HomeWork 3

1. sugar

In [6]:
```
# product has the least amount of sugar per ounce
import numpy as np
raw_data = np.loadtxt("cereal.csv", delimiter=',', dtype='str')
print(raw_data)
```

```
[['name' 'mfr' 'type' ... 'weight' 'cups' 'rating']
 ['100% Bran' 'N' 'C' ... '1' '0.33' '68.402973']
 ['100% Natural Bran' 'Q' 'C' ... '1' '1' '33.983679']
 ...
 ['Wheat Chex' 'R' 'C' ... '1' '0.67' '49.787445']
 ['Wheaties' 'G' 'C' ... '1' '1' '51.592193']
 ['Wheaties Honey Gold' 'G' 'C' ... '1' '0.75' '36.187559']]
```

In [20]:
```
#let extract the feature names
feature_names = raw_data[0]
print(feature_names)
```

```
['name' 'mfr' 'type' 'calories' 'protein' 'fat' 'sodium' 'fiber' 'carbo'
 'sugars' 'potass' 'vitamins' 'shelf' 'weight' 'cups' 'rating']
```

In [7]:
```
# let exclude feature names from raw_data to obtain data
data = raw_data[1:,:]
print(data)
```

```
[['100% Bran' 'N' 'C' ... '1' '0.33' '68.402973']
 ['100% Natural Bran' 'Q' 'C' ... '1' '1' '33.983679']
 ['All-Bran' 'K' 'C' ... '1' '0.33' '59.425505']
 ...
 ['Wheat Chex' 'R' 'C' ... '1' '0.67' '49.787445']
 ['Wheaties' 'G' 'C' ... '1' '1' '51.592193']
 ['Wheaties Honey Gold' 'G' 'C' ... '1' '0.75' '36.187559']]
```

In [22]:
```
# let extract the list of sugar per serving
sugar_per_serving = data[:,(feature_names=="sugars")]
print(sugar_per_serving)
```

```
[['6']
 ['8']
 ['5']
 ['0']
 ['8']
 ['10']
 ['14']
 ['8']
 ['6']
 ['5']
 ['12']
 ['1']
 ['9']
 ['7']
 ['13']
 ['3']
 ['2']
 ['12']
 ['13']
 ['7']
 ['0']
 ['3']
 ['10']
 ['5']
 ['13']
 ['11']
 ['7']
 ['10']
 ['12']
 ['12']
 ['15']
 ['9']
 ['5']
 ['3']
 ['4']
 ['11']
 ['10']
 ['11']
 ['6']
 ['9']
 ['3']
 ['6']
 ['12']
 ['3']
 ['11']
 ['11']
 ['13']
 ['6']
 ['9']
 ['7']
 ['2']
 ['10']
 ['14']
 ['3']
 ['0']
 ['0']
 ['6']
```

```
['-1']
['12']
['8']
['6']
['2']
['3']
['0']
['0']
['0']
['15']
['3']
['5']
['3']
['14']
['3']
['3']
['12']
['3']
['3']
['8']]
```

In [24]:
```
# let extract the list of ounce per serving
ounce_per_serving = data[:,(feature_names == "weight")]
print(ounce_per_serving)
```

```
[['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1.33']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1.25']
 ['1.33']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1.3']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1']
 ['1.5']
 ['1']
 ['1']
 ['1.33']
 ['1']
 ['1.25']
 ['1.33']
 ['1']
 ['0.5']
 ['0.5']
 ['1']
```

```
['1']
['1.33']
['1']
['1']
['1']
['1']
['0.83']
['1']
['1']
['1']
['1']
['1']
['1']
['1.5']
['1']
['1']
['1']
['1']
['1']
['1']]
```

In [25]:
```
# we can calcutate the list of sugar per ounce
sugar_per_ounce = sugar_per_serving.astype(float) / ounce_per_serving.astype(f
loat)
print(sugar_per_ounce)
```

```
[[ 6.        ]
 [ 8.        ]
 [ 5.        ]
 [ 0.        ]
 [ 8.        ]
 [10.        ]
 [14.        ]
 [ 6.01503759]
 [ 6.        ]
 [ 5.        ]
 [12.        ]
 [ 1.        ]
 [ 9.        ]
 [ 7.        ]
 [13.        ]
 [ 3.        ]
 [ 2.        ]
 [12.        ]
 [13.        ]
 [ 7.        ]
 [ 0.        ]
 [ 3.        ]
 [10.        ]
 [ 5.        ]
 [13.        ]
 [11.        ]
 [ 7.        ]
 [ 8.        ]
 [ 9.02255639]
 [12.        ]
 [15.        ]
 [ 9.        ]
 [ 5.        ]
 [ 3.        ]
 [ 4.        ]
 [11.        ]
 [10.        ]
 [11.        ]
 [ 6.        ]
 [ 6.92307692]
 [ 3.        ]
 [ 6.        ]
 [12.        ]
 [ 3.        ]
 [11.        ]
 [11.        ]
 [ 8.66666667]
 [ 6.        ]
 [ 9.        ]
 [ 5.26315789]
 [ 2.        ]
 [ 8.        ]
 [10.52631579]
 [ 3.        ]
 [ 0.        ]
 [ 0.        ]
 [ 6.        ]
```

```
[-1.        ]
[ 9.02255639]
[ 8.        ]
[ 6.        ]
[ 2.        ]
[ 3.        ]
[ 0.        ]
[ 0.        ]
[ 0.        ]
[15.        ]
[ 3.        ]
[ 5.        ]
[ 3.        ]
[ 9.33333333]
[ 3.        ]
[ 3.        ]
[12.        ]
[ 3.        ]
[ 3.        ]
[ 8.        ]]
```

