homework_one

October 22, 2018

0.1 1.(a)

In [4]:
```
import numpy as np
import pandas as pd
import seaborn as sns

#
price = np.array([5.89, 49.59, 59.98, 159, 17.99, 56.99, 82.75, 142.19, 31, 125.5,
         4.5, 22, 52.9, 61, 33.5, 328, 128, 142.19, 229, 189.4])

weight = np.array([1.4, 1.5, 2.2, 2.7, 3.2, 3.9, 4.1, 4.1, 4.6, 4.8,
          4.9, 5.3, 5.5, 5.8, 6.2, 8.9, 11.6, 18, 22.9, 38.2])
#
quartile = np.array([25, 50, 75])



#
price_q1, price_median, price_q3 = np.percentile(price, quartile)
weight_q1, weight_median, weight_q3 = np.percentile(weight, quartile)

print("price Q1, MEDIAN, Q3 =", price_q1, price_median, price_q3)
print("weight Q1, MEDIAN, Q3 =", weight_q1, weight_median, weight_q3)
```

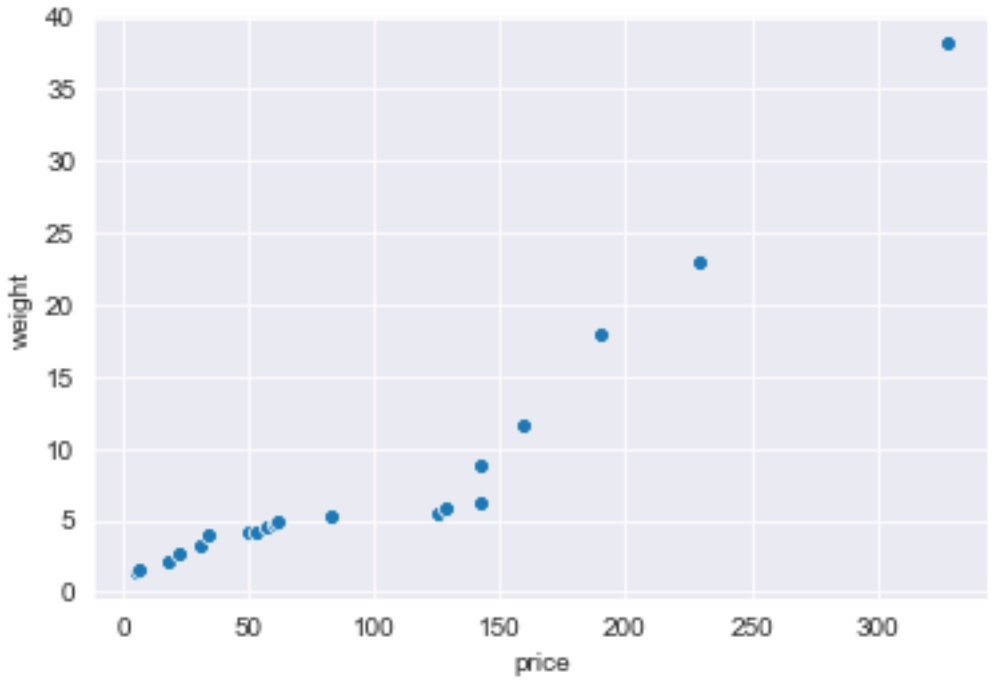
```
price Q1, MEDIAN, Q3 = 32.875 60.489999999999995 142.19
weight Q1, MEDIAN, Q3 = 3.7249999999999996 4.85 6.875
```

0.2 1.b)

In [5]:
```
#
data = np.column_stack((sorted(price), sorted(weight)))
df = pd.DataFrame(data, columns=["price", "weight"])

# Q-Q
sns.set_style("darkgrid")
sns.scatterplot(x="price", y="weight", data=df, sizes=80)
```

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x117c8a668>



0.3 1.(c)

```
In [6]: p_min, p_max = min(price), max(price)
        w_min, w_max = min(weight), max(weight)
        new_min, new_max = 1, 10

        price_normalized = [(n - p_min)/(p_max - p_min)*(new_max - new_min) + new_min for n in
        weight_normalized = [(n - w_min)/(w_max - w_min)*(new_max - new_min) + new_min for n i


        print("price", price_normalized)
        print()
        print("weight", weight_normalized)
```

```
price [1.0386707882534776, 2.2544358578052552, 2.5434930448222564, 5.298299845440495, 1.3753013

weight [1.0, 1.0244565217391304, 1.1956521739130435, 1.3179347826086958, 1.440217391304348, 1.6
```

0.4 1.(d)

```
In [7]: from scipy.stats import pearsonr

        print("pearson", pearsonr(price, weight)[0])
```

```
pearson 0.5363070272140884
```

```
In [8]: print("pearson", np.corrcoef(price, weight)[0, 1])

pearson 0.5363070272140884
```

0.5 2.(a)

x1, x4, x3, x5, x2;
x1, x4, x3, x5, x2;
x1, x4, x3, x5, x2;
x1, x3, x4, x2, x5;

```
In [9]: #

        x = np.array([1.4, 1.6])
        dataset = np.array([[1.5, 1.7],
                    [2, 1.9],
                    [1.6, 1.8],
                    [1.2, 1.5],
                    [1.5, 1.0]])


        #
        euc_dist = [np.linalg.norm(x-data) for data in dataset]
        print(":", euc_dist)

        #
        man_dist = [np.linalg.norm(x-data, ord=1) for data in dataset]
        print(":", man_dist)

        #
        sup_dist = [np.linalg.norm(x-data, ord=np.inf) for data in dataset]
        print(":", sup_dist)

        #
        cos_dist = [float(np.dot(x, data))/(np.linalg.norm(x)*np.linalg.norm(data)) for data in
        print(":", cos_dist)

: [0.14142135623730948, 0.6708203932499369, 0.28284271247461906, 0.22360679774997896, 0.608276
: [0.19999999999999996, 0.8999999999999999, 0.40000000000000013, 0.30000000000000004, 0.700000
: [0.10000000000000009, 0.6000000000000001, 0.20000000000000018, 0.19999999999999996, 0.600000
: [0.999991391443956, 0.9957522612528874, 0.9999694838187877, 0.9990282349375618, 0.965363393
```

0.6 2.(b)

x1, x3, x4, x2, x5

```
In [10]: from sklearn import preprocessing


         x_normalized = preprocessing.normalize([[1.4, 1.6]], norm="l2")
         dataset_normalized = preprocessing.normalize(dataset, norm="l2")

         # print(x_normalized)
         # print(dataset_normalized)

         #
         euc_dist = [np.linalg.norm(x_normalized[0]-data) for data in dataset_normalized]
         print(":", euc_dist)

: [0.004149350803200864, 0.09217091457843411, 0.007812321193114019, 0.044085486555962686, 0.263
```