KHUDA

다양한 추천 알고리즘을 활용한 책 추천시스템 구현

발표자1: 쿠다 4기 이준혁

발표자2: 쿠다 4기 손형진

목차

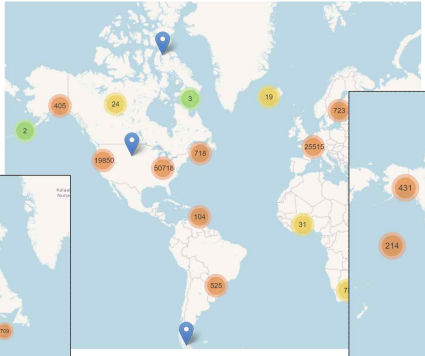
01 주제 선정 배경

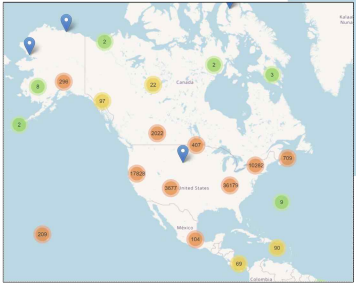
KHUDA

왜 이 주제를 골랐냐면…

02 EDA 과정

KHUDA





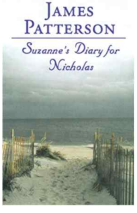
02 EDA 과정

KHUDA

도서 추천 예시

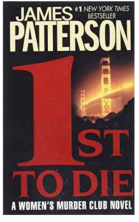
WOULD YOU LIKE to TRY THESE BOOKS?

RATING: 7.9

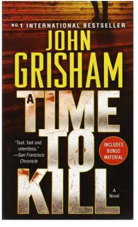


RATING: 7.7

RATING: 9.1



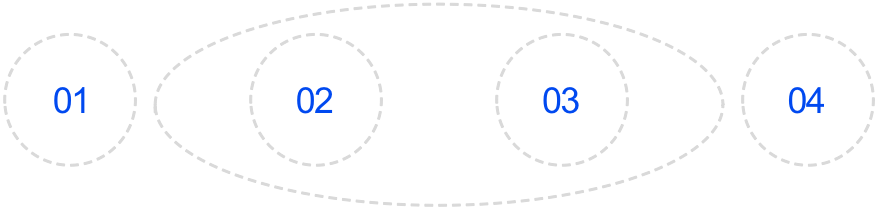
RATING: 7.7



RATING: 8.0

03 알고리즘 및 모델

KHUDA

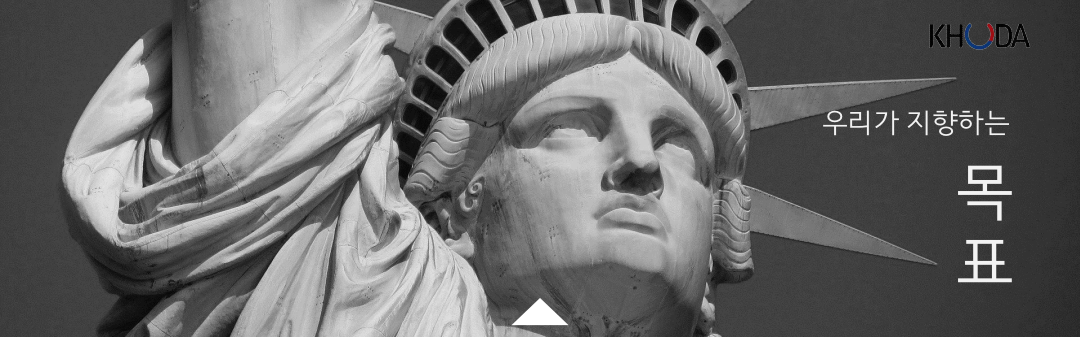


유사도 지표

User Based CF

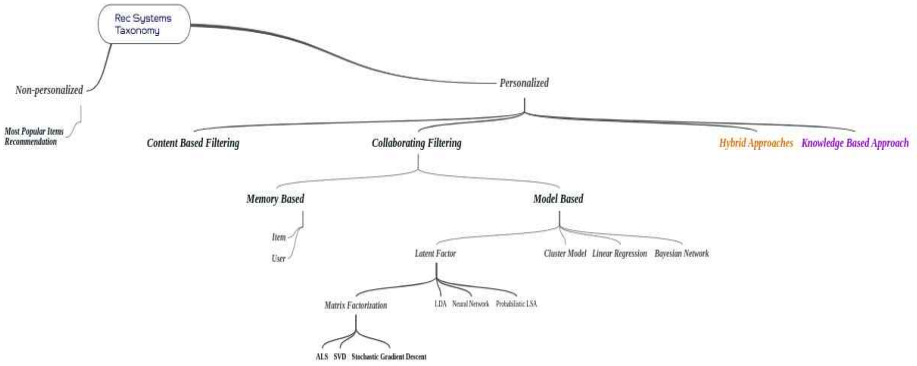
Item Based CF

Bayesian Network CF


KHUDA
우리가 지향하는
목
표

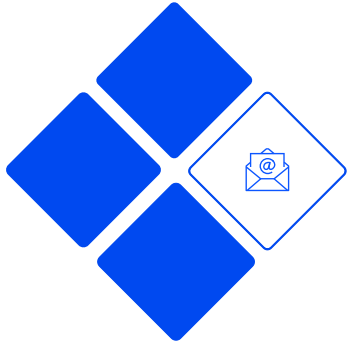
03 추천 시스템(Recommendation System)

KHUDA



03-1 유사도 지표

KHUDA



유사도 함수

유사도 함수는 자신과 가장 비슷한 컨텐츠를 찾아줄 때 사용되며 이를 통해 찾은 비슷한 컨텐츠가 사용자에게 추천되어진다.

1) 유클리디안 유사도

$$L_2=\sqrt{(p_1-q_1)^2 + (p_2-q_2)^2 + \cdots + (p_n-q_n)^2} = \sqrt{\sum_{i=1}^{n}(p_i-q_i)^2}$$

2) 코사인 유사도

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

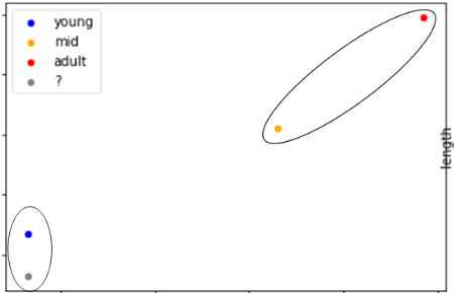
03-1 유사도 지표

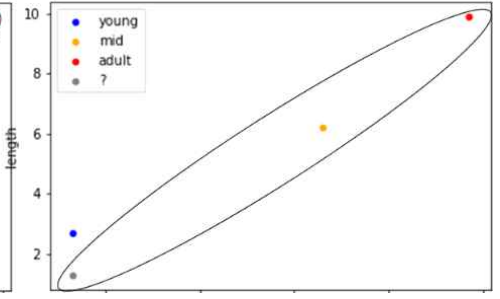
KHUDA

	유클리디안 유사도	코사인 유사도
특징	거리를 이용하여 유사도를 계산	각도를 이용하여 유사도 계산
장점	계산하기가 쉽다	1.다양한 차원에도 적용이 가능하다. 2. 스케일이 커도 사용이 가능하다.
단점	분포가 다르거나 범위가 다른 경우에 상관성이 왜곡될 수 있다	1.상호 상관관계를 가지는 feature(키, 몸무게 등)를 갖는 원소들 간의 유사도를 계산할 때 좋지 못하다. 2.벡터의 크기가 중요한 경우 잘 작동하지 않는다.

03-1 유사도 지표

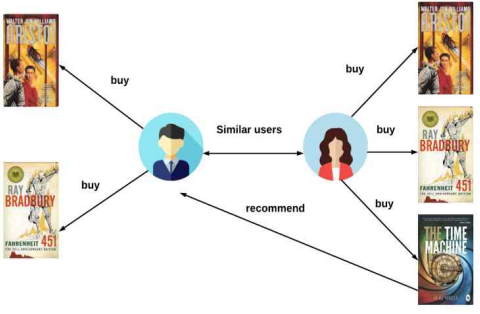
KHUDA





03-2 User-based Collaborative Filtering

KHUDA



유저 기반 추천시스템

1. 특정 유저에 대해 비슷한 성향을 가진 다른 유저들을 찾는다.
2. 해당 유저들이 높게 평가한 책들을 추천한다.

