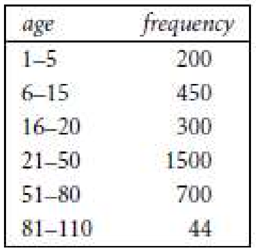
Data Mining Assignment 2
REPORT

Aradhya Mathur

1. In the textbook (3rd Edition):
   - 2.3, 2.6, 2.7, 2.8
   - 3.1, 3.3, 3.5, 3.7, 3.11
   - 3.13 [MUST be done in a programming language - pseudo code does NOT count]

   Due: 9/20 11:59pm (late penalty will be enforced, 10% per day, up to 50%)

2.3) Suppose that the values for a given set of data are grouped into intervals. The intervals and corresponding frequencies are as follows:



age	*frequency*
1–5	200
6–15	450
16–20	300
21–50	1500
51–80	700
81–110	44

Compute an *approximate median* value for the data.

Answer) Step 1: Finding cumulative frequency

Age	Frequency	Cf
1-5	200	200
6-15	450	650
16-20	300	950
21-50	1500	2450
51-80	700	3150
81-110	44	3194

Step 2: Finding median class

N = 3194 => N/2 = 1597

So median class is 21-50

width= 30 ; $(\sum \text{freq})_l$= 950 ; freq $_{\text{median}}$ = 1500 ; L_1 = 21

Step 3: Finding median

Median = $L_l + \frac{(\frac{N}{2} - (\sum freq)_1)}{freq_{median}}$width , Now substituting values

Median = 21 + ($\frac{1597-950}{1500}$)30 => Median = 21 + 647*30/1500

Median = 21 + 12.94 = > Median = 33.94

Median is 33.94

2.6) Given two objects represented by the tuples (22, 1, 42, 10) and (20, 0, 36, 8):

(a) Compute **the Euclidean distance** between the two objects.

Answer) Euclidean distance = $d_{(i,j)}$ = $\sqrt{\left(x_{i1}-x_{j1}\right)^2+\left(x_{i2}-x_{j2}\right)^2+..+\left(x_{ip}-x_{jp}\right)^2}$

x_{i1} = 22 ; x_{j1} = 20 ; x_{i2} = 1 ; x_{j2} = 0 ; x_{i3} = 42 ; x_{j3} = 36 ; x_{i4} = 10 ; x_{j4} = 8

$d_{(i,j)} = \sqrt{(22-20)^2+\ (1-0)^2+(42-36)^2+(10-8)^2}$

$d_{(i,j)} = \sqrt{4+1+36+4}$

$d_{(i,j)} = \sqrt{45}$

$d_{(i,j)}$ = **6.7082** is the Euclidean distance

(b) Compute **the Manhattan distance** between the two objects.

$d_{(i,j)} = |x_{i1}-x_{j1}| + \left|x_{i2}-x_{j2}\right| + \cdots ++ \mid x_{ip}-x_{jp}|$

$d_{(i,j)} = |x_{i1}-x_{j1}| + |x_{i2}-x_{j2}| + |x_{i3}-x_{j3}| +| x_{i4}-x_{j4}|$

$d_{(i,j)}$ = |22-20|+|1-0|+|42-36|+|10-8| => $d_{(i,j)}$ = |2|+|1|+|6|+|2|

$d_{(i,j)}$ = **11** is the Manhattan distance

(c) Compute **the Minkowski distance** between the two objects, using h= 3.

$d_{(i,j)} = h\sqrt{\left(|x_{i1}-x_{j1}|\right)^h+\left(|x_{i2}-x_{j2}|\right)^h+..+\left(|x_{ip}-x_{jp}|\right)^h}$

$d_{(i,j)} = 3\sqrt{\left(|x_{i1}-x_{j1}|\right)^3+\left(|x_{i2}-x_{j2}|\right)^3+..+\left(|x_{ip}-x_{jp}|\right)^3}$

$d_{(i,j)} = 3\sqrt{(22-20|)^3+(|1-0|)^3+(|42-36|)^3+(|10-8|)^3}$

$d_{(i,j)} = \sqrt[3]{8+1+216+8}$ => $d_{(i,j)} = \sqrt[3]{233}$

$d_{(i,j)}$ = **6.1534** is the Minkowski distance for h=3

(d) Compute **the supremum distance** between the two objects.