In [23]:
```
# the least amount of sugar per ounce
# the index of the least amount of sugar per ounce
least_sugar_per_ounce = sugar_per_ounce.min()
index_sugar_per_ounce = np.argmin(sugar_per_ounce)
print("least_sugar_per_ounce:", least_sugar_per_ounce)
print("index_sugar_per_ounce:", index_sugar_per_ounce)
print("Name of this product is:", data[57,0])
```

```
least_sugar_per_ounce: -1.0
index_sugar_per_ounce: 57
Name of this product is: Quaker Oatmeal
```

In [26]:
```
# the average of sugar per ounce
average_sugar_per_ounce = sugar_per_ounce.mean()
print("average_sugar_per_ounce:", average_sugar_per_ounce)
```

```
average_sugar_per_ounce: 6.555489623158796
```

2. calories

In [27]:
```
# calories per gram of each cereal product
calories_per_gram = data[:, (feature_names=="calories")]
print(calories_per_gram)
```

```
[['70']
 ['120']
 ['70']
 ['50']
 ['110']
 ['110']
 ['110']
 ['130']
 ['90']
 ['90']
 ['120']
 ['110']
 ['120']
 ['110']
 ['110']
 ['110']
 ['100']
 ['110']
 ['110']
 ['110']
 ['100']
 ['110']
 ['100']
 ['100']
 ['110']
 ['110']
 ['100']
 ['120']
 ['120']
 ['110']
 ['100']
 ['110']
 ['100']
 ['110']
 ['120']
 ['120']
 ['110']
 ['110']
 ['110']
 ['140']
 ['110']
 ['100']
 ['110']
 ['100']
 ['150']
 ['150']
 ['160']
 ['100']
 ['120']
 ['140']
 ['90']
 ['130']
 ['120']
 ['100']
 ['50']
 ['50']
 ['100']
```

```
['100']
['120']
['100']
['90']
['110']
['110']
['80']
['90']
['90']
['110']
['110']
['90']
['110']
['140']
['100']
['110']
['110']
['100']
['100']
['110']]
```

In [30]:
```
#product with the highest value of calorie per gram
# let extract the highest calorie
highest_calories = calories_per_gram.astype(float).max()
print("highest_calories:",highest_calories)
```

```
highest_calories: 160.0
```

In [32]:
```
#let find the index of the highest calorie
index_highest_calories = np.argmax(calories_per_gram.astype(float))
print("index_highest_calories:", index_highest_calories)
```

```
index_highest_calories: 46
```

In [33]:
```
# the name of the product with the highest calorie per gram
print("Name of product with highest calories:", data[46,0])
```

```
Name of product with highest calories: Mueslix Crispy Blend
```

In [36]:
```
# product with the lowest calorie
# index of the lowest calorie
# product with lowest calorie name
print("lowest calories:", calories_per_gram.astype(float).min())
print("index lowest calories:", np.argmin(calories_per_gram.astype(float)))
print("Name product lowest calories:", data[3,0])
```

```
lowest calories: 50.0
index lowest calories: 3
Name product lowest calories: All-Bran with Extra Fiber
```

3. ratings

In [58]:
```
# five highest rated cereal product
cereal_ratings = data[:,-1]
n = 5
idx = np.argpartition(cereal_ratings, -n)[-n:]
#indices = idx[np.agrsort((-cereal_ratings)[idx])]
print(cereal_ratings)
print(idx)
print(data[[idx],0])
```

```
['68.402973' '33.983679' '59.425505' '93.704912' '34.384843' '29.509541'
 '33.174094' '37.038562' '49.120253' '53.313813' '18.042851' '50.764999'
 '19.823573' '40.400208' '22.736446' '41.445019' '45.863324' '35.782791'
 '22.396513' '40.448772' '64.533816' '46.895644' '36.176196' '44.330856'
 '32.207582' '31.435973' '58.345141' '40.917047' '41.015492' '28.025765'
 '35.252444' '23.804043' '52.076897' '53.371007' '45.811716' '21.871292'
 '31.072217' '28.742414' '36.523683' '36.471512' '39.241114' '45.328074'
 '26.734515' '54.850917' '37.136863' '34.139765' '30.313351' '40.105965'
 '29.924285' '40.692320' '59.642837' '30.450843' '37.840594' '41.503540'
 '60.756112' '63.005645' '49.511874' '50.828392' '39.259197' '39.703400'
 '55.333142' '41.998933' '40.560159' '68.235885' '74.472949' '72.801787'
 '31.230054' '53.131324' '59.363993' '38.839746' '28.592785' '46.658844'
 '39.106174' '27.753301' '49.787445' '51.592193' '36.187559']
[63  0 65 64  3]
[['Shredded Wheat' '100% Bran' 'Shredded Wheat spoon size'
  "Shredded Wheat 'n'Bran" 'All-Bran with Extra Fiber']]
```

In [ ]:

In [ ]: