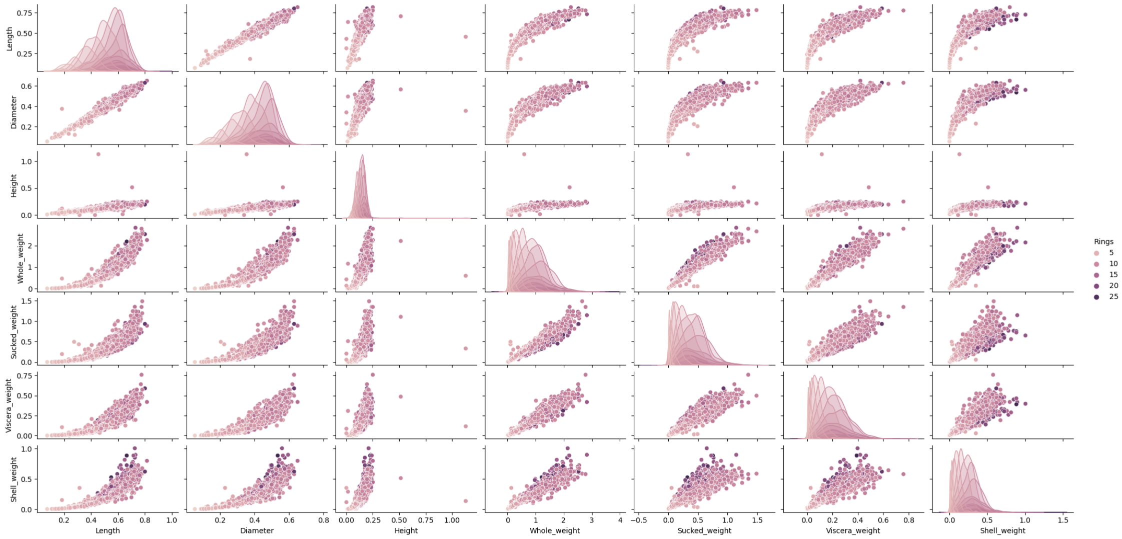
```
[141]: plt.scatter(x=df['Diameter'], y=df['Whole_weight'])
       plt.xlabel('Diameter')
       plt.ylabel('Whole_weight')
```

```
[141]: Text(0, 0.5, 'Whole_weight')
```



```
[142]: sns.pairplot(df, hue='Rings', height=1.5, aspect=2)
```

```
[142]: <seaborn.axisgrid.PairGrid at 0x264b4f79a90>
```



[143]:

```
df['Rings'].value_counts()
```

[143]:

```
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
```

The above data shows that the dataset is imbalanced as there are more samples available for Rings between 10 - 15. The number of samples in the current dataset is not very high (=4177) and hence, we can perform oversampling in order to balance it.

1 Splitting the dataset

[144]:

```
independent = df.iloc[:, 0:8]
dependent = df.iloc[:, 8]
```

[145]:

```
from sklearn.model_selection import train_test_split
ind_train, ind_test, dep_train, dep_test = train_test_split(independent,␣
 ↪dependent, test_size = 0.2, random_state=1)

# Balance the training dataset
from imblearn.over_sampling import RandomOverSampler
```

```
os = RandomOverSampler(random_state=1)
ind_train_sampled, dep_train_sampled = os.fit_resample(ind_train, dep_train)
```

[146]:

```
ind_train_sampled.shape, dep_train_sampled.shape
```

[146]:

```
((15039, 8), (15039,))
```

[147]:

```
print(f"The sampled training dataset is: {dep_train_sampled.value_counts()}")
print(f"The original training dataset is: {dep_train.value_counts()}")
```

```
The sampled training dataset is: 11    557
16    557
26    557
29    557
1     557
27    557
25    557
22    557
18    557
3     557
5     557
14    557
6     557
21    557
4     557
17    557
20    557
19    557
12    557
9     557
15    557
10    557
7     557
23    557
13    557
8     557
24    557
Name: Rings, dtype: int64
The original training dataset is: 9     557
10    491
8     461
11    396
7     311
6     206
12    204
13    164
14    102
5      95
```

```
15      82
16      57
17      46
4       46
18      32
19      25
20      22
3       13
21      11
23       7
22       5
27       2
24       2
25       1
1        1
29       1
26       1
Name: Rings, dtype: int64
```

2 Apply Z-Score normalization on the sampled dataset

[148]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
```

[149]:
```
ind_train_sampled.iloc[:, 1:] = sc.fit_transform(ind_train_sampled.iloc[:, 1:])
```

[153]:
```
ind_train_sampled, dep_train_sampled
```

[153]:
```
(      Sex    Length  Diameter    Height  Whole_weight  Sucked_weight  \
 0       M -0.403965 -0.443999 -0.489373     -0.796526      -0.791146
 1       I -1.578054 -1.544685 -1.612751     -1.459856      -1.436658
 2       I -0.022386 -0.017927 -0.662201     -0.740038      -0.643479
 3       I  0.564658  0.408146  0.115523     -0.017800       0.329009
 4       I  0.535306  0.550170  0.115523      0.018514       0.191890
 …     ..        …         …         …            …             …
 15034   F  1.034294  1.224784  0.634005      1.340332       1.419630
 15035   F  1.034294  1.224784  0.634005      1.340332       1.419630
 15036   F  1.034294  1.224784  0.634005      1.340332       1.419630
 15037   F  1.034294  1.224784  0.634005      1.340332       1.419630
 15038   F  1.034294  1.224784  0.634005      1.340332       1.419630

        Viscera_weight  Shell_weight
 0           -0.417408     -0.734652
 1           -1.440575     -1.400995
 2           -0.840371     -0.619599
 3           -0.199884     -0.140215
 4           -0.006530      0.152209
…               …           …
15034        1.012608      0.775408
15035        1.012608      0.775408
15036        1.012608      0.775408
15037        1.012608      0.775408
15038        1.012608      0.775408

[15039 rows x 8 columns],
0        11
1         4
2         8
3        11
4        13
         ..
15034    29
15035    29
15036    29
15037    29
15038    29
Name: Rings, Length: 15039, dtype: int64)
```

[ ]:

[ ]: