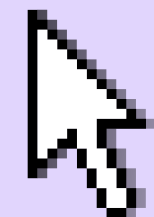


RECOMMENDATION SYSTEM

Group 2



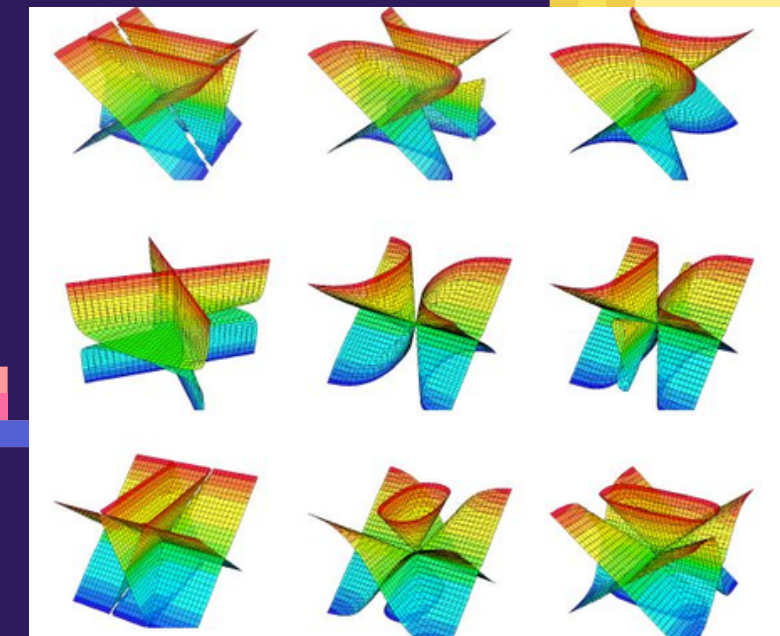
ALGORITHM

นำข้อมูลมา plot เป็น hyperplane

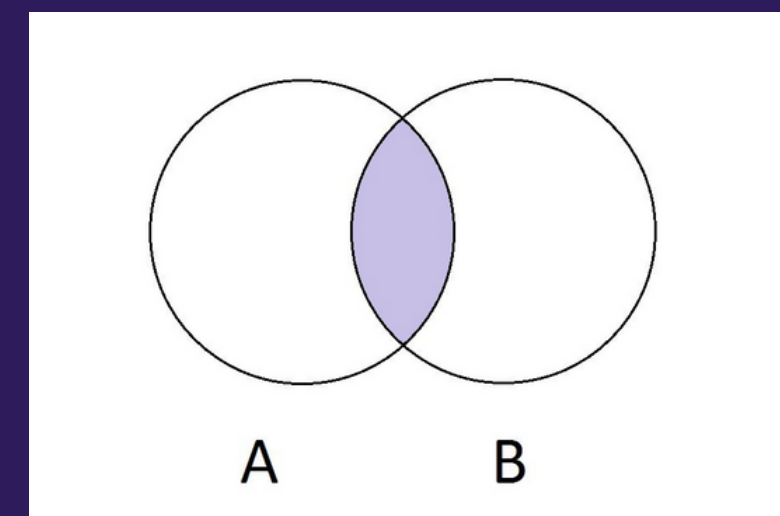
- แกนคือ user_id
- จุดบนกราฟคือ name (ชื่อ anime)

จากนั้นนำเรื่องที่มีระยะห่างระหว่างเรื่องที่ input
ที่น้อยที่สุด 10 เรื่อง มาเป็น output

ถ้า input เข้ามาหลายเรื่อง จะนำระยะห่างระหว่าง
เรื่องที่ input แต่ละเรื่อง มา intercept กัน แล้ว
output เป็น 10 เรื่องที่ระยะห่างน้อยที่สุด

