# Datamining Homework: Clustering

นาย ดรันภพ เป็งคำตา

580610642

**รูปแบบการทดลอง**

เป็นการทดลองการ การทำงานของ clustering algorithm กับ dataset ที่กำหนดไว้เพื่อเป็นการทดลองเพื่อหาว่า การทำ clustering นั้น เหมาะสมกับ dataset ดังกล่าว หรือไม่ และทำการทดสอบด้วยค่า purity ซ่งเป็นค่าบอกถึงความ บริสุทธิ์โดยรวมของคลัสเตอร์

ในการทดลองนี้ จะใช้ algorithm K-mean clustering โดยจะทดลองของเปลี่ยนค่า K เรื่อย ๆ จาก 2 ไปจนถึง 20 เพื่อตรวจสอบว่า ค่า k และ purity จะสัมพันธ์กันมากน้อยเพียงใด

**รูปแบบของดาต้าเซทที่นำมาทดสอบ**

เป็นดาต้าเซทชื่อ **Student performance dataset** เป็นดาต้าเซทของข้อมูลต่าง ๆ ของนักศึกษา เช่น โรงเรียน, อาชีพของบิดา มารดา, เพศ, อายุ และอื่น ๆ เป็นจำนวน 33 attributes โดยเป้าหมายของดาต้าเซทนี้คือ การทำการประเมินผลของคะแนนสอบภาคเรียนสุดท้ายของนักศึกษาแต่ละคน โดยข้อมูลของดาต้าเซทนี้ ประกอบด้วย ข้อมูลลักษณะต่าง ๆดังนี้ เป็นattributeที่เป็นไปได้ 2 ค่า (binary), เป็นจานวนเต็มในช่วง ๆ หนึ่ง (numeric) และ เป็น ข้อความต่าง ๆ กันจำนวนหนึ่ง (nominal)

**การ Preprocess ข้อมูล**

เนื่องจากข้อมูลที่ได้มา พบว่าเป็นข้อมูลที่ไม่มี missing value จึงไม่จำเป็นที่จะต้องจัดการกับปัญหานี้ แต่จะมีการทำ normalize ข้อมูลทั้งหมดให้อยู่อยู่ในช่วง [0, 1] อย่างเท่าเทียมกัน ยกเว้นเว้น attribute ‘G3’ เนื่องจากเป็น class label ซึ่งจะไม่ได้ใช้ในการทำ clustering ครั้งนี้ และ เนื่องจากพบว่า attribute ทั้งหมด มีข้อมูลอยู่ สามแบบ จังได้ออกแบบ function สาหรับการ normalize ไว้เพื่อความสะดวกในการใช้งาน

* Function สำหรับ normalize ข้อมูล

def normalize(input):

input['school'] = input['school'].apply(lambda x:binary('GP', x))

input['sex'] = input['sex'].apply(lambda x:binary('F', x))

input['age'] = input['age'].apply(lambda x:scaling(15, 22, x))

input['address'] = input['address'].apply(lambda x:binary('U', x))

input['famsize'] = input['famsize'].apply(lambda x:binary('LE3', x))

input['Pstatus'] = input['Pstatus'].apply(lambda x:binary('T', x))

input['Medu'] = input['Medu'].apply(lambda x:scaling(0, 4, x))

input['Fedu'] = input['Fedu'].apply(lambda x:scaling(0, 4, x))

input['Mjob'] = input['Mjob'].apply(lambda x:nominal(['teacher', 'health', 'services', 'at\_home', 'other'], x))

input['Fjob'] = input['Fjob'].apply(lambda x:nominal(['teacher', 'health', 'services', 'at\_home', 'other'], x))

input['reason'] = input['reason'].apply(lambda x:nominal(['home', 'reputation', 'course', 'other'], x))

input['guardian'] = input['guardian'].apply(lambda x:nominal(['mother', 'father', 'other'], x))

input['traveltime'] = input['traveltime'].apply(lambda x:scaling(1, 4, x))

input['studytime'] = input['studytime'].apply(lambda x:scaling(1, 4, x))

input['failures'] = abs(input['failures'].apply(lambda x:scaling(1, 4, x)))