$d_{(i,j)} = \quad max_f^p |x_{if} - x_{jf}|$

$\max x_{if} = 42$; $\max x_{jf} = 36$

$d_{(i,j)} = \quad |42-36|$

$d_{(i,j)} =$ **6** is the supremum distance

2.7) The median is one of the most important holistic measures in data analysis. Propose several methods for median approximation. Analyze their respective complexity under different parameter settings and decide to what extent the real value can be approximated. Moreover, suggest a heuristic strategy to balance between accuracy and complexity and then apply it to all methods you have given.

In data analysis, median is a very important holistic measure. For high number of observations, median is quite expensive and approximation can be done easily using few ways. First and foremost, method to determine is using the formula Median = $L_1 + \frac{(\frac{N}{2} - (\sum \text{freq})_1)}{freq_{median}}$width. In this method observations have to be grouped in classes where each class will have a lower boundary L_1, N stands for total number of observations, $\text{freq}_{\text{median}}$ refers to frequency of median class. Cumulative frequency is determined and median class is the class in which N/2 lies. $(\sum \text{freq})_1$ is the cumulative frequency till the median class (median class not involved). Width is the class range.

Number of classes/intervals$\propto$ Accuracy $\propto$Time Required $\propto$ Cost

Using iterations, we can find the median quite precisely. Initially dividing the observations into let's say x classes, next step is to find median class. After getting the median class we repeat the process by dividing median class into x sub classes. By using this method, we can locate median more accurately, but the time and cost required would be very high.

Median can also be find using histograms.

To find the ideal class, accuracy/error percentage and time required are the most important parameters. Also, cost is parameter that needs to be considered while finding median for large number of observations. Now trials can be run based on time and error parameters to find the ideal number of classes from range of options each with different number of classes. There isn't a fix formula to get the ideal number of classes, using comparisons and iterations, one can find the best-case scenario.

2.8) It is important to define or select similarity measures in data analysis. However, there is no commonly accepted subjective similarity measure. Results can vary depending on the similarity measures used. Nonetheless, seemingly different similarity measures may be equivalent after some transformation.

Suppose we have the following 2-D data set:

	A_1	A_2
x_1	1.5	1.7
x_2	2	1.9
x_3	1.6	1.8
x_4	1.2	1.5
x_5	1.5	1.0

(a) Consider the data as 2-D data points. Given a new data point, x = 1.4, 1.6/ as a query, rank the database points based on similarity with the query using Euclidean distance, Manhattan distance, supremum distance, and cosine similarity.

Euclidean distance = $d_{(i,j)} = \sqrt{(x_{i1}-x_{j1})^2+(x_{i2}-x_{j2})^2+..+(x_{ip}-x_{jp})^2}$

For X1: $d_{(i,j)} = \sqrt{(x_{i1}-x_{j1})^2+(x_{i2}-x_{j2})^2+..+(x_{ip}-x_{jp})^2}$

$d_{(i,j)} = \sqrt{(1.5-1.4)^2+(1.7-1.6)^2}$ => $d_{(i,j)} = \sqrt{(0.1)^2+(0.1)^2}$

$d_{(i,j)} = \sqrt{0.02}$ => $d_{(i,j)} = 0.1414$

Manhattan distance = $d_{(i,j)} = |x_{i1}-x_{j1}| + |x_{i2}-x_{j2}| + \cdots ++ | x_{ip}-x_{jp}|$

For X1: $d_{(i,j)} = |x_{i1}-x_{j1}| + |x_{i2}-x_{j2}|$

$d_{(i,j)} = |1.5-1.4| + |1.7-1.6|$ => $d_{(i,j)} = 0.2$

Supremum distance = $d_{(i,j)} = max_f^p|x_{if}-x_{jf}|$

Cosine similarity = $sim(x,y) = \frac{x^t.y}{||x||||y||}$

For x1 = $\frac{x^t.y}{||x||||y||}$

x.y = 1.5x1.4 +1.6x1.7 = 4.82

$||x|| = \sqrt[2]{1.5^2+1.7^2} = \sqrt[2]{5.14} = 2.2672$; $||y|| = \sqrt[2]{1.4^2+1.6^2} = \sqrt[2]{4.52} = 2.1260$