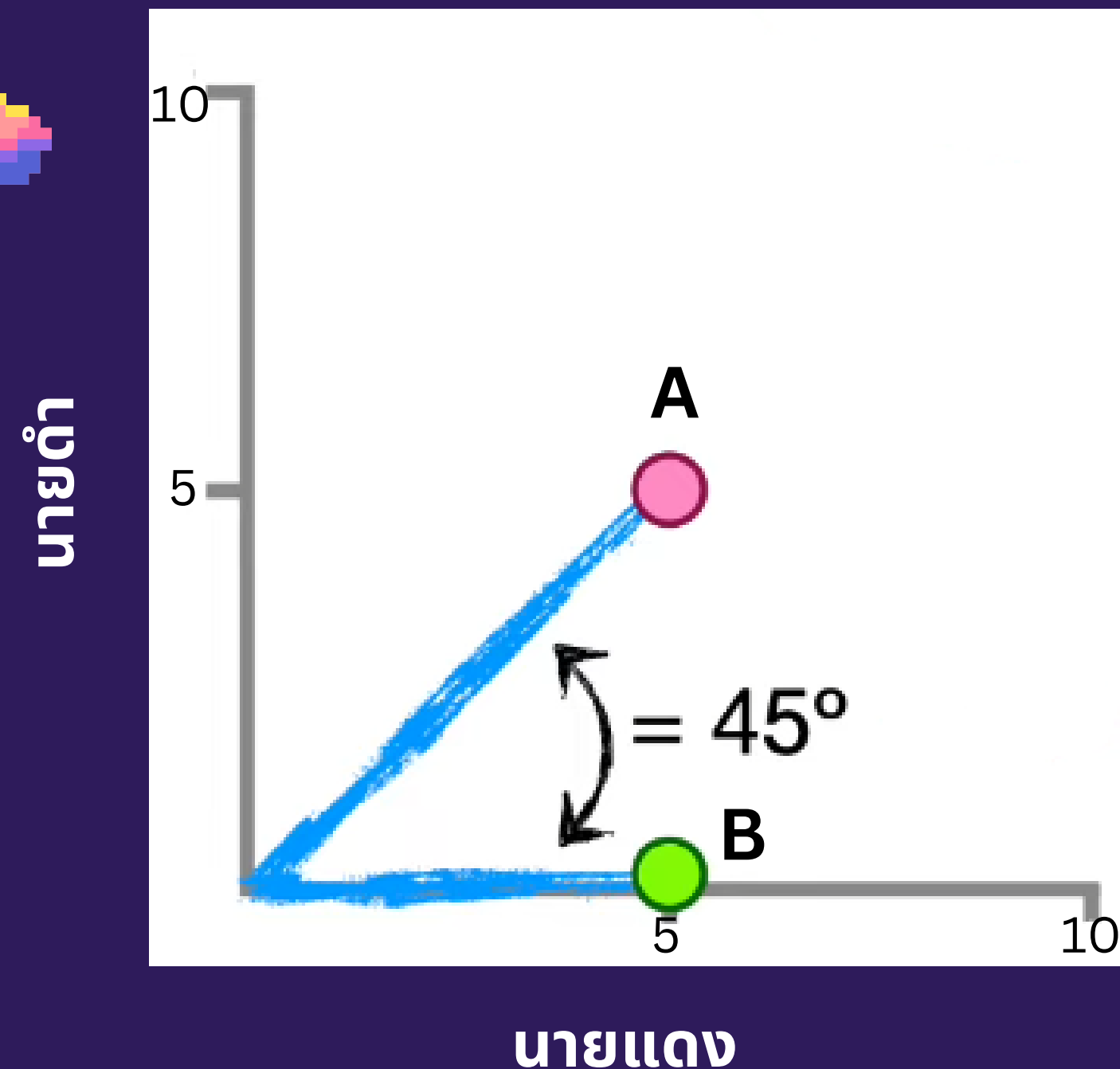


Hyperplane



Interception

COSINE SIMILARITY



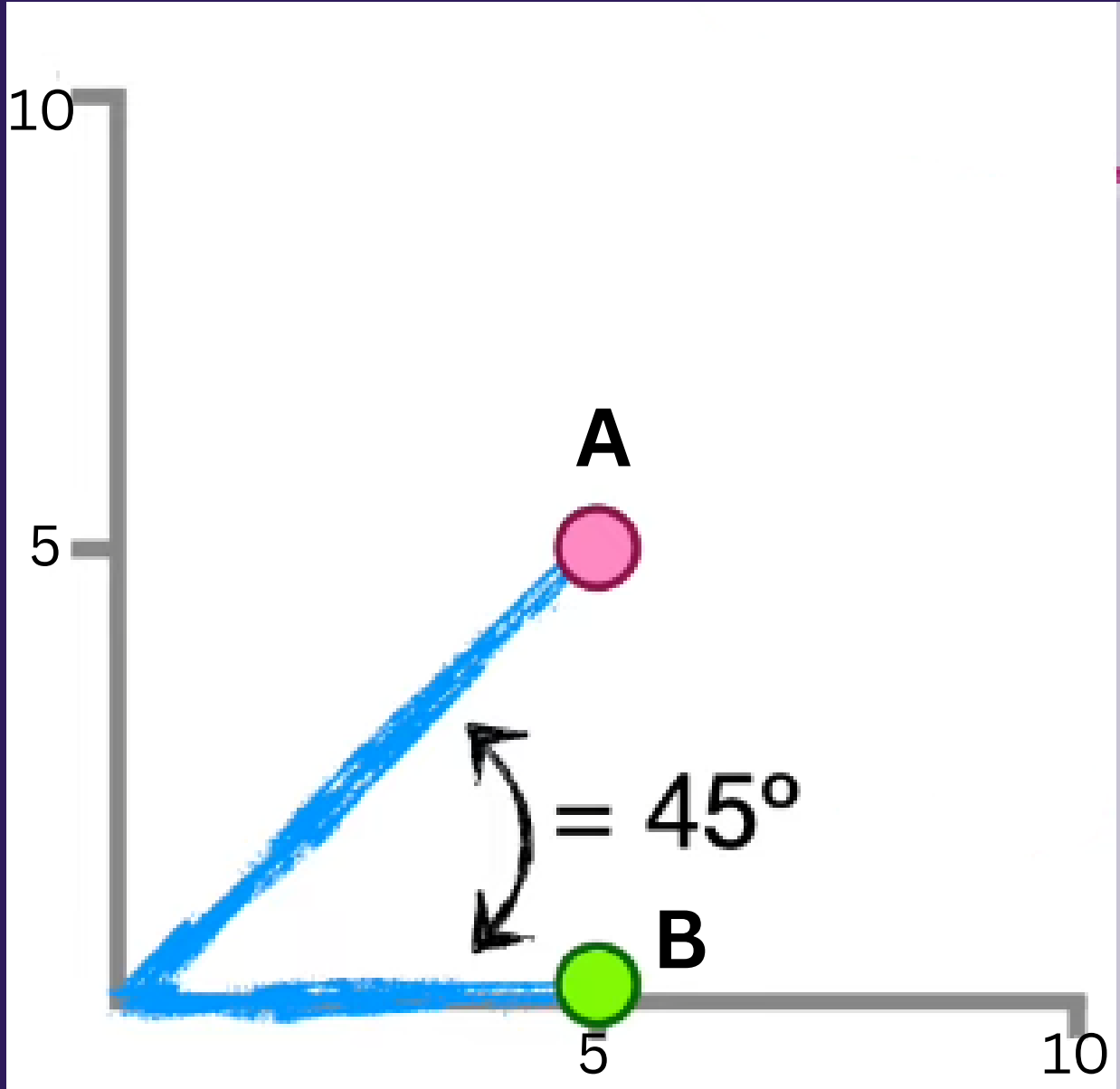
	แนวตั้ง	แนวนอน
A	5	5
B	0	5

Cosine Similarity :
 $\cos(45^\circ) = 0.71$

Cosine Distance :
 $1 - \cos(45^\circ) = 0.29$

COSINE SIMILARITY

ນ້ອມ



ນ້ອມ

	ນ້ອມ	ນ້ອມ
A	5	5
B	0	5

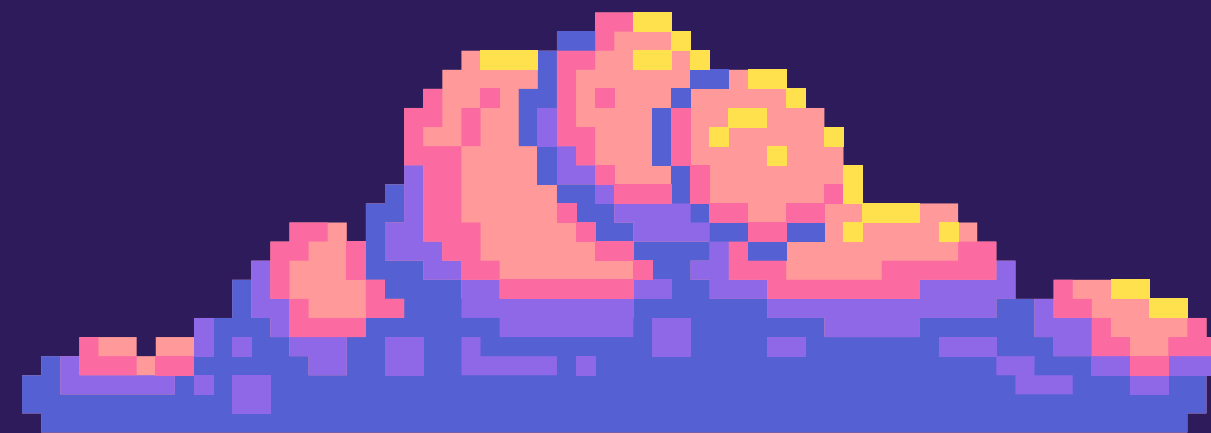
$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$



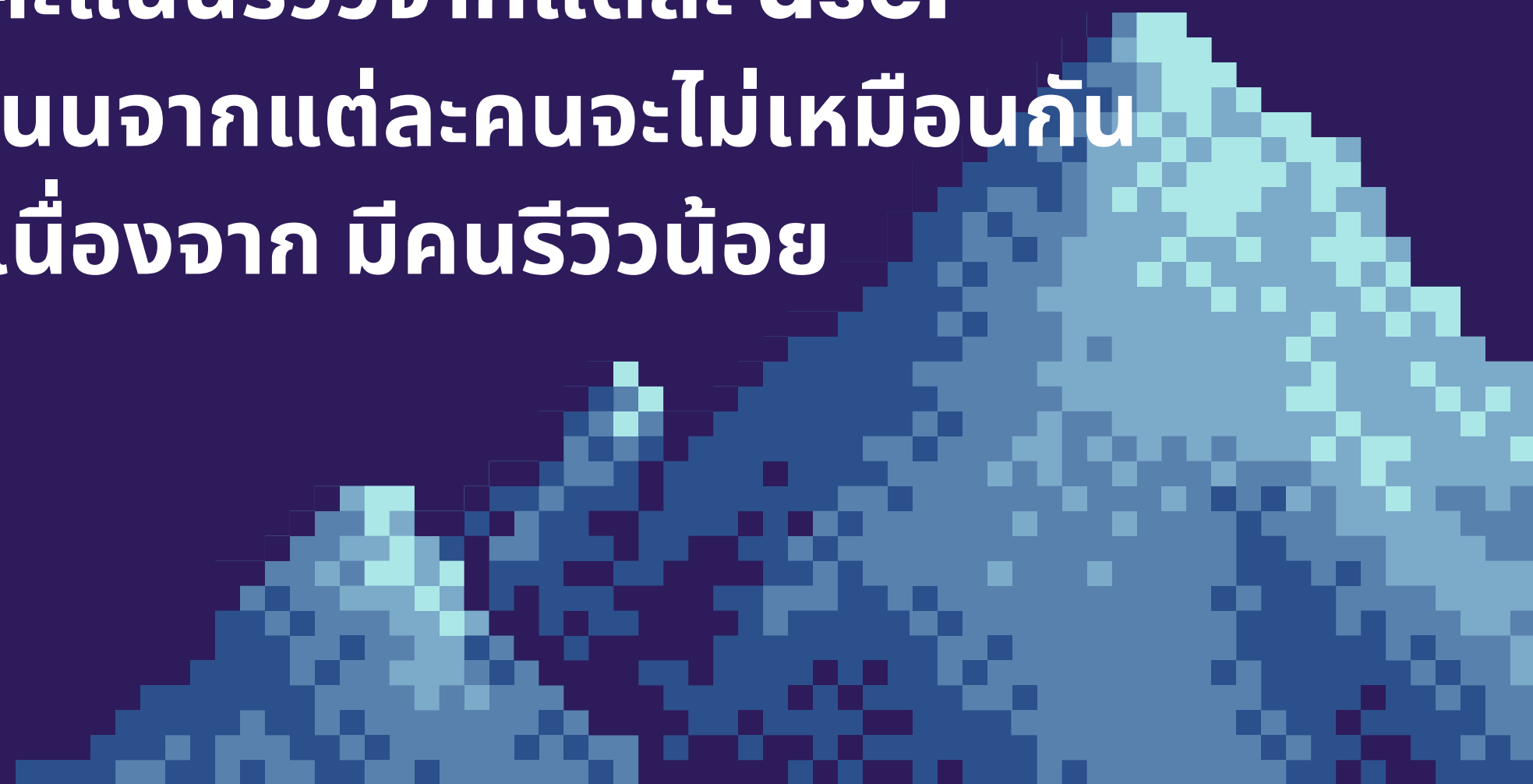
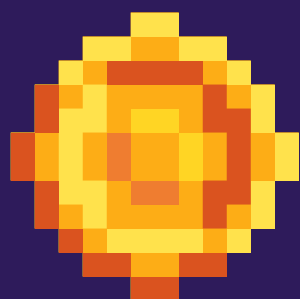
PRO

