ชื่อ-สกุล_____ห้อง_____รหัสนักศึกษา_____

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

วิชา Machine Learning Laboratory

การทดลองที่ 4 : การทดลองปรับค่าพารามิเตอร์เพื่อสร้างโมเดลการแนะนำสินค้าและการจัดกลุ่ม

วัตถุประสงค์

1. เพื่อศึกษาและทดลองการใช้งานโมเดลแนะนำสินค้า
2. เพื่อศึกษาและทดลองการปรับค่าพารามิเตอร์ที่เหมาะสมกับโมเดลสำหรับชุดข้อมูลทดสอบ

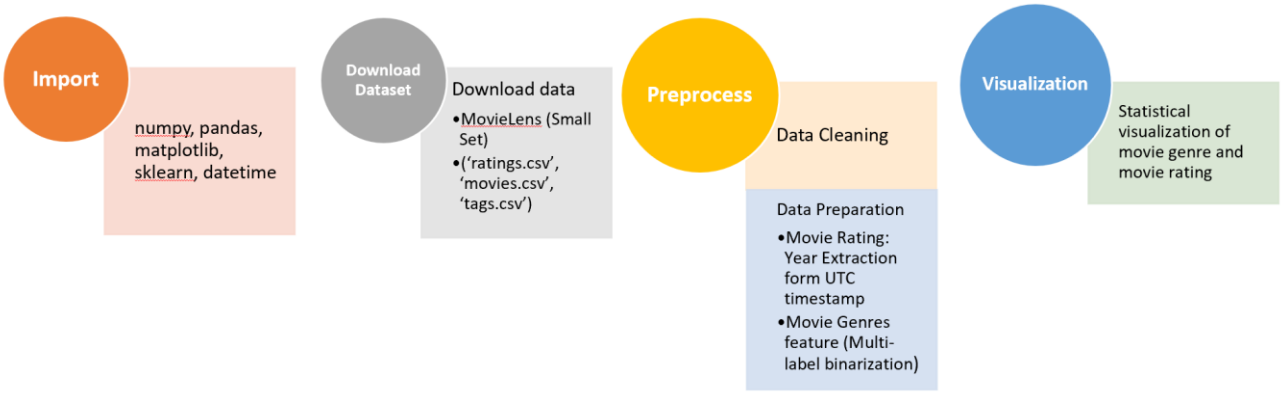
อุปกรณ์ และเครื่องมือที่ใช้ในการทดลอง

1. โปรแกรม python

ข้อกำหนดในการตรวจการทดลอง

1. แสดง source code และภาพผลการทดลองที่ทำพร้อมอธิบาย โดยกลุ่มใดพร้อม ให้ Post แจ้งชื่อกลุ่มและ Email ที่ต้องการให้ติดต่อไว้ใน Facebook group เพื่อตรวจการทดลองตามหัวข้อส่งงาน โดยจะทำการตรวจผ่าน google hangout -> share screen ตามลำดับที่แจ้งไว้
2. นศ.ที่ได้รับการตรวจจากอาจารย์เรียบร้อย อาจารย์จะเช็คส่งงานในระบบ
3. ให้นศ. นำ source code และ ภาพ figure ที่ให้แสดงทุกภาพ โพสลงใน google form พร้อมตอบคำถามท้ายการทดลอง ส่งภายในวันที่ 13 เม.ย. 2563 เวลา 18.00 น.