03-2 User-based Collaborative Filtering

KHUDA

1. 모델 Logic

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

$$w_{uv} = \frac{\sum_{i \in I} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum (r_{ui} - \bar{r}_u)^2} \sqrt{\sum (r_{vi} - \bar{r}_v)^2}}$$

유저간 유사도 측정 (cosine similarity)

u의 평균 별점

v는 u와의 유사도가 높은 유저들

$$S(u,i) = \bar{r}_u + \frac{\sum_{v \in V} (r_{vi} - \bar{r}_v) \cdot w_{uv}}{\sum_{v \in V} w_{uv}}$$

유저 u의 책 i에 대한 별점(score) 예측

03-2 User-based Collaborative Filtering

KHUDA

2. 데이터 전처리

1. books 데이터와 ratings 데이터를 합친 후 평점이 100개 이하인 책들과 평가를 10개 이하로 한 유저들은 drop한다.

```
#평점이 100개 이하로 있는 책들은 삭제하자!
rare_books = rating_counts[rating_counts['Book-Title'] < 100].index
common_books = df[~df['Book-Title'].isin(rare_books)]
```

```
#책 평점을 10개 이하로 남긴 사람을 드랍하자
common_books = common_books[common_books["User-ID"].map(common_books["User-ID"].value_counts()) >= 10]
```

2. 모델 훈련 및 평가를 위해 train_set과 test_set으로 나누었다.

```
# common_books를 train_set과 test_set으로 나누기 (User-ID가 균등하게 분포하도록 설정)
from sklearn.model_selection import train_test_split
train_set, test_set = train_test_split(common_books, stratify=common_books["User-ID"], random_state=42)
```

03-2 User-based Collaborative Filtering

KHUDA

3. 유저 별 평가 경향성 파악

1. 유저 별 평점의 평균값을 구하고,

2. 평가 경향성을 파악하기 위해 평점 편차('작품에 대한 유저의 평점 - 해당 유저의 평점 평균')을 구한다.

> 모든 책에 대한 각 유저의 평점 편차가 저장된 데이터프레임
Cf) 평가를 하지 않은 책에 대해서는 각 작품 별 평균 평점으로 대체

Book-ID	5	18	26	27	28	37	38	39	47	52	...
User-ID											
254.0	0.177809	-0.295633	-3.917988	0.009706	-0.17828	0.977788	0.889088	-0.620435	-1.058612	-0.33729	...
638.0	0.177809	-0.295633	-3.917988	0.009706	-0.17828	0.977788	0.889088	-0.620435	-1.058612	-0.33729	...
2766.0	0.177809	-0.295633	-2.000000	0.009706	-0.17828	2.000000	0.889088	-0.620435	-1.058612	-0.33729	...
4017.0	0.177809	-0.295633	-3.917988	0.009706	-0.17828	0.977788	0.889088	-0.620435	-1.058612	-0.33729	...
6242.0	0.177809	-0.295633	-3.917988	0.009706	-0.17828	0.352941	0.889088	-0.620435	-1.058612	-0.33729	...

03-2 User-based Collaborative Filtering

KHUDA

4. 유저 간 유사도 측정

```
from sklearn.metrics.pairwise import cosine_similarity
cosine = cosine_similarity(final_book)
np.fill_diagonal(cosine, 0)
similarity_between_users = pd.DataFrame(cosine, index=final_book.index)
similarity_between_users.columns = final_book.index
similarity_between_users
```

책에 대한 평점 편차를 바탕으로 유저 간의 유사도를 나타낸 행렬(cosine similarity).

User-ID	254.0	638.0	2766.0	4017.0	6242.0	6251.0	6543.0	6563.0	6575.0	7158.0	...
User-ID											
254.0	0.000000	0.890344	0.870728	0.766577	0.857518	0.793591	0.874476	0.647815	0.617959	0.825949	...
638.0	0.890344	0.000000	0.893914	0.808073	0.914362	0.832915	0.898097	0.655059	0.586762	0.842624	...
2766.0	0.870728	0.893914	0.000000	0.774586	0.851169	0.818501	0.880109	0.654732	0.612959	0.831280	...
4017.0	0.766577	0.808073	0.774586	0.000000	0.717854	0.718142	0.774422	0.562956	0.539538	0.730763	...
6242.0	0.857518	0.914362	0.851169	0.717854	0.000000	0.800862	0.876740	0.643025	0.606766	0.816993	...
...	...	...	...	...	...	...	...	...	...	...	...

03-2 User-based Collaborative Filtering

KHODA

5-1. 유사도가 가장 높은 15명의 유저 추출

```
def find_n_neighbors(df, n):
    order = np.argsort(df.values, axis=1)
    df = df.apply(lambda x: pd.Series(x.sort_values(ascending=False).iloc[:n].index,
                                      index=['top{}'.format(i) for i in range(1, n+1)]), axis=1)
    return df
```

```
# 가장 가까운 유저 top 15
sim_user_15_u = find_n_neighbors(similarity_between_users, 15)
sim_user_15_u
```

	top1	top2	top3	top4	top5	top6	top7	top8	top9	top10	top11	top12	top13	top14	top15
User-ID															
254.0	217318.0	165758.0	124876.0	129084.0	144531.0	269566.0	182459.0	133706.0	100846.0	99720.0	102702.0	41084.0	173679.0	66680.0	88229.0
638.0	217318.0	236283.0	100846.0	144531.0	102702.0	165758.0	88229.0	269566.0	99720.0	133706.0	41084.0	66680.0	85426.0	182993.0	160819.0
2766.0	237271.0	144531.0	217318.0	66680.0	88229.0	165758.0	93047.0	182459.0	133706.0	269566.0	102702.0	263325.0	187145.0	259259.0	100846.0
4017.0	46443.0	160819.0	114868.0	100846.0	217318.0	144531.0	113519.0	165758.0	269566.0	85426.0	182459.0	133706.0	102702.0	222220.0	99720.0
6242.0	100846.0	21404.0	88229.0	217318.0	165758.0	85426.0	98263.0	638.0	144531.0	269566.0	182459.0	133706.0	102702.0	99720.0	142524.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

03-2 User-based Collaborative Filtering

KHUDA

5-2. 평점 예측

```
avg_user = Mean.loc[Mean['User-ID'] == user, 'Book-Rating'].values[0] # user가 주는 평균 별점
index = f.index.values.squeeze().tolist() # user와 유사도가 높은 유저들
corr = similarity_between_users.loc[user, index] # user와 index의 유사도
fin = pd.concat([f,corr], axis=1)
fin.columns = ['adg_score', 'correlation'] # fin에는 책에 대한 경향성과 유저에 대한 유사도가 저장되어 있다
fin['score'] = fin.apply(lambda x:x['adg_score'] + x['correlation'], axis=1)
nume = fin['score'].sum() # 수식의 분자
deno = fin['correlation'].sum() # 수식의 분모
final_score = avg_user + (nume/deno)
return final_score
```

```
# User-ID = 254.0 , Book-ID = 52 인 경우의 score 출력해보기
score = user_book_score(254.0, 52)
print('score (User {}, Book {}) is '.format(254.0, 52) , score)
```

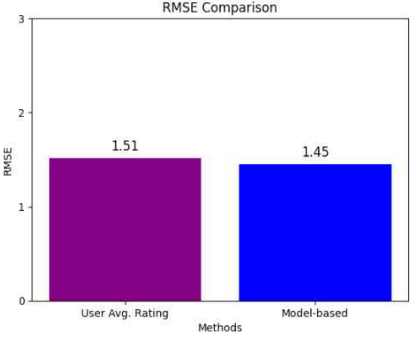
```
score (User 254.0, Book 52) is  8.486239324172299
```

03-2 User-based Collaborative Filtering

KHUDA

5-3. 모델 평가

> RMSE(평균제곱근오차)를 비교했을 때, 기존의 평균 평점을 기반으로 새로운 책을 예측했을 때의 오차는 (1.51)이었으나, 유저 기반 별점 예측을 통한 오차를 구했을 땐 (1.45)로 유의미한 차이를 보였다.