- 1.คุณภาพของเรื่องที่ใส่ไป กับ เรื่องที่แนะนำจะใกล้เคียงกัน
- 2.กลุ่มคนดูเป็นกลุ่มใกล้เคียงกัน ต่อให้เป็นคนละแนว
- 3.สามารถใส่เรื่องได้หลายเรื่องพร้อมกัน

CON

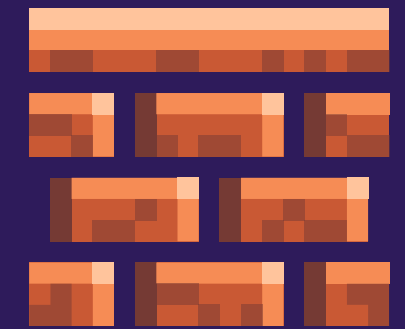
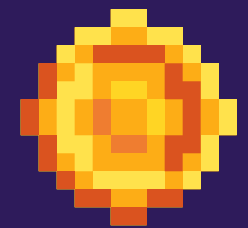
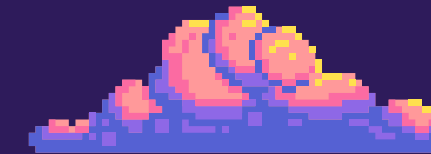
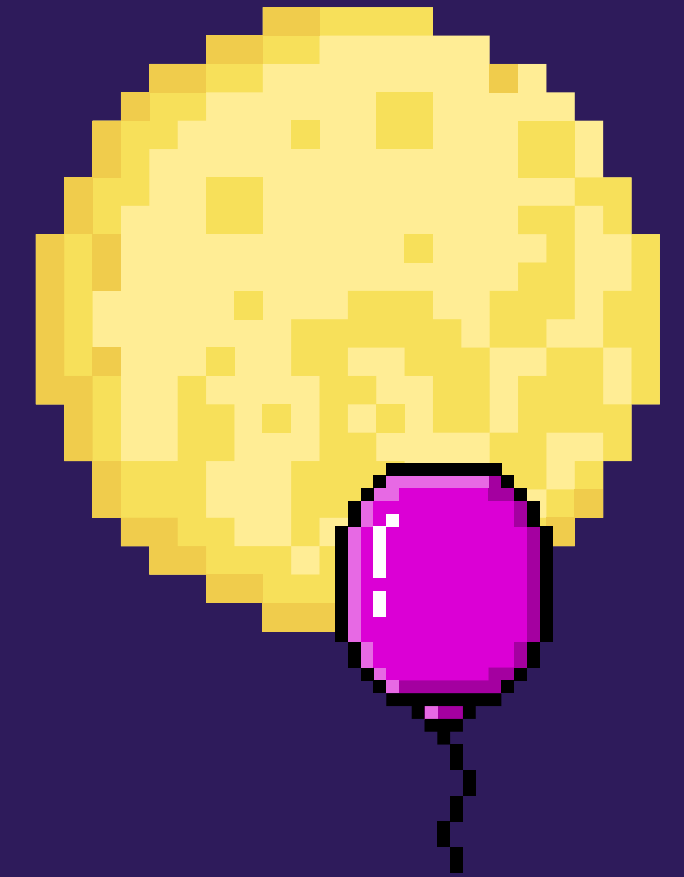


- 1. หากเรื่องที่ใส่ไปเป็นเรื่องที่ดังและดี จะทำให้ระบบแนะนำเรื่องที่ดังและดีเรื่องอื่น ๆ ด้วย โดยอาจจะเป็นคนละประเภทกัน
- 2. เนื่องจาก Feature เป็นคะแนนรีวิวจากแต่ละ user ทำให้วิธีการตัดสินใจเลือกคะแนนจากแต่ละคนจะไม่เหมือนกัน
- 3. บางเรื่องไม่มีใน Model เนื่องจาก มีคนร่ววน้อย