Sim = 4.82/(2.2672 x 2.1260) = 0.9999860

Similarly, for x2, x3, x4 and x5

	Euclidean	Manhattan	Supremum	Cosine
X1	0.1414	0.2	0.1	0.9999860
X2	0.6708	0.9	0.6	0.9957522
X3	0.2828	0.4	0.2	0.9999694
X4	0.2236	0.3	0.2	0.9990282
X5	0.6083	0.7	0.6	0.9653633

Rank

Euclidean: X1, X4, X3, X5, X2

Manhattan: X1, X4, X3, X5, X2

Supremum X1, X3, X4, X2, X5 (in this X3=X4 and X5=X2)

Cosine: X1, X3, X4, X2, X5

(b) Normalize the data set to make the normof each data point equal to 1. Use Euclidean distance on the transformed data to rank the data points.

For (x,y)

a = $\sqrt[2]{x^2 + y^2}$

Norm(x,y) = ($\frac{x}{a}$, $\frac{y}{a}$)

For (1.4,1.6)

= $\sqrt[2]{1.4^2 + 1.6^2}$ = $\sqrt[2]{4.52}$ = 2.126

Normalized (1.4,1.6) = ($\frac{1.4}{2.126}$, $\frac{1.6}{2.126}$) = (0.6585,0.7526)

Similarly, for x1, x2, x3, x4 and x5

Euclidean distance = $d_{(i,j)}$ = $\sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + .. + (x_{ip} - x_{jp})^2}$

	A1	A2	Norm A1	Norm A2	Euclidean
X1	1.5	1.7	0.6616	0.7498	0.004177
X2	2	1.9	0.7250	0.6887	0.092225
X3	1.6	1.8	0.6644	0.7474	0.007864
X4	1.2	1.5	0.6247	0.7809	0.044083
X5	1.5	1.0	0.8320	0.5547	0.263186

Ranking data points: X1, X3, X4, X2 and X5

- **3.1, 3.3, 3.5, 3.7, 3.11**

3.1) *Data quality* can be assessed in terms of several issues, including accuracy, completeness, and consistency. For each of the above three issues, discuss how data quality assessment can depend on the *intended use* of the data, giving examples. Propose two other dimensions of data quality.

1) Accuracy: Amazon might only require your zip code to know the availability of a certain product in that area but the delivery manager would require accurate address to deliver the product to deliver the product accurately. Inaccurate address might result in late delivery or no delivery.
2) Completeness: While writing the address in the delivery option, a person forgets to mention street name and landmark. Missing data is a serious problem faced by delivery manager as he has to deliver the product based only on house number and zip code. With the availability of rest of the data, task would have been simplified.
3) Consistency: Suppose the street number entered in the address does not lie in the zip code entered in the database for amazon while it lies in database of US postal service. This means that street number has a consistency issue.

Data quality has other dimensions too:

1) Believability: Trustworthiness of data is believability. Phone numbers must have ten digits. Any entry in which digits are either less than or more than 10 is of no use. So, data values within possible range are required.
2) Interpretability: Ease of understanding the data is interpretability. For example: A movie database has given information about limitations of the data such as not all information is available or true.

3.3) Exercise 2.2 gave the following data (in increasing order) for the attribute age:13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70.

(a) Use smoothing by bin means to smooth these data, using a bin depth of 3. Illustrate your steps. Comment on the effect of this technique for the given data.