ตอนที่ 1: การทดลองเตรียมข้อมูลและแสดงรายละเอียดข้อมูลเชิงกราฟ



1.1 Import Lib (numpy, pandas, matplotlib, sklearn, datetime)

1.2 โหลดข้อมูล MovieLens Dataset file ('ratings.csv', 'movies.csv', 'tags.csv') โดยใช้

1.3 Data Preprocessing

1.3.1 Data Cleaning

1.3.2 Data Preparation

- เตรียมข้อมูล movie rating ด้วยการดึงข้อมูลปีคศ. ('year') จากข้อมูล UTC 'timestamp'
- เตรียมข้อมูล movie genre feature โดยกำหนดให้มีคอลัมน์ดังนี้โดยกำหนดให้ปรับรูปแบบ genres เป็น multi-label binarization

No.	movieid	Movie title	No genres Listed	Action	Adventure	...	Western
0	1	0	0	0	1	...	0

1.4 Data Visualization แสดงกราฟข้อมูลการเปลี่ยนแปลงของ movie genres และ movie rating ในแต่ละปี

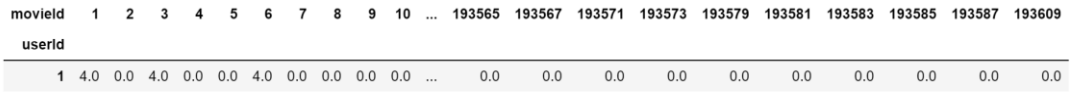
- กราฟที่ 1: แสดงกราฟค่า จำนวน released movies ในแต่ละปี
- กราฟที่ 2: แสดงกราฟค่า จำนวนการให้ rating ในแต่ละปี
- กราฟที่ 3: แสดงกราฟค่า จำนวน movies ในแต่ละ genre
- กราฟที่ 4: แสดงกราฟ (y-axis: stacked graph) ค่าจำนวน movie แต่ละ genre ในแต่ละปี (x-axis)
- กราฟที่ 5: แสดงกราฟ Histogram ของการกระจายของค่าเฉลี่ย movie rating ใน dataset โดยค่าเฉลี่ย movie rating คำนวณจาก ค่า rating เฉลี่ยของแต่ละ movie เพื่อเป็นข้อมูล input ให้กับ histogram (โดยให้กำหนดจำนวน histogram bins ที่จะเห็นรายละเอียด)

ตอนที่ 2: การทดลองสร้างระบบแนะนำสินค้า (Recommendation system) จากข้อมูล user_matrix

2.1 สร้างข้อมูลความชอบของผู้ใช้แต่ละคน (user_ matrix)

- row: user_id (sorted)

- column: rating for each movie of each user



movieId	1	2	3	4	5	6	7	8	9	10	...	193565	193567	193571	193573	193579	193581	193583	193585	193587	193609
userId																					
1	4.0	0.0	4.0	0.0	0.0	4.0	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

2.2 คำนวณความคล้ายกันของความชอบดูหนังของคู่ ‘userId’ ใดๆ

2.2.1 สุ่มหยิบข้อมูล user_matrix มาจำนวน nUser ไม่น้อยกว่า 20 คน

2.2.2 คำนวณความคล้ายของความชอบ movie ของคู่ user โดยใช้ตัววัด cosine_similarity()

สำหรับข้อมูล user_matrix ที่สุ่มขึ้นมาจากข้อ 2.2.1 โดยใช้

cosine_similarity(user_matrix)

2.2.3 คำนวณความคล้ายของความชอบ movie ของคู่ user โดยใช้ตัววัด Pearson’s similarity()

สำหรับข้อมูล item_matrix ที่คำนวณจาก user_matrix^{T} (user_matrix transpose) โดยใช้

df.user_matrix.T.corr (method ='pearson')

2.3 แสดงตารางรายการดังนี้

2.3.1 ตาราง user ที่มีความชอบคล้ายกันที่สุด 5 อันดับ (เปรียบเทียบจากการวัดความคล้ายด้วย cosine_similarity() ข้อ 2.2.2 และ Person’s similarity ข้อ 2.2.3

2.3.2 ตาราง user ที่มีความชอบตรงกันข้ามกันที่สุด 5 อันดับ (จากการวัดความคล้ายด้วย Pearson’s similarity ข้อ 2.2.3)

2.3.3 ตาราง จากผลการวัดความคล้ายกันของแต่ละ user_matrix ให้สร้างรายการของคนที่มีความชอบคล้ายกันที่สุด และรายการคนที่มีความชอบตรงข้ามกันที่สุด โดยแสดงในรูปของกราฟความชอบ