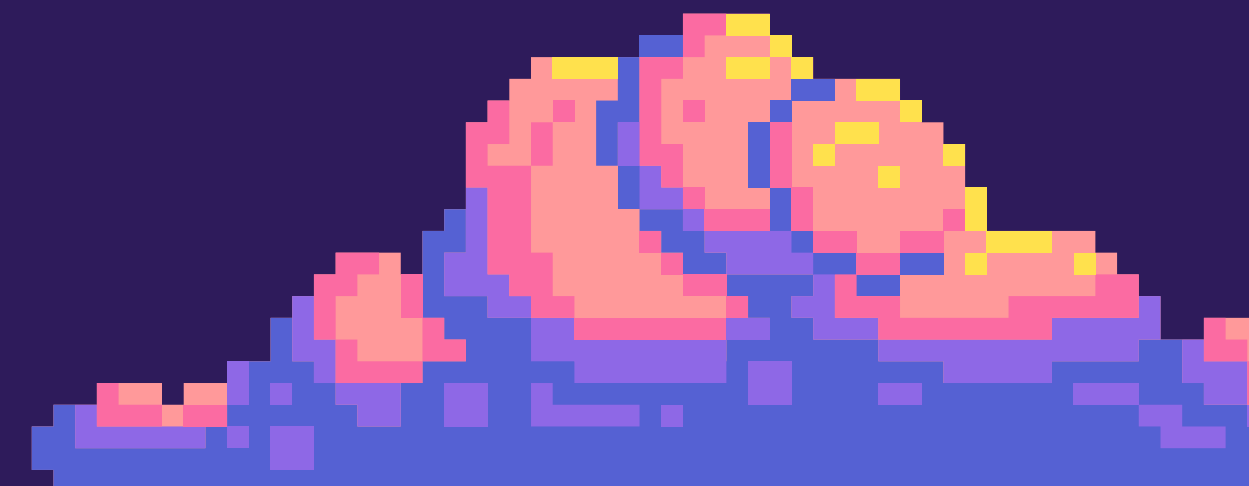


IMPORT LIBRARIES

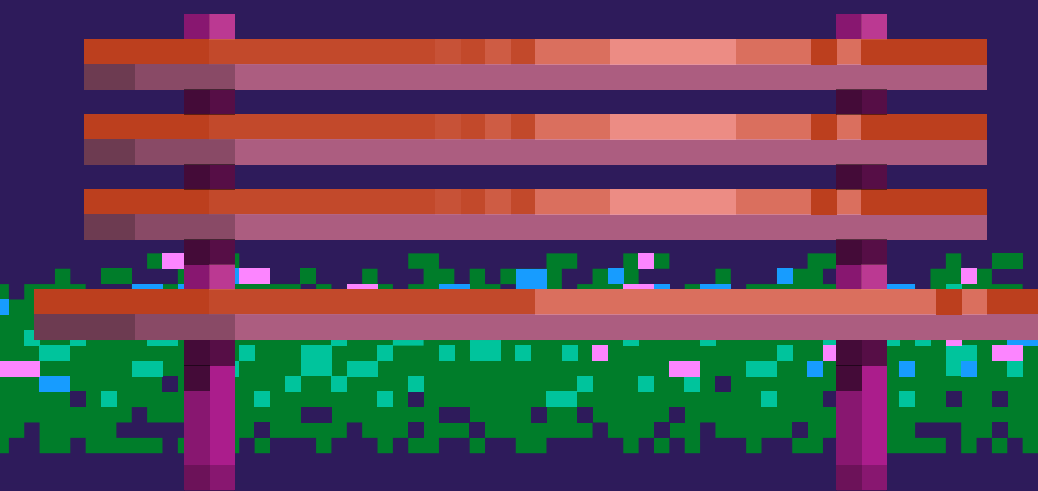
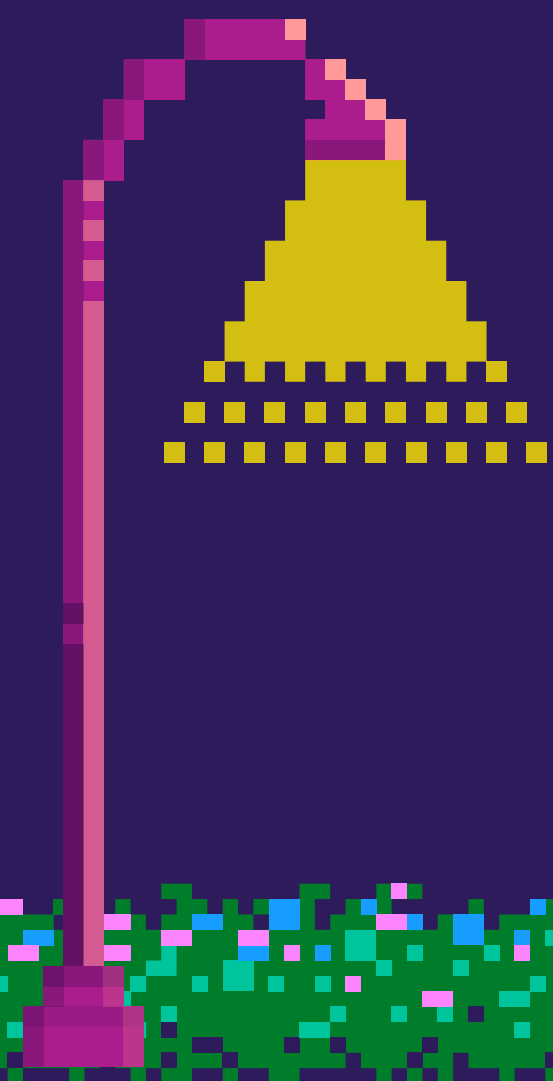
```
import pandas as pd
from scipy.sparse import csr_matrix
from sklearn.neighbors import NearestNeighbors
import re
import numpy as np
```



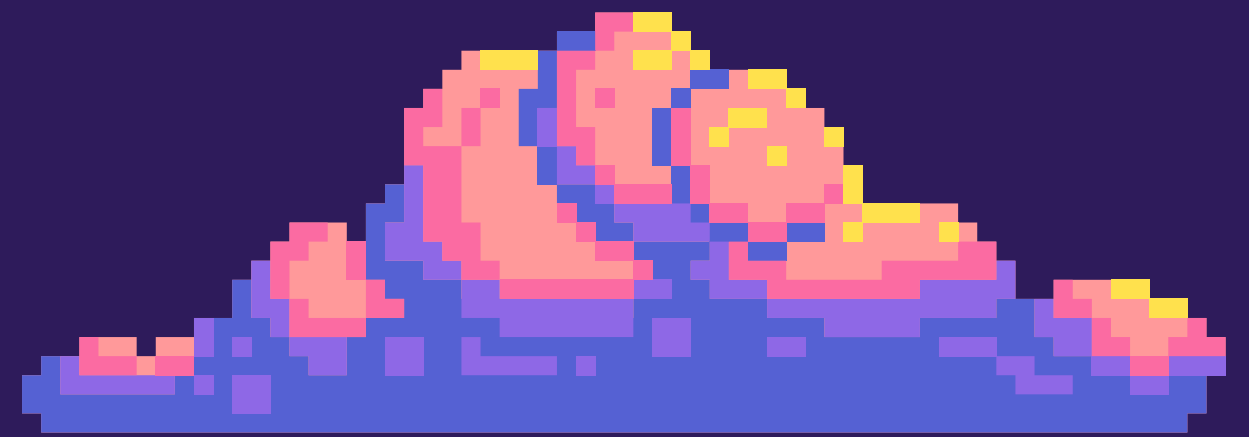
READ DATA



```
df1 = pd.read_csv("/content/animelist.csv")  
df2 = pd.read_csv("/content/anime.csv")
```



CLEAN DATA

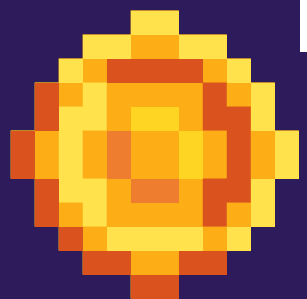


★ เนื่องจากชื่อเรื่องของอนิเมะบางเรื่อง มีตัวอักษรพิเศษติดมาด้วย เราจึงต้องลบมันออก เพื่อให้ง่ายต่อการเข้าถึง และนำมาใช้ต่อ

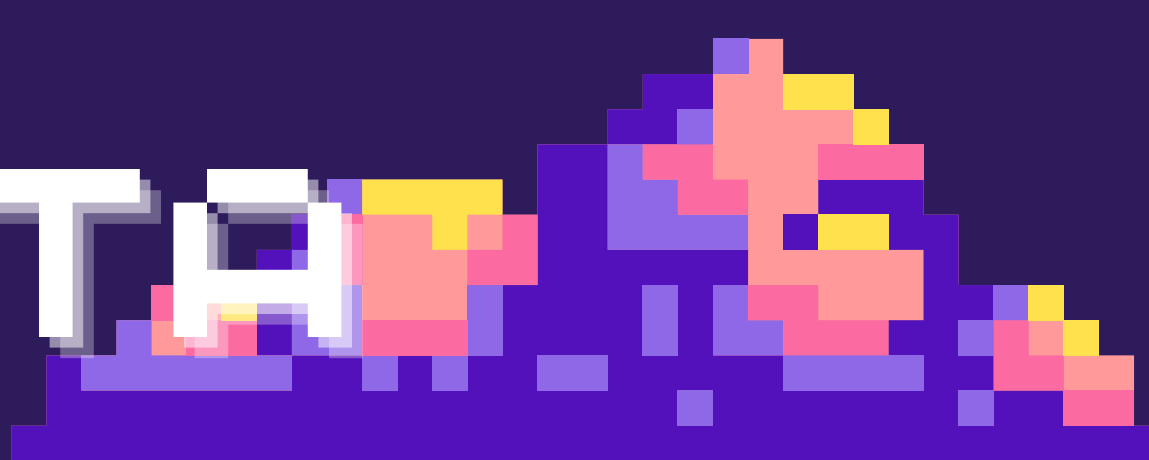
```
: #สร้าง function ลบตัวอักษรพิเศษ
def text_cleaning(text):
    otext = text
    ls1 = []
    text = re.sub(r'"', '', text)
    text = re.sub(r'.hack//', '', text)
    text = re.sub(r'&#039;', '', text)
    text = re.sub(r'A&#039;s', '', text)
    text = re.sub(r'I&#039;', 'I\\', text)
    text = re.sub(r'&', 'and', text)
    return text

: #สร้างตัว copy
anime_features = anime_complete.copy()
#ลบตัวอักษรพิเศษใน column "anime_title"
anime_features['Name'] = anime_features['Name'].apply(text_cleaning)
```

โดยทำการสร้างฟังก์ชัน `text_cleaning()` เพื่อไว้ลบตัวอักษรพิเศษโดยเฉพาะ ฟังก์ชันนี้ทำงานได้โดยใช้ คำสั่ง `.sub()` ที่มาจาก Library Regular expression หรือ `re` ซึ่งทำหน้าที่แทนที่ตัวอักษรในสตริง ไม่ว่าจะเป็น คำหรือประโยค



MERGE DATA



anime.csv (5.66 MB)

Detail Compact Column 10 of 35 columns

MAL_ID	Name	Score	Genres	English na...	Japanese ...
1	Cowboy Bebop	8.78	Action, Adventure, Comedy, Drama, Sci-Fi, Space	Cowboy Bebop	カウボーイビバップ
5	Cowboy Bebop: Tengoku no Tobira	8.39	Action, Drama, Mystery, Sci-Fi, Space	Cowboy Bebop:The Movie	カウボーイビバップ 天国の扉
6	Trigun	8.24	Action, Sci-Fi, Adventure, Comedy, Drama, Shounen	Trigun	トライガン
7	Witch Hunter Robin	7.27	Action, Mystery, Police, Supernatural, Drama, Magic	Witch Hunter Robin	Witch Hunter ROBIN (ウィッチハンターロビン)
8	Bouken Ou Beet	6.98	Adventure, Fantasy, Shounen, Supernatural	Beet the Vandel Buster	冒険王ビート
15	Eyeshield 21	7.95	Action, Sports, Comedy, Shounen	Unknown	アイシールド21
16	Hachimitsu to Clover	8.06	Comedy, Drama, Josei, Romance, Slice of Life	Honey and Clover	ハチミツとクローバー

+

animelist.csv (2.03 GB)

Detail Compact Column 5 of 5 columns

user_id	anime_id	rating	watching_sta...	watched_epis...
0	67	9	1	1
0	6702	7	1	4
0	242	10	1	4
0	4898	0	1	1
0	21	10	1	0
0	24	9	1	5
0	2104	0	1	4
0	4722	8	1	4
0	6098	6	1	2
0	3125	9	1	29
0	481	10	1	79
0	68	6	2	23
0	1689	6	2	3
0	2913	6	2	40
0	1250	7	2	26

CODE



```
[ ] #เลือก columns ที่จะใช้
    anime = df2.iloc[:,[0, 1]]
    #เปลี่ยนชื่อ columns ให้ตรงกับ df1
    anime.columns = ['anime_id', 'Name']
```

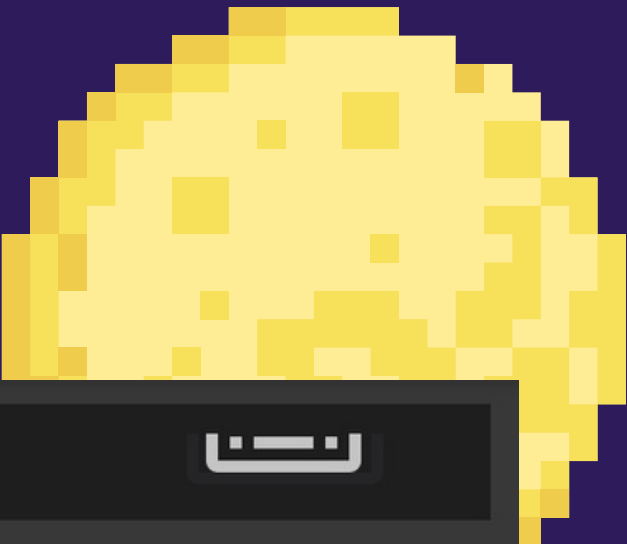


```
[ ] #innerjoin ระหว่าง df1 กับ anime
    anime_complete = pd.merge(df1, anime, on='anime_id')
    #check ตัว NaN (Not a Number)
    anime_complete.isna().sum()
```



```
user_id      0
anime_id     0
rating       0
watching_status 0
watched_episodes 0
Name         0
dtype: int64
```

RESULT



[] anime_complete



	user_id	anime_id	rating	watching_status	watched_episodes	Name
--	---------	----------	--------	-----------------	------------------	------

0	0	67.0	9.0	1.0	1.0	Basilisk: Kouga Ninpou Chou
---	---	------	-----	-----	-----	-----------------------------

1	14	67.0	0.0	6.0	0.0	Basilisk: Kouga Ninpou Chou
---	----	------	-----	-----	-----	-----------------------------

2	34	67.0	10.0	2.0	24.0	Basilisk: Kouga Ninpou Chou
---	----	------	------	-----	------	-----------------------------

3	55	67.0	0.0	1.0	0.0	Basilisk: Kouga Ninpou Chou
---	----	------	-----	-----	-----	-----------------------------

4	57	67.0	0.0	1.0	0.0	Basilisk: Kouga Ninpou Chou
---	----	------	-----	-----	-----	-----------------------------

...
-----	-----	-----	-----	-----	-----	-----

5453104	17559	30978.0	0.0	2.0	1.0	Okaasan no Yasashii te
---------	-------	---------	-----	-----	-----	------------------------

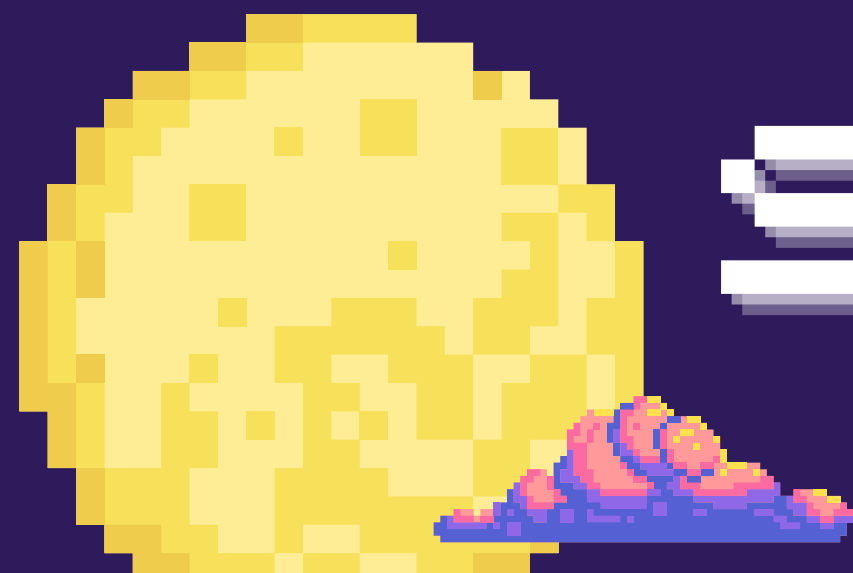
5453105	17559	34191.0	0.0	2.0	1.0	Santa-san wa Dai Isogashi
---------	-------	---------	-----	-----	-----	---------------------------

5453106	17559	26239.0	0.0	2.0	1.0	Tenki ni Naare
---------	-------	---------	-----	-----	-----	----------------

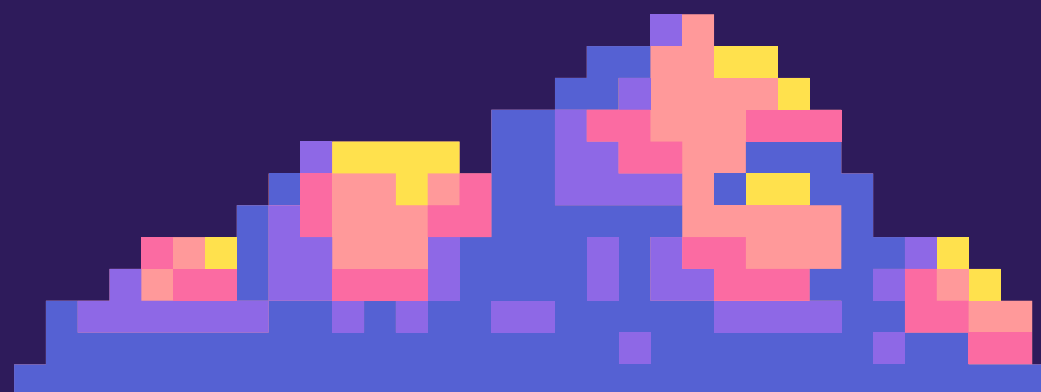
5453107	17559	33821.0	0.0	2.0	1.0	Tokyo Dai Kuushuu: Aoyo, Kaette Koi
---------	-------	---------	-----	-----	-----	-------------------------------------