input['schoolsup'] = input['schoolsup'].apply(lambda x:binary('yes', x))

input['famsup'] = input['famsup'].apply(lambda x:binary('yes', x))

input['paid'] = input['paid'].apply(lambda x:binary('yes', x))

input['activities'] = input['activities'].apply(lambda x:binary('yes', x))

input['nursery'] = input['nursery'].apply(lambda x:binary('yes', x))

input['higher'] = input['higher'].apply(lambda x:binary('yes', x))

input['internet'] = input['internet'].apply(lambda x:binary('yes', x))

input['romantic'] = input['romantic'].apply(lambda x:binary('yes', x))

input['famrel'] = input['famrel'].apply(lambda x:scaling(1, 5, x))

input['freetime'] = input['freetime'].apply(lambda x:scaling(1, 5, x))

input['goout'] = input['goout'].apply(lambda x:scaling(1, 5, x))

input['Dalc'] = input['Dalc'].apply(lambda x:scaling(1, 5, x))

input['Walc'] = input['Walc'].apply(lambda x:scaling(1, 5, x))

input['health'] = input['health'].apply(lambda x:scaling(1, 5, x))

input['absences'] = input['absences'].apply(lambda x:scaling(0, 93, x))

input['G1'] = input['G1'].apply(lambda x:scaling(0, 20, x))

input['G2'] = input['G2'].apply(lambda x:scaling(0, 20, x))

* Function สำหรับ normalize ข้อมูลที่เป็นช่วงค่า (numeric)

def scaling(min, max, x):

return (x - min)/(max - min)

* Function สำหรับ normalize ข้อมูลที่เป็นสองค่า (binary)

def binary(a, x):

return 1 if x == a else 0.5

* Function สำหรับ normalize ข้อมูลที่เป็นชุดของข้อความ (nominal)

def nominal(p\_list, x):

for n, i in enumerate(p\_list):

if i == x:

return scaling(0, len(p\_list)-1, n)

เนื่องจากในกระบวนการของ K-mean cluster จะมีการใช้ geometric mean ซึ่งเป็นค่ารูทของผลคูณ ต่าง ๆ ซึ่ง หากค่าใดค่าหนึ่งมีค่า เป็น **0** จะทำให้การทำงานในส่วนนี้ผิดเพี้ยนไป ดังนั้นในการทำการ normalize จะพยายามลีกเลี่ยงให้เกิดค่า 0 ขึ้น

# จำนวนรอบของการทดลอง

จะทำการทดสอบการทำงานของ K-mean เพื่อหาค่า purity ข้อแต่ละค่า k โดยจะเริ่มจาก ค่า 1 ไปจนถึง 20 เนื่องจากผลจากดาต้าเซทนี้มีคลาสคำตอบทั้งหมด 21 คลาส คือ 0 – 20 เพื่อตรวจสอบว่า หากทำการแบ่งคลัสเตอร์ เป็นจำนวนเท่ากับคลาส จะได้ผลอย่างไร

เพื่อความแม่นยำที่มากขึ้น ในการหาค่า purity ของแต่ละ k จะมีการทำซ้ำในแต่ละ k 3 รอบ และนำมาหาค่าเฉลี่ยเพื่อเป็นคำตอบของแต่ละค่า k

# โค้ดส่วนการทำงานหลัก

import input

import k\_mean as km

import numpy as np

if \_\_name\_\_ == '\_\_main\_\_':

data = input.loadFile() # dataset loading

input.normalize(data) # normalize dataset

for i in range(2,21): # Loop k from 2 - 20

k = i

avg\_r = 3 # set repeating step

sum\_p = 0

for j in range(1, avg\_r+1): # loop for avr\_r round

final\_centroidz, final\_labels = km.kmeans(data.values, k)

# get the final centroid points -

# and final set of clusterd data

p = km.getPurity(final\_labels)

# calculate the purity

sum\_p += p

print('%sK = %2d (%d) Purity %f'% (''.ljust(5), i, j, p))

print('K = %2d Average Purity %f\n'% (i, sum\_p/avg\_r))

# show final purity for each k