2.4 แสดงรูปภาพ

2.4.1 กราฟความคล้ายของความชอบ movie ที่วัดด้วย Pearson’s similarity() โดยให้แสดงเส้นเชื่อมความชอบของคู่ user ตามเงื่อนไข 2 แบบต่อไปนี้

- เฉพาะความชอบคล้ายกันที่เกินค่าที่กำหนด (Th) เป็นสี โดยแบ่งระดับความชอบที่เกินค่าที่กำหนดเป็น 3 ช่วงระดับ เพื่อแสดงเส้นกราฟเป็นสี 3 สี (ให้เลือกสีที่ให้ความรู้สึกทางบวก)

- เฉพาะความชอบที่ตรงข้ามกันที่เกินค่าที่กำหนด (Th) เป็นสี โดยแบ่งระดับความชอบตรงข้ามที่เกินค่าที่กำหนดเป็น 3 ช่วงระดับ เพื่อแสดงเส้นกราฟเป็นสี 3 สี (ให้เลือกสีที่ให้ความรู้สึกทางลบ)

โดยใช้ฟังก์ชันกราฟดังนี้

```
from networkx import nx
from matplotlib.lines import Line2D

# Create New Graph
G = nx.Graph()

# Create #node = #user ใน Pearson's similarity และใส่ label เป็น user_id ในแต่ละ node
for x in ( user_id จาก ความคล้ายจาก Pearson's Similarity ):
        G.add_node( x , label )

# Create #edge of graph ตามค่าใน Pearson's similarity ที่เป็นตามเงื่อนไข 2 เงื่อนไขที่ตั้งไว้ข้างต้น
คือแสดงสีเฉพาะ user ที่ชอบคล้ายกันเกิน Th 3 ระดับสี และชอบตรงข้ามกันเกิน Th 3 ระดับสี
   Ex.
        # Add Edges with weight and assign color to the lines depend on 3 conditions
        for x in range( #user_id in Pearson's similarity) ):
          for y in range(x+1, len (#user_id in Pearson's similarity) ):
             if Pearson's similarity [ index(x), index(y) ] > Th1 :
               G.add_edge( x, y,
                           weight = Pearson'similarity ของคู่ user (x,y),
                           color="color#1")
            elif Pearson's similarity [ index(x), index(y) ] > Th2 :
               G.add_edge( x, y,
                           weight = Pearson'similarity ของคู่ user (x,y),
                           color="color#2")
```

2.4.2 แสดงรูปตาราง movie title ที่ rating สูงสุด ที่ควรแนะนำของคนที่มีความชอบคล้ายกันที่สุดที่ควรแนะนำให้ดู

Row: user_id in Pearson's similarity

Column:

#1: user_id ที่ชอบคล้ายที่สุด

#2: movie title ที่ user_id นั้น ให้ rating สูงสุด

#3: movie rating ของ #2

2.4.3 แสดงรูปตาราง movie title ที่ rating สูงสุด ของคนที่มีความชอบตรงข้ามกันที่สุด ที่ไม่ควรแนะนำให้ดู

Row: user_id in Pearson's similarity

Column:

#1: user_id ที่ชอบตรงข้ามที่สุด

#2: movie title ที่ user_id นั้น ให้ rating สูงสุด

#3: movie rating ของ #2

ตอนที่ 3: การทดลองสร้างระบบแนะนำสินค้า (Recommendation system) จากข้อมูล movie_matrix

3.1 คำนวณความคล้ายกันของ movie genre ของคู่ 'movieId' ใดๆ จากตาราง movie genre feature จากข้อ 1.3.2 โดย

3.1.1 สุ่มหยิบข้อมูล movie_matrix จำนวน n movies จาก movie genre feature

(ไม่น้อยกว่า 20 เรื่อง)

3.2.2 คำนวณโดยใช้ตัววัด cosine_similarity() สำหรับข้อมูล movie_matrix จากข้อ 3.2.1