5453108	17559	36325.0	0.0	2.0	1.0	Wao-kun no Hane
---------	-------	---------	-----	-----	-----	-----------------

5453109 rows x 6 columns



SPLIT



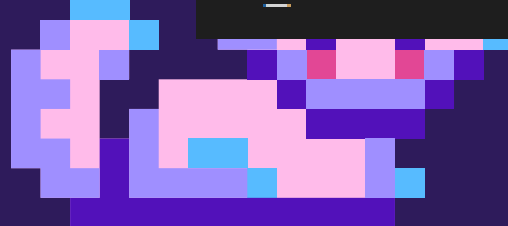
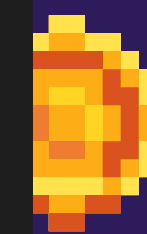
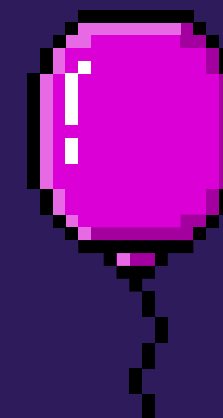
```
#สร้างตัว copy
anime_features = anime_complete.copy()
#ลบตัวอักษรพิเศษใน column "anime_title"
anime_features['Name'] = anime_features['Name'].apply(text_cleaning)

user_id_count = anime_features['user_id'].value_counts()
user_id_count = user_id_count[user_id_count >= 100]#กรอง user ที่รีวิวมากกว่า 100 ครั้ง
test_index_num = user_id_count.sample(frac = 0.3).index#แบ่งเป็น test 30%
anime_features = anime_features[anime_features['user_id'].isin(user_id_count.index)]

test = anime_features[anime_features['user_id'].isin(test_index_num)]#ถ้าใน column user_id มีตัวใน test_index_num ให้ใส่เข้าไป
train = anime_features[anime_features['user_id'].isin(test_index_num) == False]#ถ้าใน column user_id ไม่ได้มีตัวใน test_index_num ให้ใส่เข้าไป

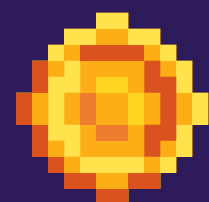
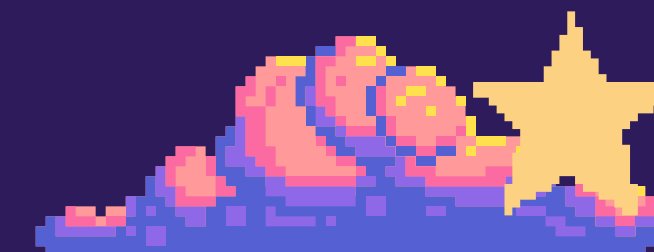
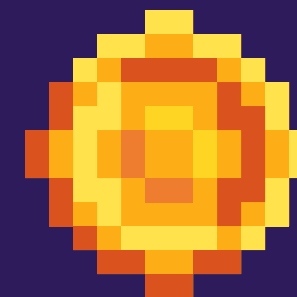
test = test.sort_values("user_id")
test = test[test["rating"] > 8.0]# rating มากกว่า 8
test = test.loc[:,["user_id","Name"]]  
# ลด column เหลือ แค่ 2 column

ls_test = list(test.groupby("user_id"))
```





PIVOT DATA

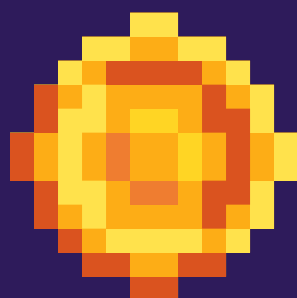


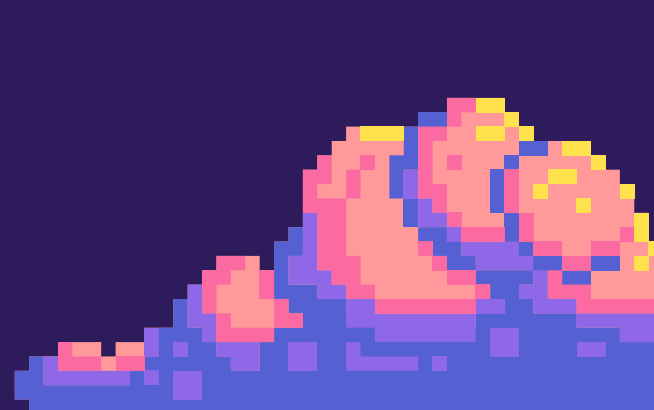
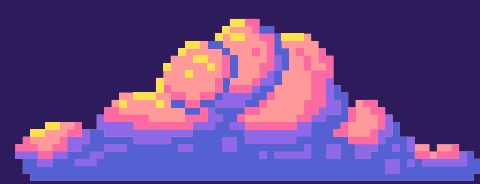
```
train_pivot = train.pivot_table(index='Name', columns = 'user_id', values = 'rating').fillna(0)
```

	user_id	1	2	3	4	5	6	7	8	11	12	...	1008	1009	1010	1013	1014	1016	1017	1020	1021	1023
	Name																					
	"0"	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	"Aesop" no Ohanashi yori: Ushi to Kaeru, Yokubatta Inu	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	"Bungaku Shoujo" Kyou no Oyatsu: Hatsukoi	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	"Bungaku Shoujo" Memoire	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	"Bungaku Shoujo" Movie	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

	xxxHOLiC Movie: Manatsu no Yoru no Yume	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	9.0	0.0	0.0	0.0
	xxxHOLiC Rou	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	8.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	xxxHOLiC Shunmuki	0.0	0.0	0.0	8.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	8.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	ēiDLIVE	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	6.0	0.0
	○	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

12381 rows × 701 columns





SERIES -> CSR_MATRIX

```
[ ] #เปลี่ยนจาก series เป็น csr_matrix (Compressed Sparse Row Matrix) เพื่อให้เวลา fit ข้อมูลใน Knn มันเร็วขึ้น  
anime_matrix = csr_matrix(anime_pivot.values)
```

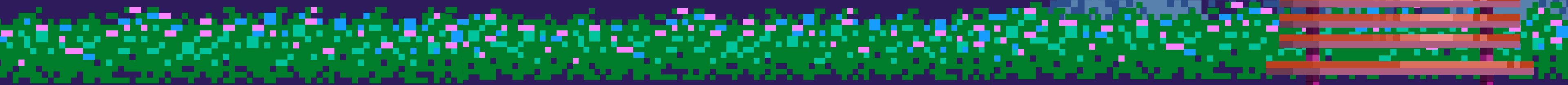


แปลง series ที่มี cellจำนวนมาก และส่วนมากมีค่าเป็น 0

les nu cell๑๒๓๔๕๖๗๘๙๑๒๓๔๕๖



ให้เป็น csr matrix (Compress Spares Row matrix)



CSR_MATRIX

CSR matrix (Compress Sparse Row matrix)