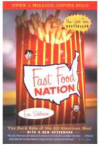


03-2 User-based Collaborative Filtering

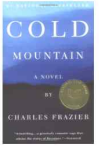
KHUDA

6. 책 추천

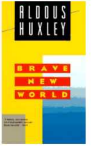
YOUR FAVORITE BOOKS



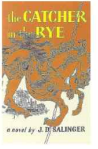
RATING: 10.0



RATING: 10.0



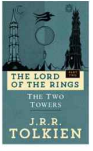
RATING: 10.0



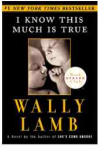
RATING: 10.0

RATING: 9.0

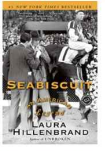
YOU MAY ALSO LIKE THESE BOOKS



RATING: 8.9



RATING: 8.9



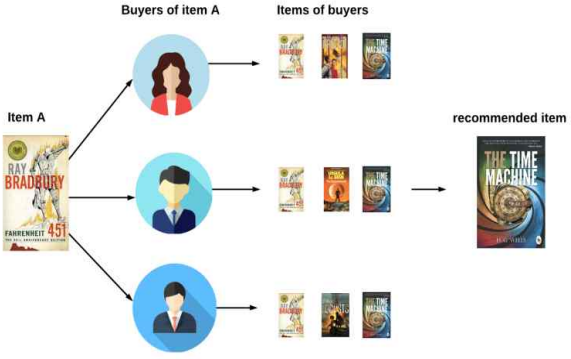
RATING: 8.9

RATING: 8.8

RATING: 8.7

03-3 Item-based Collaborative Filtering

KHUDA



아이템 기반 추천시스템

1. 사용자들의 상품에 매긴 평가 패턴을 바탕으로 아이템 간의 유사도를 계산한다.
2. 사용자의 특정 아이템에 대한 예측 평점을 계산하여 유사한 제품을 추천한다.

03-3 Item-based Collaborative Filtering

KHUDA

1-1. 데이터 전처리

```
df=books_data.copy()
df.dropna(inplace=True)
df.reset_index(drop=True,inplace=True)
df.drop(columns=["ISBN","Year-Of-Publication","Image-URL-S","Image-URL-M"],axis=1,inplace=True)
df.drop(index=df[df["Book-Rating"]==0].index,inplace=True)
df["Book-Title"]=df["Book-Title"].apply(lambda x: re.sub("[₩W_]+"," ",x).strip())
df.head()
```

- ✓ 결측값 제거하고 데이터 프레임의 인덱스 재설정
- ✓ 필요 없는 열 삭제
- ✓ Book-Rating이 0인 행 삭제(구매는 했지만 평점을 남기지 않은 사람)
- ✓ Book-Title열의 각 항목에서 특수 문자 및 밑줄 제거하고 공백 정리

03-3 Item-based Collaborative Filtering

KHUDA

1-2. 데이터 전처리

```
def item_based(bookTitle):
    bookTitle = str(bookTitle)

    if bookTitle in df["Book-Title"].values:
        rating_count = pd.DataFrame(df["Book-Title"].value_counts())
        rare_books = rating_count[rating_count["Book-Title"] <= 200].index
        common_books = df[~df["Book-Title"].isin(rare_books)]
```

✓ 각 책의 평가횟수를 계산하고 200개 이상의 평가를 받은 책은 ‘common_books’로, 200개 이하 평가를 받은 책은 ‘rare_books’로 분류

03-3 Item-based Collaborative Filtering

KHODA

2. ‘rare_books’인 경우의 책 추천

```
if bookTitle in rare_books:
    # 희귀 도서일 경우, 인기 있는 책 5권을 추천하고 시각화
    popular_books = rating_count[rating_count["Book-Title"] > 200].index
    popular_recommendations = pd.Series(popular_books).sample(5).values
    print("No Recommendations for this Book 😟 \n ")
    print("YOU MAY TRY THESE POPULAR BOOKS: \n ")
    for i, popular_book in enumerate(popular_recommendations):
        print("{}. {}".format(i + 1, popular_book))
```

- ✓ rare_books는 평점이 적은 책이므로 다른 도서와의 유사도를 계산하는 데 사용할 수 있는 충분한 데이터가 없을 가능성이 높아 Item based CF 진행 불가
- ✓ 입력된 책이 ‘rare_books’ 범주에 속하는 경우, 가장 유명한 책 5개 추천

03-3 Item-based Collaborative Filtering

KHUDA

3-1. ‘common books’ 인 경우의 추천

```
else:
    common_books_pivot = common_books.pivot_table(index=["User-ID"], columns=["Book-Title"], values="Book-Rating")
    title = common_books_pivot[bookTitle]

    # 코사인 유사도 계산
    cosine_sim = cosine_similarity(common_books_pivot.T.fillna(0))
    sim_scores = list(enumerate(cosine_sim))
```

- ✓ common_books인 경우 Item-based CF 진행
- ✓ 사용자와 책 제목을 각각 행과 열로 가지는 pivot table 생성
- ✓ pivot table의 코사인 유사도를 계산

