In [1]:

**Time Spent for Question 1: 3 hours**

import statistics

from scipy import stats

import matplotlib.pyplot as plt

import numpy as np

. . .

In [2]:

data=(11, 13, 15, 17, 19, 21, 21, 23, 23, 23, 23, 25, 27, 30, 33, 33, 33, 33, 36, 36, 38, 40, 46, 48, 54)

. . .

In [3]:

print(data, end=' ')

(11, 13, 15, 17, 19, 21, 21, 23, 23, 23, 23, 25, 27, 30, 33, 33, 33, 33, 36, 36, 38, 40, 46, 48, 54)

. . .

In [4]:

**#a**

**(i)Mean:**

statistics.mean(data)

Out[4]:

28.84

. . .

In [5]:

**(ii)Median:**

statistics.median(data)

Out[5]:

27

. . .

In [6]:

listData = list(data)

. . .

In [7]:

print(listData, end =' ')

[11, 13, 15, 17, 19, 21, 21, 23, 23, 23, 23, 25, 27, 30, 33, 33, 33, 33, 36, 36, 38, 40, 46, 48, 54]

. . .

In [8]:

. . .

In [9]:

**#b Mode:**

from collections import Counter

​

counter = Counter(listData)

max\_count = max(counter.values())

m = [k for k,v in counter.items() if v == max\_count]

print(m)

[23, 33]

. . .

In [10]:

print("bimodals:" ,m)

print("Comments : The given data is bimodal")

bimodals: [23, 33]

Comments : The given data is bimodal

. . .

. . .

In [12]:

max =max(listData)

. . .

In [13]:

print("Maximum :", max)

Maximum : 54

. . .

In [14]:

min = min(listData)

. . .

In [15]:

print("Minimum :", min)

Minimum : 11

. . .

In [25]:

**#c Midrange:**

midrange = (min+max)/2

. . .

In [24]:

print("Midrange :", midrange)

Midrange : 32.5

. . .

In [18]:

ndata = np.asarray(listData)

. . .

In [19]:

ndata

Out[19]:

array([11, 13, 15, 17, 19, 21, 21, 23, 23, 23, 23, 25, 27, 30, 33, 33, 33,

33, 36, 36, 38, 40, 46, 48, 54])

. . .

In [26]:

**#d The first quartile (Q1) and the third quartile (Q3) of the data?**

print("Q1 quantile of arr : ", np.quantile(ndata, .25))

print("Q3 quantile of arr : ", np.quantile(ndata, .75))

Q1 quantile of arr : 21.0

Q3 quantile of arr : 36.0

. . .

In [27]:

**#e the five-number summary of the data**

print("Minimum :", np.min(ndata))

print("Q1 quantile of data : ", np.quantile(ndata, .25))

print("Q2 quantile of data : ", np.quantile(ndata, .50))

print("Q3 quantile of data : ", np.quantile(ndata, .75))

print("Maximum :", np.max(ndata))

Minimum : 11

Q1 quantile of data : 21.0

Q2 quantile of data : 27.0

Q3 quantile of data : 36.0

Maximum : 54

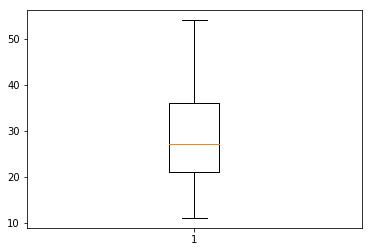
. . .

In [28]:

**#f Boxplot:**

plt.boxplot(ndata)

plt.show()