Dense Matrix

	0	1	2	3
0	a	0	0	0
1	0	b	0	0
2	c	0	d	0
3	e	0	f	g

Sparse Matrix in CSR

rowptr

0	1	2	4	7
---	---	---	---	---

col

0	1	0	2	0	2	3
---	---	---	---	---	---	---

val

a	b	c	d	e	f	g
---	---	---	---	---	---	---

row index ของ value นั้น

column index ของ value นั้น

เอาตัว values ที่เป็น nonzeros

CSR_MATRIX

CSR matrix (Compress Sparse Row matrix)

```
row = np.array([0, 0, 1, 2, 2, 2])
col = np.array([0, 2, 2, 0, 1, 2])
data = np.array([1, 2, 3, 4, 5, 6])
print(csr_matrix((data, (row, col)), shape=(3, 3)))
print("-----")
print(csr_matrix((data, (row, col)), shape=(3, 3)).toarray())
```

index (row, col) →

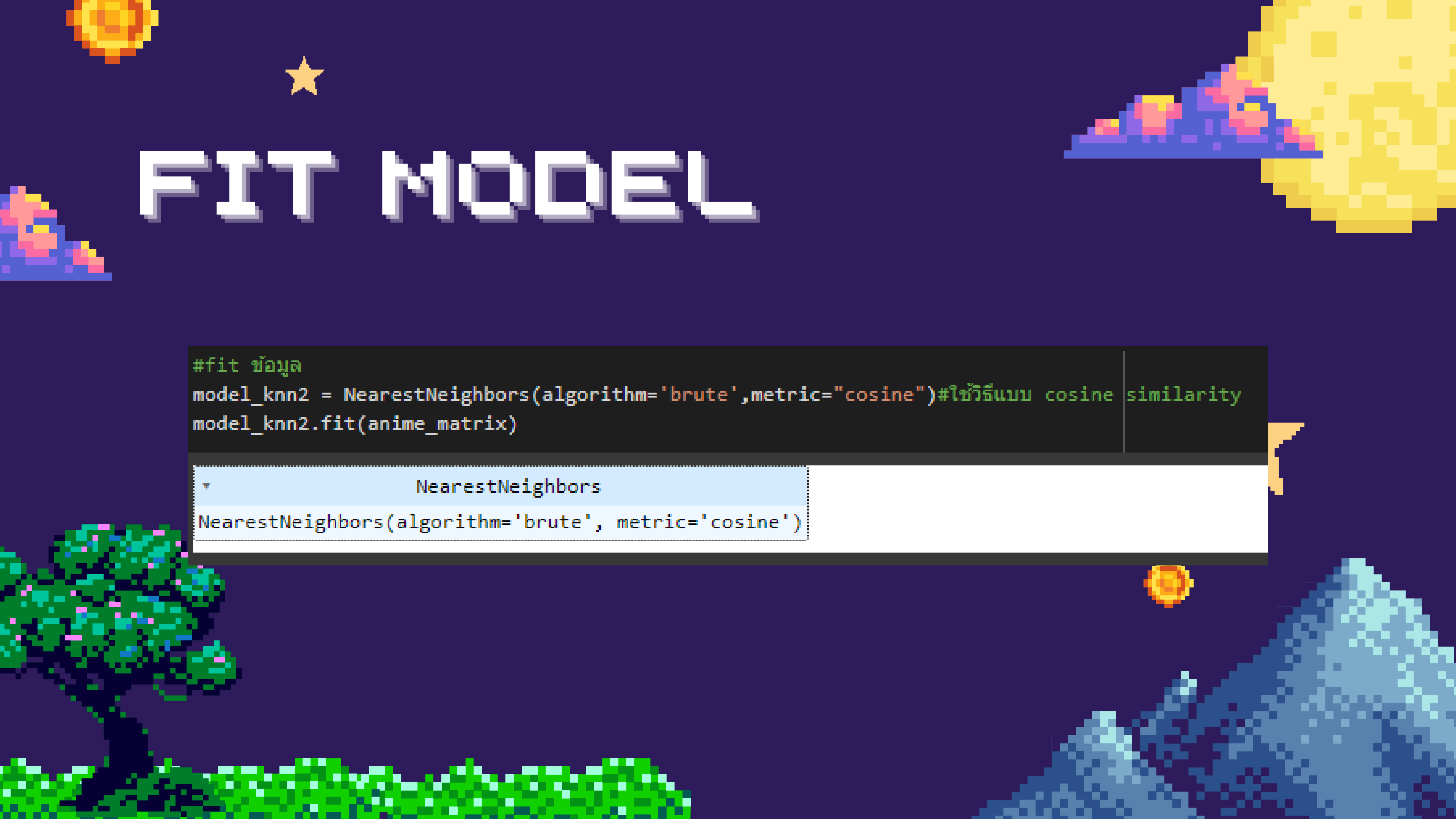
(0, 0)
(0, 2)
(1, 2)
(2, 0)
(2, 1)
(2, 2)

1
2
3
4
5
6

data

[[1 0 2]
[0 0 3]
[4 5 6]]

ทำเพื่อเพิ่มความเร็วในการ fit ข้อมูล



FIT MODEL

```
#fit ข้อมูล  
model_knn2 = NearestNeighbors(algorithm='brute',metric="cosine")#ใช้วิธีแบบ cosine similarity  
model_knn2.fit(anime_matrix)
```

NearestNeighbors
NearestNeighbors(algorithm='brute', metric='cosine')

FUNCTION

```
def give_ls_knn(model=model_knn2,anime_title = [np.random.choice(anime_pivot.index)],anime_pivot=anime_pivot, num = 100):
    a = np.array([[]])
    b = np.array([[]])
    # Print the selected anime title
    print("*****")
    print(f"Selected anime title: {anime_title}")
    # Find the row index of the selected anime title
    # ใช้ knn โดยหาว่าตัวไหนใกล้มันที่สุด
    for var in anime_title:
        query_index = anime_pivot.index.get_loc(var)#เปลี่ยนชื่อ anime เป็น index
        distances, indices = model.kneighbors(anime_pivot.iloc[query_index, :].values.reshape(1, -1), n_neighbors=num) #เลือกมาใกล้ตัวที่ใกล้เคียงกัน
        #เอาแต่ละรอบมารวมกันเป็น Array เดียว
        a = np.concatenate((a,distances),axis = 1)
        b = np.concatenate((b,indices), axis = 1)
```

DROP ตัวที่ไม่ใช่

```
con = np.transpose(np.concatenate((b,a), axis = 0)) #Transpose  
df = pd.DataFrame(con, columns= ["Indices", "Distances"])#สร้าง Dataframe
```

```
#หาว่า anime เรื่องไหนไม่อยู่ใกล้ภายใน 100 ตัวแรกของ anime ที่อยู่ใน anime_title ทั้งหมด  
df2 = df.groupby("Indices").count() != len(anime_title)  
#เอา anime_id จากเงื่อนไขข้างบนแปลงเป็น list  
ls1 = [int(var) for var in list(df.groupby("Indices").sum()["Distances"][df2["Distances"]].index)]  
#แปลง index ของ df ให้เป็นตาม anime_id  
df.index = ([int(var) for var in list(df["Indices"])])  
#drop ตัวที่เข้าเงื่อนไขออก  
for var in ls1:  
    df = df.drop(var)
```

DROP ตัวที่ไม่ใช่

```
#sum distance ทั้งหมดของแต่ละตัวและเรียงตาม Distance จากน้อยไปมาก
df3 = df.groupby("Indices").sum().apply(lambda x: x).sort_values(['Distances', 'Indices'])
#แยก Indices เป็น list
indices = [int(var) for var in list(df3.index)][len(anime_title):len(anime_title)+20]
#แยก Distances เป็น list
distances = list(df3["Distances"])[len(anime_title):len(anime_title)+20]
```

SIGNiFiCANT

```
minest = []  
# หากว่ามีเรื่องไหนจาก 20 เรื่อง ที่แนะนำพิเศษสำหรับ กลุ่มเมะ ใน anime_title อันไหนเป็นพิเศษ  
for var2 in range(len(indices)):  
    check = a[np.where(b == indices[var2])]#Distance ของตัวนั้นจาก แต่ละเรื่องใน anime_title  
    add = []  
    scoring = distances[var2]*((1/len(anime_title))*0.75)#เกณฑ์  
    if len(check[np.where(check < scoring)]) != 0:  
        for var in check[np.where(check < scoring)]:  
            add.append(anime_title[list(check).index(var)])#เชื่อว่าตัวที่น้อยกว่าเกณฑ์มาจากเรื่องไหน แล้วเพิ่มเข้า add  
        minest.append(add)#เพิ่ม add เข้าไปใน minest  
    else:  
        minest.append(len(anime_title))
```