Step 1) Sorting data: 13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70

Step 2) Dividing into bins of depth = 3

Bin 1: 13, 15, 16

Bin 2: 16, 19, 20

Bin 3: 20, 21, 22

Bin 4: 22, 25, 25

Bin 5: 25, 25, 30

Bin 6: 33, 33, 35

Bin 7: 35, 35, 35

Bin 8: 36, 40, 45

Bin 9: 46, 52, 70

Step 3) Calculating mean of each bin

Bin 1: 13, 15, 16 = 14.67

Bin 2: 16, 19, 20 = 18.34

Bin 3: 20, 21, 22 = 21

Bin 4: 22, 25, 25 = 24

Bin 5: 25, 25, 30 = 26.67

Bin 6: 33, 33, 35 = 33.67

Bin 7: 35, 35, 35 = 35

Bin 8: 36, 40, 45 = 40.34

Bin 9: 46, 52, 70 = 56

Step 4) Substitute mean value in place of values in the bin

Bin 1: 14.67, 14.67, 14.67

Bin 2: 18.34, 18.34, 18.34

Bin 3: 21, 21, 21

Bin 4: 24, 24, 24

Bin 5: 26.67, 26.67, 26.67

Bin 6: 33.67, 33.67, 33.67

Bin 7: 35, 35, 35

Bin 8: 40.34, 40.34, 40.34

Bin 9: 56, 56, 56

Smoothing of sorted data can be done by binning methods by checking the values around it. These sorted values are stored in bins. Local smoothing is performed.

(b) How might you determine outliers in the data?

Using clustering. Clustering is a process in which data are collected and organised into clusters along with similar kind of data. If a data doesn't belong to any cluster, it is an outlier.

(c) What other methods are there for data smoothing?

Data smoothing can be done via binning and regression. Apart from bin means, smoothing can be achieved by bin medians, bin boundaries, equal width bins in which class range remains same in each bin. Linear and multiple regression can be used for smoothing. Regression fits data values to function. Apart from these concept hierarchy is also used for smoothing. Smoothing can also be achieved through classification methods like neural network.

3.5) What are the value ranges of the following normalization methods?

(a) min-max normalization: In this normalization original data is linearly transformed. There is not a defined range, it can range between any two values.

(b) z-score normalization: In z-score normalization new values are found using mean and standard deviation = $\frac{v_i - mean_a}{std_a}$

So, range will be [$\frac{min_a - me\quad a}{std_a}$, $\frac{max_a - me\quad a}{std_a}$]

(c) z-score normalization using the mean absolute deviation instead of standard deviation: This is more robust than the earlier normalization

Mean absolute deviation = s_a = $\frac{1}{n}$($|v_1 - mean_a|$ +$|v_2 - mean_a|$+…..+$|v_n - mean_a|$)

Range will be [$\frac{min_a - mean_a}{s_a}$, $\frac{max_a - mea\quad a}{s_a}$]

(d) normalization by decimal scaling: Moving decimal places to normalize

$v_i{}' = \frac{v_i}{10^j}$, here j is smallest integer which satisfies max($|v_i{}'|$) <1

Range: [$\frac{min_a}{10^j}$, $\frac{max_a}{10^j}$]

3.7) Using the data for age given in Exercise 3.3, answer the following:

(a) Use min-max normalization to transform the value 35 for age onto the range

[0.0, 1.0].

Given:

min_a= 13 and max_a=70

$nmin_a$= 0 and $nmax_a$=1

v = 35

v' = $\frac{(vi - min_a)}{max_a - min_a}$($nmax_a$-$nmin_a$) + $nmin_a$

v' = $\frac{(35-1\ \ )}{70-13}$ (1-0) + 0 => $\frac{(22)}{57}$

v' = 0.385965

(b) Use z-score normalization to transform the value 35 for age, where the standard deviation of age is 12.94 years.

Here v'= $\frac{v_i - mean_a}{std_a}$

Given: v_i = 35, std_a=12.94, $mean_a$= (13+15...+52+70)/ 27 = 809/27 =29.962963

So v' = $\frac{v_i - mean_a}{std_a} = \frac{35-29.962963}{12.94}$ = 0.389261

(c) Use normalization by decimal scaling to transform the value 35 for age.