03-3 Item-based Collaborative Filtering

KHUDA

3-2. ‘common books’ 인 경우의 추천

```
# 입력된 책과의 유사도를 기준으로 추천 도서 정렬
sim_scores = sorted(sim_scores, key=lambda x: x[1][bookTitle], reverse=True)
sim_scores = sim_scores[1:6]  # 입력된 책 자체를 제외하고 상위 5개 추천

book_indices = [i[0] for i in sim_scores]
recommendation_df = common_books_pivot.columns[book_indices]

less_rating = []
for i in recommendation_df:
    if df[df["Book-Title"] == i]["Book-Rating"].mean() < 5:
        less_rating.append(i)
if len(recommendation_df) - len(less_rating) > 5:
    recommendation_df = recommendation_df[~recommendation_df.isin(less_rating)]

recommendation_df = recommendation_df[0:5]
```

✓ 입력된 책과의 유사도를 기준으로 추천 도서 정렬
✓ 평균 평점이 5 미만인 책은 추천 목록에서 제외

✓ 추천 목록에서 입력된 책 자체를 제외하고 상위 5개의 책 추천

03-3 Item-based Collaborative Filtering

KHUDA

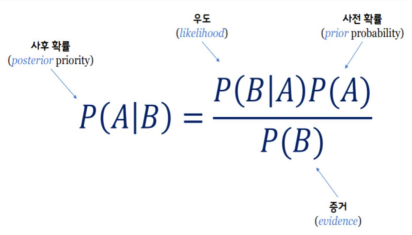
4. 추천 책 시각화

✓ 유사도가 가장 높고 좋은 평점을 가진 상위 5권의 책을 선택하여 표지 이미지와 평균 평점 시각화

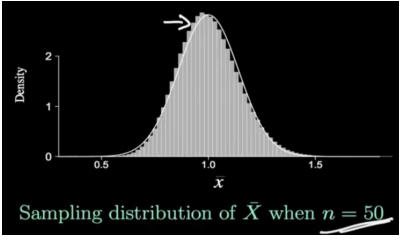
03-4 Bayesian Network Collaborative Filtering

KHUDA

1. 모델 Logic



알고리즘:**Naive-Bayes**



Clustering = CLT, Bootstraping

03-4 Bayesian Network Collaborative Filtering

KHUDA

1. 모델 Logic

나이브 베이지안을 사용하기에 여러가지 문제가 많았다. 데이터에 책 구매자가 사지않은 데이터 즉,
보고 지나친 데이터가 존재하지 않았다. 그렇기에 사용자가 봤을 만한 책 데이터를 예상해야하는데 이를
군집화를 통해 찾아보려고 했다.
여러가지 군집화 방법이 존재하지만 다른 팀원들이 이미 군집화를 진행하였기에 다른 방법을 한번 찾아보았다.

03-4 Bayesian Network Collaborative Filtering

KHUDA

1. 모델 Logic

CLT: Central Limit Theorem

중심극한정리: i.i.d. 확률변수 X1, X2의 표본평균의 분포는 정규분포에 수렴한다.

i.i.d. :**independent and identically distribution**

1)independent(독립적이고), 2)identically distribution(같은 확률분포를)

여기서 역으로 생각해 표본평균들이 정규분포로 근사화 된다면 동일한 분포에서 왔다고 할 수 있지 않을까?

03-4 Bayesian Network Collaborative Filtering

KHUDA

2. 군집 개수 정하고 샘플링 진행

- 군집의 개수는 임의대로 4개로 정하였다.

```
    X_0, X_1= train_test_split(merge_encoding_data_sum_0, test_size = 0.5, random_state = random_state[i])
    X_0_0, X_0_1 = train_test_split(X_0,test_size = 0.5, random_state = random_state[i])
    X_1_0, X_1_1 = train_test_split(X_1, test_size = 0.5, random_state = random_state[i])


a_0 = np.array(X_0_0)
a_1 = np.array(X_0_1)
a_2 = np.array(X_1_0)
a_3 = np.array(X_1_1)
result = np.vstack((a_0, a_1,a_2,a_3))

for j in range(4):

    storige = []
    data = result[j]
    for k in range(1000):
        storige.append(sum(np.random.choice(data,size=len(data), replace = True))/len(data))
    n_test = stats.normaltest(storige)
    list_n_test = list(n_test)
    list_n_test.append(j)
    new_list.append(list_n_test)
new_list_1.append(new_list)
```