CODE

```
minest = []  
# หากว่ามีเรื่องไหนจาก 20 เรื่อง ที่แนะนำพิเศษสำหรับ กลุ่มเมะ ใน anime_title อันไหนเป็นพิเศษ  
for var2 in range(len(indices)):  
    check = a[np.where(b == indices[var2])]#Distance ของตัวนั้นจาก แต่ละเรื่องใน anime_title
```

print(check)

distance ของอนิเมะ
เรื่องนี้กับ Naruto

```
Selected anime title: ['Naruto', 'One Punch Man']  
[0.32728082 0.23642234]  
[0.16926407 0.46056262]  
[0.29683518 0.33551759]  
[0.41708418 0.21763665]  
[0.40168828 0.27224636]  
[0.37407734 0.30133739]  
[0.42833559 0.26319651]  
[0.36636764 0.32538038]  
[0.45845931 0.28196036]  
[0.43612404 0.31017056]
```

CODE

```
add = []
scoring = distances[var2]*((1/len(anime_title))*0.75)#เกณฑ์
if len(check[np.where(check < scoring)]) != 0:
    for var in check[np.where(check < scoring)]:
        add.append(anime_title[list(check).index(var)])#เชื่อว่าตัวที่น้อยกว่าเกณฑ์มาจากเรื่องไหน แล้วเพิ่มเข้า add
    minest.append(add)#เพิ่ม add เข้าไปใน minest
else:
    minest.append(len(anime_title))
```

```
minest = []
```

→ **print(scoring)**

```
Selected anime title: ['Naruto', 'One Punch Man']
[0.32728082 0.23642234]
0.21138868841774422
[0.16926407 0.46056262]
0.2361850070074679
[0.29683518 0.33551759]
0.2371322905156884
[0.41708418 0.21763665]
0.23802030962315468
[0.40168828 0.27224636]
0.25272549169472824
[0.37407734 0.30133739]
0.25328052161084436
[0.42833559 0.26319651]
0.2593245369380103
[0.36636764 0.32538038]
0.25940550705694354
[0.45845931 0.28196036]
0.27765737493597553
[0.43612404 0.31017056]
0.2798604752529387
```


RESULT

Selected anime title: ['Naruto', 'One Punch Man']

[0.32728082 0.23642234]

0.21138868841774422

[0.16926407 0.46056262]

0.2361850070074679

[0.29683518 0.33551759]

0.2371322905156884

[0.41708418 0.21763665]

0.23802030962315468

[0.40168828 0.27224636]

0.25272549169472824

[0.37407734 0.30133739]

0.25328052161084436

[0.42833559 0.26319651]

0.2593245369380103

[0.36636764 0.32538038]

0.25940550705694354

[0.45845931 0.28196036]

0.27765737493597553

[0.43612404 0.31017056]

0.2798604752529387



give_ls_knn(anime_title=["Naruto", "One Punch Man"], model=model_knn2)

Selected anime title: ['Naruto', 'One Punch Man']

01: Shingeki no Kyojin, with total distance of 0.56

02: Naruto: Shippuuden, with total distance of 0.63

Significant from : Naruto

03: Death Note, with total distance of 0.63

04: Boku no Hero Academia. with total distance of 0.63

Significant from : One Punch Man

05: Tokyo Ghoul, with total distance of 0.67

06: Sword Art Online, with total distance of 0.68

07: No Game No Life, with total distance of 0.69

08: Fullmetal Alchemist: Brotherhood, with total distance of 0.69

09: Boku no Hero Academia 2nd Season, with total distance of 0.74

10: Nanatsu no Taizai, with total distance of 0.75

OUTPUT

```
ans = []
print("*****")
#print ชื่อ anime ที่ recommend
for i, (distance, index) in enumerate(zip(distances, indices)):#len(anime_title): คือการตัดตัวเรื่องจาก anime_title ออก
    txt = ""
    if minest[i] != len(anime_title) and len(anime_title)!= 1:#เชื่อว่าเรื่องนั้น แนะนำพิเศษสำหรับอนิเมะเรื่องไหน หรือไม่
        txt = "Significant from :"
        if len(minest[i])!= 1:
            for var in minest[i]:
                txt += f" {var}"
        else:
            txt += f" {minest[i][0]}"
    print(f"{i+1:02d}: {anime_pivot.index[index]}, with total distance of {distance:.2f}")
    ans.append(anime_pivot.index[index])
    if txt != "":
        print(txt)
    print("-----")
return ans
```

OUTPUT

```
give_ls_knn(anime_title=["One Piece","Fairy Tail", "Bleach"], model=model_knn2)
```

```
*****
```

```
Selected anime title: ['One Piece', 'Fairy Tail', 'Bleach']
```

```
*****
```

```
01: Naruto, with total distance of 1.15
```

```
02: Naruto: Shippuuden, with total distance of 1.22
```

```
03: Death Note, with total distance of 1.26
```

```
04: Shingeki no Kyojin, with total distance of 1.32
```

```
05: Fullmetal Alchemist: Brotherhood, with total distance of 1.34
```

```
06: Ao no Exorcist, with total distance of 1.34
```

```
07: Code Geass: Hangyaku no Lelouch, with total distance of 1.34
```

```
08: Sword Art Online, with total distance of 1.35
```

```
09: Code Geass: Hangyaku no Lelouch R2, with total distance of 1.43
```

```
10: Fairy Tail (2014), with total distance of 1.47
```

```
Significant from : Fairy Tail
```

```
-----  
11: Soul Eater, with total distance of 1.47
```

```
-----  
12: One Punch Man, with total distance of 1.47
```

```
-----  
13: Nanatsu no Taizai, with total distance of 1.47
```

```
-----  
14: Tokyo Ghoul, with total distance of 1.48
```

```
-----  
15: Highschool of the Dead, with total distance of 1.48
```

```
-----  
16: Fullmetal Alchemist, with total distance of 1.49
```


```
-----  
17: Hunter x Hunter (2011), with total distance of 1.49
```

```
-----  
18: Angel Beats!, with total distance of 1.51
```

```
-----  
19: Boku no Hero Academia, with total distance of 1.51
```

```
-----  
20: No Game No Life, with total distance of 1.52
```

```
-----
```



EVALUATE

```
score = 0
for var in ls_test:
    ls1 = var[1]["Name"].values[5:]
    ls2 = give_ls_knn(anime_title = list(var[1]["Name"].values[:5]))
    for i in ls1:
        if i in ls2:
            score += 1
            break
```

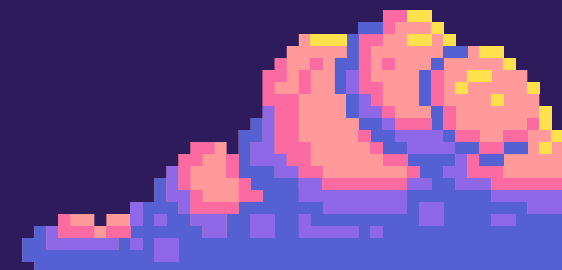
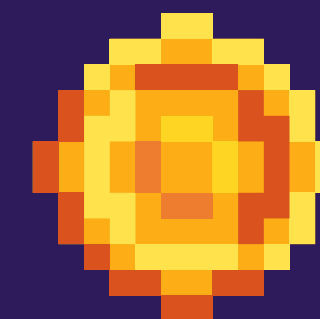
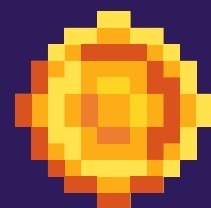
```
print(f"Accuracy:{score/len(ls_test)}")
```

```
Accuracy:0.4789473684210526
```

MEMBER

- 65070012 Miss GAVINNAR POONVILYLE
- 65070044 Mr. CHANANYU LIMCHAROEN
- 65070054 Miss YANOUC JADSON
- 65070063 Miss NAPAT WANDEE
- 65070134 Miss PANNATHORN KOMKRIS
- 65070172 Miss PATSORN JIWAROENG
- 65070173 Mr. PAKIN KITTICHAIKULKIT
- 65070227 Mr. SUPPHAWIT PERKSUAN





THANKYOU