$v_i' = \frac{v_i}{10^j}$, here j=2

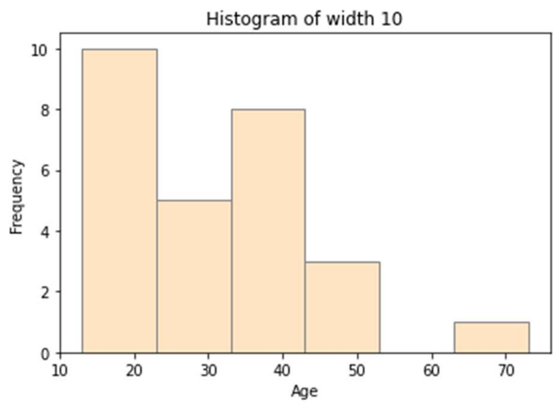
So, v'=35/100=0.35

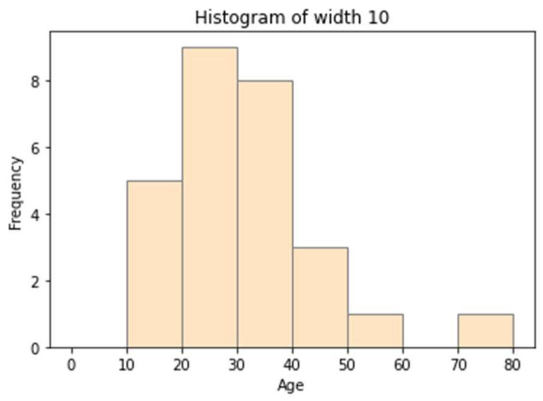
(d) Comment on which method you would prefer to use for the given data, giving reasons as to why.

Min-max method may give out-of-bounds error if a future input case for normalization lies outside of the initial range. In z-score normalization, standard deviation is susceptible to outliers. Considering this and ease of understanding, normalization by decimal scaling is a better method. Decimal scaling also maintains data distribution.

3.11) Using the data for age given in Exercise 3.3,

(a) Plot an equal-width histogram of width 10.





(b) Sketch examples of each of the following sampling techniques: SRSWOR, SRSWR, cluster sampling, and stratified sampling. Use samples of size 5 and the strata "youth," "middle-aged," and "senior."

Ans) 13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70. Colour coding: Youth, middle-aged, senior

Sample Size = 5 for all the sampling techniques

SRSWOR: {13,20,22,33,35} (Each data can be chosen once, no repetition allowed)

SRSWR: {20,20,22,33,33} (Repetition allowed)

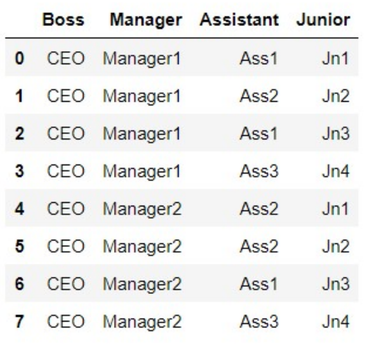
Cluster Sampling: {22,25,30,33,35} (Data in a cluster)

Stratified Sampling: {15,20,25,35,70} where {15,20} are youth, {25,35} are middle aged and {70} is senior.

{13, 15, 16, 16, 19, 20, 20}- Youth, {21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40}- Middle Aged, {45, 46, 52, 70}- Senior

3.13) Propose an algorithm, in pseudocode or in your favourite programming language, for the

following:

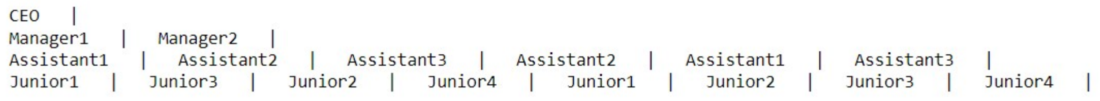


	Boss	Manager	Assistant	Junior
0	CEO	Manager1	Ass1	Jn1
1	CEO	Manager1	Ass2	Jn2
2	CEO	Manager1	Ass1	Jn3
3	CEO	Manager1	Ass3	Jn4
4	CEO	Manager2	Ass2	Jn1
5	CEO	Manager2	Ass2	Jn2
6	CEO	Manager2	Ass1	Jn3
7	CEO	Manager2	Ass3	Jn4