3.2.3 คำนวณโดยใช้ตัววัด Pearson's similarity สำหรับข้อมูล movie_matrix จากข้อ 3.2.1

Ex.

movieId	2606	162578	175475
movieId			
2606	1.000000	-0.166667	0.326732
162578	-0.166667	1.000000	0.490098
175475	0.326732	0.490098	1.000000

3.2 แสดงรูปภาพ

3.2.1 รายการของ movie ที่มีประเภท (genre) คล้ายกันที่สุด 5 อันดับ พร้อมค่า Pearson's Sim

3.2.2 รายการของ movie ที่มีประเภท (genre) ตรงข้ามกันที่สุด 5 อันดับ พร้อมค่า Pearson's Sim

3.2.3 รายการของ user ที่ให้ rating >= 3.0 ซึ่งสามารถแนะนำ movie ในรายการข้อ 3.2.1 ให้ได้

ตอนที่ 4: การจัดกลุ่ม User ที่มีความชอบคล้ายกันด้วยเทคนิค k-mean และ Gaussian Mixture

4.1 สร้างข้อมูลความชอบของ user ตาม genre ของ movie ในรูปของ genre_rating_matrix โดย

Row: user_id

Column: average rating for each genre of each user

4.2 สำหรับ K-mean model ให้กำหนดจำนวน n_cluster

- ตัวอย่าง Lib: from sklearn.cluster import KMeans (สามารถใช้ตัวอื่นได้)
- ประกาศโมเดล: Kmean(n_clusters)
- แบ่งข้อมูลเป็นชุด Train, Validate, Test
- Train: model.fit()
- Validation Test: model.prdict(validate)
- แสดงรูปภาพผลลัพธ์ของการจัดกลุ่ม กำหนดให้เทียบรูปภาพอย่างน้อย n_cluster อย่างน้อย 3 ค่า
  - List ของ kmeans.cluster_centers_
  - 2D Scatter Plot
    1. Plot DataSamples: เลือก features 1 คู่ที่เมื่อแสดงภาพแล้วเห็นการแบ่งกลุ่ม ตามจำนวน n_clusters ได้ดีที่สุด

```
plt.scatter( f1, f2,
            c=prediction_label,
            s=radius1, cmap='...')
```

    2. Plot Cluster Center:

```
plt.scatter( f1_centriod, f2_centroids, c  =selected_color, s = radius2)
```

4.3 สำหรับ Gaussian Mixture Model

- ตัวอย่าง Lib: from sklearn.mixture import GMM (สามารถใช้ตัวอื่นได้)
- ประกาศโมเดล: GMM(n_components, covariance_type='full' , random_state)
- แบ่งข้อมูลเป็นชุด Train, Validate, Test
- Train: model.fit()
- Validation Test: model.prdict(validate)
- แสดงรูปภาพผลลัพธ์ของการจัดกลุ่ม กำหนดให้เทียบรูปภาพอย่างน้อย n_cluster อย่างน้อย 3 ค่า
  - List ของ gmm_model.means_, gmm_model.covars_, gmm_model.weights_

- 2D Scatter Plot

```
def plot_gmm(gmm_model, X, col_id0, col_id1, label=True, ax=None):
    ax = ax or plt.gca()
    labels = gmm_model.fit(X).predict(X)
    if label:
       ax.scatter( X[:, col_id0], X[:, col_id1],
                   c=labels, s= radius1, cmap = color1, zorder=2)
    else:
        ax.scatter(X[:, col_id0], X[:, col_id1],
                   s=radius2, zorder=2)
    ax.axis('equal')
    w_factor = 0.2 / gmm_model.weights_.max()
    for pos, covar, w in
          zip(gmm_model.means_, gmm_model.covars_, gmm_model.weights_):
                   draw_ellipse(pos, covar, alpha=w * w_factor)
```

Tutorial

Recommendation Dataset

[1] MovieLens: https://grouplens.org/datasets/movielens/