**Time spent for Question 2: 2:00 hours**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| age | %fat |  | age |  | %fat |  |
| 20 | 8.4 |  |  |  |  |  |
| 22 | 25.3 |  | **#a** |  |  |  |
| 25 | 7.6 |  | 45.77777778 | Mean | 28.25 |  |
| 25 | 18.8 |  | 50 | Median | 30 |  |
| 36 | 27.5 |  | 13.97837837 | Standard Deviation | 8.775486691 |  |
| 40 | 24.6 |  |  |  |  |  |
| 45 | 28.1 |  | **#b** |  |  |  |
| 48 | 28.8 |  | 20 | Minium | 7.6 |  |
| 49 | 30.2 |  | 37 | Quartile Q1 | 25.85 |  |
| 51 | 32.7 |  | 50 | Quartile Q2 | 30 |  |
| 53 | 40.2 |  | 57.75 | Quartile Q3 | 33 |  |
| 53 | 29.8 |  | 62 | Maximum | 40.2 |  |
| 57 | 32.3 | **#c** | | | | |
| 58 | 30.7 |
| 59 | 33.9 |
| 60 | 40.1 |
| 61 | 33.1 |
| 62 | 36.4 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **#d**   |  | | --- | |  | |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

**Time Spent for Question 3: 2.5 hours**

**#3**

import math

import numpy as np

from scipy.spatial import distance

# Example points in 3-dimensional space...

x = (15, 7, 24, 21)

y = (12, 0, 16, 10)

#distance = math.sqrt(sum([(a - b) \*\* 2 for a, b in zip(x, y)]))

#print("Euclidean distance from x to y (between two objects): ",distance)

​

p1 = np.array(x)

p2 = np.array(y)

. . .

In [6]:

#--- scalar product as similarity indicator

#dist1 = np.dot(p1,p2)

​

**#(a) Euclidean distance**

euclidean = np.linalg.norm(p1-p2)

print("Euclidean distance between two objects:", euclidean)

Euclidean distance between two objects: 15.588457268119896

. . .

In [9]:

**#(b) Manhattan distance**

manhattan = np.sum(np.abs(p1 - p2))

print("Manhattan distance between two objects:", manhattan)

Manhattan distance between two objects: 29

. . .

. . .

In [11]:

**#(c) Minkowski distance**

print("Minkowski distance between two objects using h value:",distance.minkowski([p1], [p2], 3))

Minkowski distance between two objects using h value: 13.031481884703163

. . .

In [12]:

**#(d) Supremum distance**

print("Supremum distance between two objects:",distance.minkowski(p1, p2, p=float('inf')))

Supremum distance between two objects: 11.0

. . .

In [13]:

**Time Spent for Question 4: 3.5 hours**

**# 4**# Example points in 2-dimensional space...

x = (1.3, 1.5)

x1 = (1.6, 1.8)

x2 = (2.1, 1.6)

x3 = (1.7, 1.2)

x4 = (1.2, 1.4)

x5 = (1.5, 1.3)

#distance = math.sqrt(sum([(a - b) \*\* 2 for a, b in zip(x, y)]))

#print("Euclidean distance from x to y (between two objects): ",distance)

p = np.array(x)

p1 = np.array(x1)

p2 = np.array(x2)

p3 = np.array(x3)

p4 = np.array(x4)

p5 = np.array(x5)

​

. . .

In [14]:

**#(a)**

**Euclidean distance**

print("Euclidean distance d1:", np.linalg.norm(p -p1))

print("Euclidean distance d2:", np.linalg.norm(p-p2))

print("Euclidean distance d3:", np.linalg.norm(p-p3))

print("Euclidean distance d4:", np.linalg.norm(p-p4))

print("Euclidean distance d5:", np.linalg.norm(p-p5))

​

Euclidean distance d1: 0.42426406871192857

Euclidean distance d2: 0.806225774829855

Euclidean distance d3: 0.49999999999999994

Euclidean distance d4: 0.14142135623730964

Euclidean distance d5: 0.28284271247461895

. . .

In [15]:

print("From high to low: d2>d3>d1>d5>d4")

**Rank from high to low: d2>d3>d1>d5>d4**

. . .

In [16]:

**Manhattan distance**

print("Manhattan distance d1:", np.sum(np.abs(p - p1)))

print("Manhattan distance d2:", np.sum(np.abs(p - p2)))

print("Manhattan distance d3:", np.sum(np.abs(p - p3)))

print("Manhattan distance d4:", np.sum(np.abs(p - p4)))

print("Manhattan distance d5:", np.sum(np.abs(p - p5)))

Manhattan distance d1: 0.6000000000000001

Manhattan distance d2: 0.9000000000000001

Manhattan distance d3: 0.7

Manhattan distance d4: 0.20000000000000018

Manhattan distance d5: 0.3999999999999999

. . .