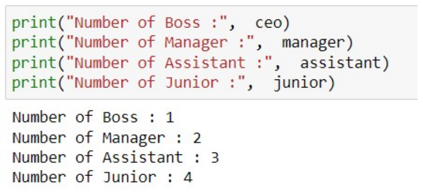
(a) The automatic generation of a concept hierarchy for nominal data based on the number of distinct values of attributes in the given schema.

```
#Using Tree

from collections import deque as queue
class Title:
  def __init__(i, j):

    i.data = j
    i.left = None
    i.right = None
    i.center = None

def hier(company):

  if (company == None):
    return
  queue_ = queue()
  queue_.append(company)
  queue_.append(None)

  while (len(queue_) > 1):
    var = queue_.popleft()
    if (var == None):
      queue_.append(None)
      print()

    else:
      if (var.left):
        queue_.append(var.left)

      if (var.center):
        queue_.append(var.center)
      if (var.right):
        queue_.append(var.right)
      print(var.data, end = "  |  ")

if __name__ == '__main__':
  company = Title("CEO")
  company.left = Title("Manager1")
  company.right = Title("Manager2")
  company.left.left = Title("Assistant1")
  company.left.center = Title("Assistant2")
  company.left.right = Title("Assistant3")
  company.right.right = Title("Assistant3")
  company.right.center = Title("Assistant1")
  company.right.left = Title("Assistant2")
  company.left.left.left = Title("Junior1")
  company.left.left.right = Title("Junior3")
  company.left.center.right = Title("Junior2")
  company.left.right.right = Title("Junior4")
  company.right.left.left = Title("Junior1")
  company.right.left.right = Title("Junior2")
  company.right.center.right = Title("Junior3")
  company.right.right.right = Title("Junior4")

  hier(company)
```

Concept Hierarchy



```
CEO   |
Manager1   |   Manager2   |
Assistant1   |   Assistant2   |   Assistant3   |   Assistant2   |   Assistant1   |   Assistant3   |
Junior1   |   Junior3   |   Junior2   |   Junior4   |   Junior1   |   Junior2   |   Junior3   |   Junior4   |
```



```
print("Number of Boss :",  ceo)
print("Number of Manager :",  manager)
print("Number of Assistant :",  assistant)
print("Number of Junior :",  junior)
```

```
Number of Boss : 1
Number of Manager : 2
Number of Assistant : 3
Number of Junior : 4
```

(b) The automatic generation of a concept hierarchy for numeric data based on the equal-width partitioning rule.

In this concept width of each bin is same.

Divides the input data into X intervals of same size

Let Min and Max be the minimum and maximum values of the input data,

Then width will be: width = (Max –Min)/X

For example: Input data = 2, 6, 13, 18, 20, 25, 27, 28, 34, 42

X=4, Max = 42, Min =2

Width = (Max-Min)/X = 40/4 = 10

Bin1= 2-12, Bin2=13-22 Bin3= 23-32 Bin4= 33-42

Bin1= {2,6}, Bin2= {13,18,20}, Bin3= {25,27,28}, Bin4= {34,42}

Here, width of each bin is equal

```
dframe = pd.dataframe(2, 6, 13, 18, 20, 25, 27, 28, 34, 42)
def hier(dframe):
   list = {}
bins = 4
min = min(dataset)
max = max(dataset)
width=(max-min)/bins;
```

c) The automatic generation of a concept hierarchy for numeric data based on the equal-frequency partitioning rule.

In this concept bins are formed in a way that frequency of each bin is same.

Divide the input data into X intervals, each having almost same number of input data.

For example: Input data = 2, 6, 13, 18, 20, 25, 27, 28, 34, 42, 46, 50

Bin1= {2,6,13}, Bin2= {18,20,25}, Bin3= {27,28,34}, Bin4= {42,46,50}

Here, frequency (depth) of each bin is equal.

```
dframe = pd.dataframe(2, 6, 13, 18, 20, 25, 27, 28, 34, 42, 46, 50)
def hier(dframe):
   list = {}
depth=3
bins = (dframe.count()) / depth
```