03-4 Bayesian Network Collaborative Filtering

KHUDA

3. 데이터셋 선택

p_value가 0.5보다 작은 데이터 셋은 날리고 전체 p_value가 가장낮은 데이터셋 선택

	0	1	2	3
1	[12.849174206583179, 0.0016212025289418963, 0]	[2.106179715821261, 0.348858159936335, 1]	[2.21264167936149, 0.3307736945570026, 2]	[7.497128744286188, 0.02355153283405207, 3]
5	[2.3835433958059786, 0.3036827530713745, 0]	[1.7380099963225555, 0.41936861426529715, 1]	[2.0614250324789194, 0.3567526779131605, 2]	[1.9793381315466805, 0.37169967881624666, 3]
54	[3.178982047518598, 0.2040294314440981, 0]	[6.3125246850048375, 0.0425846106848836, 1]	[2.379551013442231, 0.3042895673623093, 2]	[2.3363421148036063, 0.3109351039955705, 3]
61	[2.7502955952048938, 0.25280222948007747, 0]	[5.410277028151129, 0.0668610620096586, 1]	[2.280541436924878, 0.3197324526209274, 2]	[4.5115140055796627, 0.10479418421258108, 3]
67	[4.451683139973877, 0.10797651059684384, 0]	[1.437467453359877, 0.48736900776093794, 1]	[3.5237904443205803, 0.17171910937060308, 2]	[0.01144660756227673 3, 0.9942930431209833, 3]
79	[9.562395835430168, 0.00838594727334575, 0]	[7.461303037190769, 0.023977209149097102, 1]	[7.124851735785665, 0.02836991949367284, 2]	[2.0878937726415923, 0.35206238593402445, 3]

03-4 Bayesian Network Collaborative Filtering

KHUDA

4. 모델링

군집을 만든 후 해당 군집에 존재하는 책들은 보았지만 구매하지 않았다 라고 판단

```
def NB(User_id):
    a = filtered_data.loc[int(User_id),:]['Book-Author_encoded']

    a_unique = a.unique()
    for i in range(len(a_unique)):
        print(f"{User_id}가 읽은 책의 저자:{a_unique[i]}")
        b = filtered_data[filtered_data['Book-Author_encoded'] == a_unique[i]]
        prob = ((pd.DataFrame(a.value_counts()).loc[a_unique[i],:][0])/len(a))/(((pd.DataFrame(a.value_counts()).lo
        print(f"위 저자의 책을 읽을 NB확률{prob}")
```

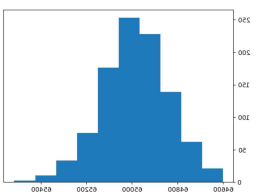
```
NB(276746)
```

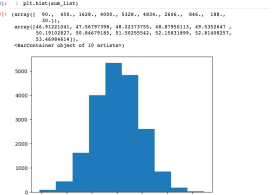
```
276746가 읽은 책의 저자:8
위 저자의 책을 읽을 NB확률0.991217370090266
276746가 읽은 책의 저자:9
위 저자의 책을 읽을 NB확률0.8493723849372385
276746가 읽은 책의 저자:10
위 저자의 책을 읽을 NB확률0.9858267716535433
276746가 읽은 책의 저자:11
위 저자의 책을 읽을 NB확률0.3571428571428571
276746가 읽은 책의 저자:12
위 저자의 책을 읽을 NB확률0.4929577464788732
```

03-4 Bayesian Network Collaborative Filtering

KHUDA

5. 문제점





04 Collaborative Filtering recommendation system

KHUDA

※ 결론 및 내용정리

	User-based CF	Item-based CF	Bayesian Network CF
특징	User의 유사도 기반 추천	Item의 유사도 기반 추천	Bayesian을 활용한 추천
장점	직관적인 해석 가능, 아이템 데이터에 대한 상세 정보 없이도 추천 제공 가능	Coverage가 높음(사용자가 아이템 하나면 평가해도 추천 가능)	확률을 통해 설명함으로 결과의 설명력이 높음
단점	사용자-아이템 데이터가 희소한 경우 정확한 추천이 어려움	과거 아이템에 의존해 추천하기 때문에 새로운 아이템(ex: 신간)이 추천되기 어려움	모든 사건이 독립이라는 나이브한 가정을 가지고 있음
한계	Cold-start 문제가 존재 -충분한 데이터가 없다면 좋은 추천이 어려움 -유저에 대한 기록이나 새로운 아이템에 대한 정보가 없다면 추천이 불가능		첫 모델 학습 시 시간과 연산량이 많이 소요