In [17]:

print("From high to low: d2>d3>d1>d5>d4")

**Rank from high to low: d2>d3>d1>d5>d4**

. . .

In [18]:

**Supremum distance**

print("Supremum distance d1:", distance.minkowski(p, p1, p=float('inf')))

print("Supremum distance d2:", distance.minkowski(p, p2, p=float('inf')))

print("Supremum distance d3:", distance.minkowski(p, p3, p=float('inf')))

print("Supremum distance d4:", distance.minkowski(p, p4, p=float('inf')))

print("Supremum distance d5:", distance.minkowski(p, p5, p=float('inf')))

Supremum distance d1: 0.30000000000000004

Supremum distance d2: 0.8

Supremum distance d3: 0.3999999999999999

Supremum distance d4: 0.10000000000000009

Supremum distance d5: 0.19999999999999996

. . .

In [19]:

print("From high to low: d2>d3>d1>d5>d4")

**From high to low: d2>d3>d1>d5>d4**

. . .

In [20]:

**# Method 1 for cosine\_similarity**

from sklearn.metrics.pairwise import cosine\_similarity

print("distance d1(p, p1):",cosine\_similarity([p, p1]))

print("distance d2(p, p2):",cosine\_similarity([p, p2]))

print("distance d3(p, p3):",cosine\_similarity([p, p3]))

print("distance d4(p, p4):",cosine\_similarity([p, p4]))

print("distance d5(p, p5):",cosine\_similarity([p, p5]))

distance d1(p, p1): [[1. 0.99992123]

[0.99992123 1. ]]

distance d2(p, p2): [[1. 0.97893277]

[0.97893277 1. ]]

distance d3(p, p3): [[1. 0.9708504]

[0.9708504 1. ]]

distance d4(p, p4): [[1. 0.99998507]

[0.99998507 1. ]]

distance d5(p, p5): [[1. 0.98984772]

[0.98984772 1. ]]

. . .

In [21]:

**# Method 2 for cosine\_similarity**

def cos\_sim(a, b):

"""Takes 2 vectors a, b and returns the cosine similarity according

to the definition of the dot product

"""

dot\_product = np.dot(a, b)

norm\_a = np.linalg.norm(a)

norm\_b = np.linalg.norm(b)

return dot\_product / (norm\_a \* norm\_b)

print("distance d1(p, p1):",cos\_sim(p, p1))

print("distance d2(p, p2):",cos\_sim(p, p2))

print("distance d3(p, p3):",cos\_sim(p, p3))

print("distance d4(p, p4):",cos\_sim(p, p4))

print("distance d5(p, p5):",cos\_sim(p, p5))

distance d1(p, p1): 0.9999212291748408

distance d2(p, p2): 0.9789327661484941

distance d3(p, p3): 0.970850399211691

distance d4(p, p4): 0.9999850700587484

distance d5(p, p5): 0.9898477157360406

. . .

In [22]:

print("From high to low: d4>d1>d5>d2>d3" )

​

**Rank from high to low: d4>d1>d5>d2>d3**

**(b): Data Normalization**

For x =(1.3, 1.5), we are given A’1=, A’2 =

For x = (1.3, 1.5), we have =0.6594, =0.7557

Therefore, the normalized query is (0.6594, 0.7557). The normalized data set is given by the following table:

|  |  |  |
| --- | --- | --- |
|  | A’1 | A’2 |
| X’1 | 0.6644 | 0.7474 |
| X’2 | 0.7974 | 0.6060 |
| X’3 | 0.8170 | 0.5767 |
| X’4 | 0.6508 | 0.7593 |
| X’5 | 0.7557 | 0.6549 |

Recomputing the Euclidean distances as before yields:

|  |  |
| --- | --- |
| X1 | 0.0126 |
| X2 | 0.2053 |
| X3 | 0.2415 |
| X4 | 0.0055 |
| X5 | 0.1426 |

print("From high to low: x3>x2>x5>x1>x4" )

**Rank from high to low:** **x3>x2>x5>x1>x4**