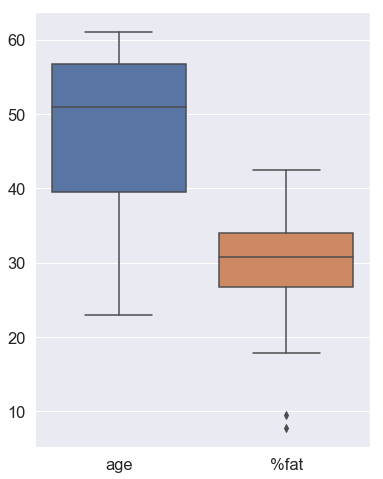
# 2.4 (a)

|  | **age** | **%fat** |
| --- | --- | --- |
| **Mean** | 46.444444 | 28.783333 |
| **Standard Deviation** | 13.218624 | 9.254395 |
| **Median** | 51.000000 | 30.700000 |

# 2.4(b)



# 2.4 (c)

A picture containing wall, sky

Description automatically generated

Fig: scatter plot

A picture containing wall

Description automatically generated

fig: qq-plot

# 2.8 (a)

Implemented 4 similarity algorithms. Compared with Cosine similarity from Scikit Learn and Scipy. There are some differences between 3 Cosine implementations. It might happen due to internal normalization and decimal precision.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A1 | A2 | Euclidean | Manhattan | Supremum | Cosine | Cosine-Scikit | Cosine-Scipy |
| x1 | 1.5 | 1.7 | 1 | 1 | 1 | 1 | 1 | 1 |
| x2 | 2.0 | 1.9 | 5 | 5 | 4 | 2 | 3 | 4 |
| x3 | 1.6 | 1.8 | 3 | 3 | 3 | 4 | 4 | 2 |
| x4 | 1.2 | 1.5 | 2 | 2 | 2 | 5 | 2 | 3 |
| x5 | 1.5 | 1.0 | 4 | 4 | 4 | 3 | 5 | 5 |

# 2.8 (b) – used L2 norm

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | A1 | A2 | A1norm | A2norm | Euclidean |
| x1 | 1.5 | 1.7 | 0.661622 | 0.749838 | 1 |
| x2 | 2.0 | 1.9 | 0.724999 | 0.688749 | 4 |
| x3 | 1.6 | 1.8 | 0.664364 | 0.747409 | 2 |
| x4 | 1.2 | 1.5 | 0.624695 | 0.780869 | 3 |
| x5 | 1.5 | 1.0 | 0.832050 | 0.554700 | 5 |

# 3.7 (a)

(35 – 13) / (70 – 13) = **0.38596491228070173**

# 3.7 (b)

**0.38926097658709724**

# 3.7 (c)

0.35

# 3.7 (d)

I would use min-max normalization if there is no additional data which can change min or max. This is because to use z-score or decimal methods, we need to save extra parameters for future data and for reconstruction of original data. But if the min-max cannot be guaranteed, I would use the decimal method here, because the distribution of this small sample may not reflect the actual distribution of data, and hence, statistics of the sample might be wrong.

# 3.9 (a)

[5, 10, 11, 13],

[15, 35, 50, 55],

[72, 92, 204, 215]

# 3.9 (b)

[5, 10, 11, 13, 15, 35, 50, 55. 72],

[92],

[204, 215]

# 3.9 (c)

Using KMeans clustering

[5, 10, 11, 13, 15, 35],

[50, 55, 72, 92],

[204, 215]