CS316 Lab 2: Preprocessing and cleaning the abalone dataset

Author:

- Name: Huu Khang Nguyen
- Student Number: 7402909

Import relevant libraries

In [ ]:
```
import pandas as pd
```

Load & Initial exploration for the abalone Dataset

In [ ]:
```
columns_name = ["Sex",
                "Length",
                "Diameter",
                "Height",
                "Whole weight",
                "Shucked weight",
                "Viscera weight",
                "Shell weight",
                "Rings"]

abalone_dataset = pd.read_csv('./data/abalone.data', names=columns_name)
```

In [ ]:
```
abalone_dataset.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 [ ]:
```
abalone_dataset.shape
```

Out[ ]:(4177, 9)

(1) Z-score normalization for `Length`

In [ ]:
```
mean = abalone_dataset['Length'].mean()
std = abalone_dataset['Length'].std()
var = abalone_dataset['Length'].var()

print("Mean: {}".format(mean))
print("Standard deviation: {}".format(std))
print("Variance: {}".format(var))
```

Mean: 0.5239920995930094
Standard deviation: 0.12009291256479956
Variance: 0.014422307648296592

In [ ]:
```
# Z score normalisation
abalone_dataset['Normalized Length'] = (abalone_dataset['Length'] - mean) / std
```

In [ ]:
```
abalone_dataset['Normalized Length'].head()
```

Out[ ]:
```
0   -0.574489
1   -1.448812
2    0.050027
3   -0.699393
4   -1.615350
Name: Normalized Length, dtype: float64
```

In [ ]:
```
print("Normalized Length Mean: {}".format(abalone_dataset['Normalized Length'].mean()))
print("Normalized Length Standard deviation: {}".format(abalone_dataset['Normalized Length'].std()))
print("Normalized Length Variance: {}".format(abalone_dataset['Normalized Length'].var()))
```

Normalized Length Mean: -5.919771894769329e-16
Normalized Length Standard deviation: 1.0
Normalized Length Variance: 1.0

(2) Create five bins for the attribute `Diameter`

Using `qcut()` for the appoximately same number of sample each bins, bins number (`q` parameter) will equal to 5

In [ ]:
```
binned_diameter = pd.qcut(abalone_dataset['Diameter'], q=5)
```

```
binned_diameter.value_counts()
```

Out[ ]:
```
(0.395, 0.45]     902
(0.054, 0.325]    863
(0.325, 0.395]    820
(0.45, 0.495]     803
(0.495, 0.65]     789
Name: Diameter, dtype: int64
```

In [ ]:
```
abalone_dataset['Diameter Binned'] = binned_diameter
```

In [ ]:
```
abalone_dataset['Diameter Binned']
```

Out[ ]:
```
0       (0.325, 0.395]
1       (0.054, 0.325]
2        (0.395, 0.45]
3       (0.325, 0.395]
4       (0.054, 0.325]
            ...
4172     (0.395, 0.45]
4173     (0.395, 0.45]
4174     (0.45, 0.495]
4175     (0.45, 0.495]
4176     (0.495, 0.65]
Name: Diameter Binned, Length: 4177, dtype: category
Categories (5, interval[float64, right]): [(0.054, 0.325] < (0.325, 0.395] < (0.395, 0.45] < (0.45, 0.495] < (0.495, 0.65]]
```

(3) One-hot-encoding the `Sex` attribute

In [ ]:
```
encoded_sex = pd.get_dummies(abalone_dataset['Sex'], prefix="Sex")
```

In [ ]:
```
abalone_dataset = abalone_dataset.join(encoded_sex)
```

In [ ]:
```
abalone_dataset.head()
```

Out[ ]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings	Normalized Length	Diameter Binned	Sex_F	Sex_I	Sex_M
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15	-0.574489	(0.325, 0.395]	0	0	1
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7	-1.448812	(0.054, 0.325]	0	0	1
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9	0.050027	(0.395, 0.45]	1	0	0
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10	-0.699393	(0.325, 0.395]	0	0	1
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7	-1.615350	(0.054, 0.325]	0	1	0

Show the unique one_hot_encoding values of the `Sex` attribute by dropping duplicate rows in the dataset

In [ ]:
```
unique_encoded_sex = encoded_sex.drop_duplicates()
```

In [ ]:
```
unique_encoded_sex
```

Out[ ]:

	Sex_F	Sex_I	Sex_M
0	0	0	1
2	1	0	0
4	0	1	0

(4) find and rank correlations between `Rings` with other continous values

In [ ]:
```
continous_cols = ["Length",
                  "Diameter",
                  "Height",
                  "Whole weight",
                  "Shucked weight",
                  "Viscera weight",
                  "Shell weight"]

map = {}
for col in continous_cols:
    key = 'Correlation between Rings and {}'.format(col)
    map[key] = abalone_dataset['Rings'].corr(abalone_dataset[col])
```

In [ ]:
```
asc_corr = sorted(map.items(), key=lambda item: item[1], reverse=True)
```

In [ ]:
```
for index,corr in enumerate(asc_corr):
    print(f"Rank {index+1}.", corr[0], corr[1])
```

Rank 1. Correlation between Rings and Shell weight 0.6275740445103217
Rank 2. Correlation between Rings and Diameter 0.5746598513059187
Rank 3. Correlation between Rings and Height 0.5574673244580373
Rank 4. Correlation between Rings and Length 0.5567195769296177
Rank 5. Correlation between Rings and Whole weight 0.5403896769239008
Rank 6. Correlation between Rings and Viscera weight 0.5038192487597712
Rank 7. Correlation between Rings and Shucked weight 0.42088365794521454

(5) Define 1 new attribute into the dataframe

I defined a variable `Age` here as this attribute can be calculated by number of rings + 1.5

In [ ]:
```
abalone_dataset['Age'] = abalone_dataset['Rings'] + 1.5
```
In [ ]:
```
abalone_dataset.head()
```
Out[ ]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings	Normalized Length	Diameter Binned	Sex_F	Sex_I	Sex_M	Age
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15	-0.574489	(0.325, 0.395]	0	0	1	16.5
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7	-1.448812	(0.054, 0.325]	0	0	1	8.5
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9	0.050027	(0.395, 0.45]	1	0	0	10.5
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10	-0.699393	(0.325, 0.395]	0	0	1	11.5
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7	-1.615350	(0.054, 0.325]	0	1	0	8.5

Perfect correlation, `Age` goes up if `Rings` goes up and vice-versa

In [ ]:
```
abalone_dataset['Rings'].corr(abalone_dataset['Age'])
```
Out[ ]:1.0

In [ ]: