무신사 리뷰 기반 옷 추천시스템

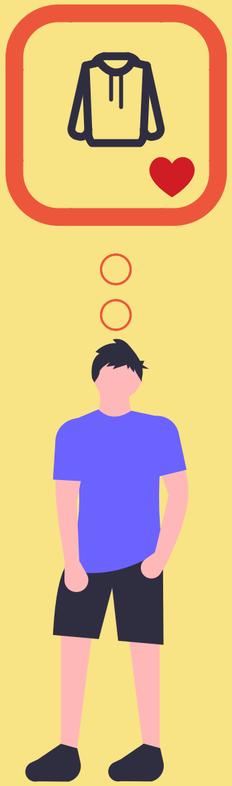
8기 B-DAY 컨퍼런스

추천시스템 1조 김진호 임성연 조유진

CONTENTS

01

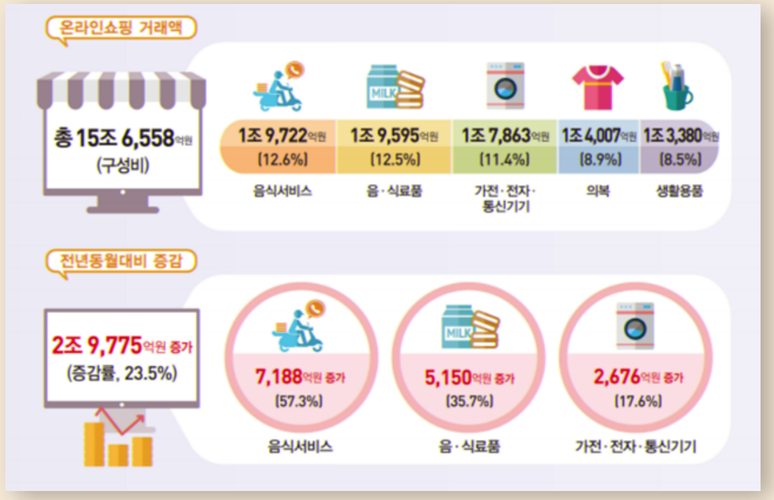
주제 선정



위클리오늘 | 2020.08.05.

[위클리오늘] **코로나19**에 **폭증**하는 **온라인 쇼핑**...'집콕 소비'가 대...

실제로 **코로나19** 여파가 가시화된 지난 3월 이후 4달 연속 **온라인쇼핑** 거래액이 1 2조 원을 상회했다. 이로 인해 올해 상반기 **온라인쇼핑** 거래액은 74조3544억 원...



| 2021년 6월 온라인 쇼핑 동향

| 2022년 6월 온라인 쇼핑 동향

코로나19로 인한 온라인 쇼핑 증가

- 유튜브 이용자 시청 시간의 70%가 추천 알고리즘에 의해 발생
- 아마존 전체 매출 중 약 35%가 협업 필터링 알고리즘에 의해 발생


AI타임스 | 2021.11.18.

"성장하는 스타트업은 AWS와 함께"...와디즈·무신사, AWS와의 AI...

(출처=AWS 기자간담회 캡처) 무신사, AI 추천서비스로 클릭율 개선과 업무 자동화 이뤄 패션 이커머스 플랫폼 기업인 무신사 역시 이번 간담회에서 AWS의 AI 서비...

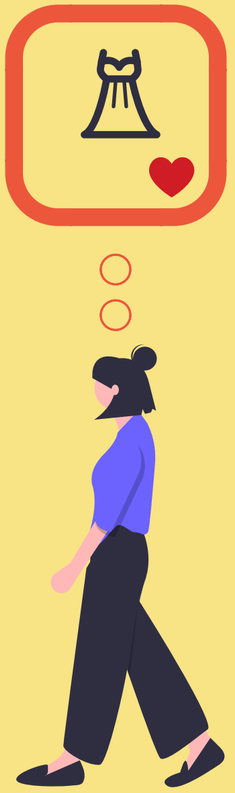
- 패션 분야의 추천 시스템 도입
- 딥러닝을 통해 개인 맞춤형으로 진화

사용자 리뷰 기반 옷 추천시스템

02

데이터 소개

- 데이터 선정
- 데이터 수집
- 데이터 전처리

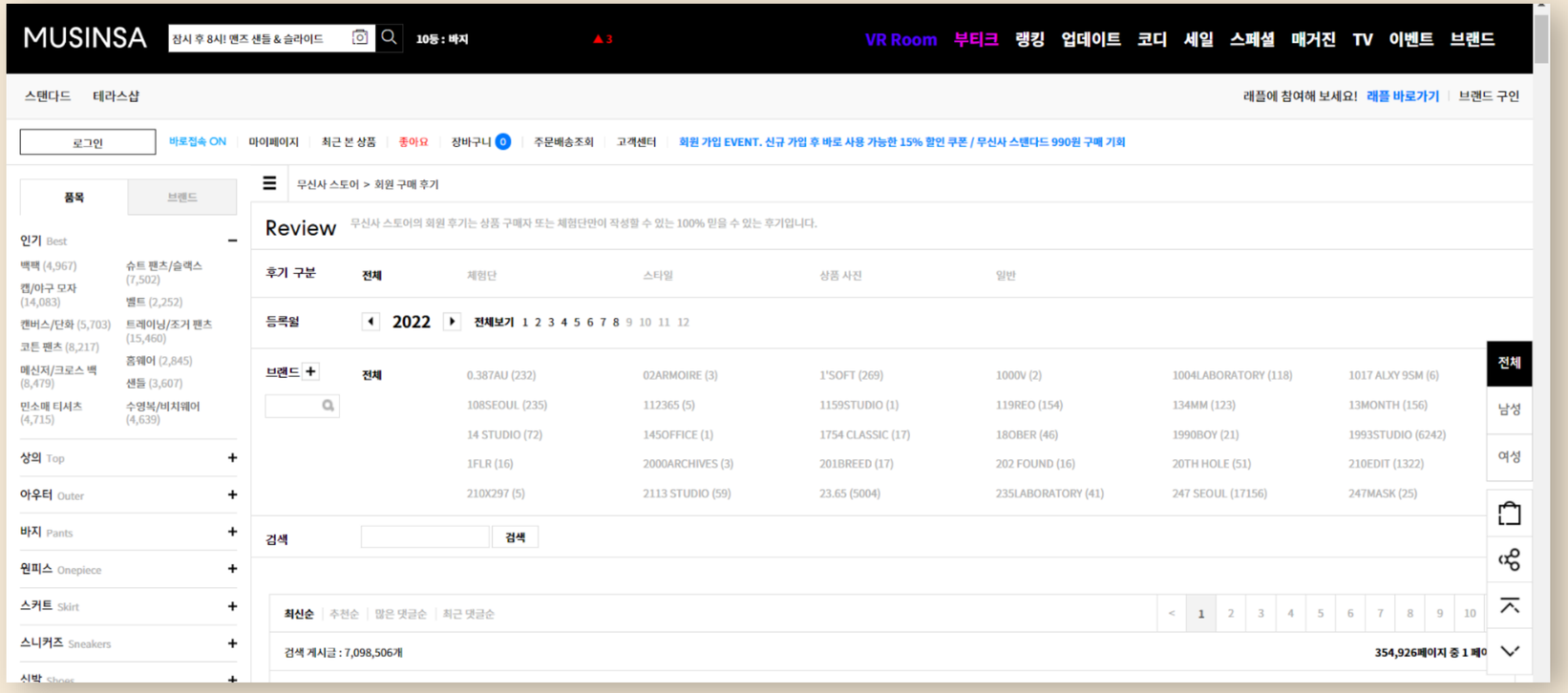


MZ세대 10명 중 9명이
온라인 채널을 통해 소비

인지도와 이용 경험이
가장 높은 플랫폼

많은 리뷰 데이터

무신사 리뷰 데이터 사용



유저명, 아이템명, 평점 데이터 크롤링

유저명	아이템명	평점
LV 5 Aa177-1	쓰리 턱 세미 벌룬 슬랙스 [블랙]	5
LV 5 Aa177-1	나일론 2웨이 와이드 커브 핏 밴딩 팬츠 라이트 그레이	5
LV 5 Aa177-1	OVERFIT STRIPE SHIRT [LIGHT YELLOW]	5
LV 5 Aa177-1	원턱 코튼 볼륨 팬츠 (네이비)	4
LV 5 Aa177-1	원턱 코튼 볼륨 팬츠 (네이비)	5
...	...	...
LV 6 shinmp	51007 HISHITOMO NATURAL CREAM JEANS [RELAX STR...	5
LV 6 shinmp	홀가먼트 모크넥_블랙	5
LV 6 shinmp	White Space - MOD7 crop	4
LV 6 shinmp	Punk Town - MOD4 crop	5

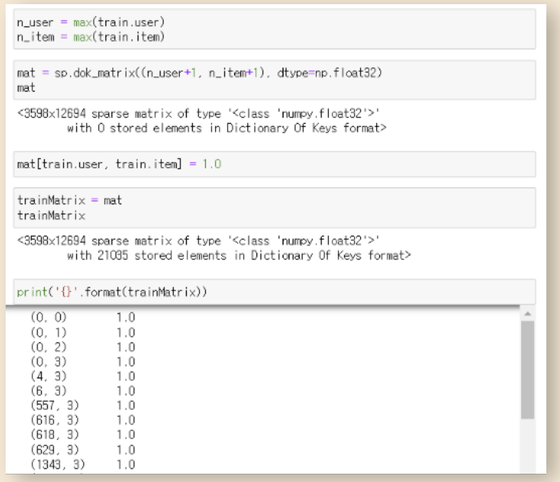
- 중복 데이터 제거 진행
- LV 5 이상의 user 데이터 사용
- user 별 특정 개수 이상의 리뷰 데이터 사용

```
a =  pd.DataFrame({'user' : np.arange(0,3598), '유저명' : df.유저명.unique()})
b =  pd.DataFrame({'item' : np.arange(0,12694), '아이템명' : df.아이템명.unique()})
```

```
df = df.merge(a, on = '유저명')
df = df.merge(b, on = '아이템명')
df = df[['user', 'item']]
df
```

	user	item
0	0	0
1	0	1
2	0	2
3	0	3
4	4	3
...	...	...
24628	3596	12689

- 데이터 라벨링 진행
- 라벨링 된 데이터로 데이터 셋 생성



```
n_user = max(train.user)
n_item = max(train.item)

mat = sp.dok_matrix((n_user+1, n_item+1), dtype=np.float32)
mat

<3598x12694 sparse matrix of type '<class 'numpy.float32'>'
        with 0 stored elements in Dictionary Of Keys format>

mat[train.user, train.item] = 1.0

trainMatrix = mat
trainMatrix

<3598x12694 sparse matrix of type '<class 'numpy.float32'>'
        with 21035 stored elements in Dictionary Of Keys format>

print('{}'.format(trainMatrix))

  (0, 0)        1.0
  (0, 1)        1.0
  (0, 2)        1.0
  (0, 3)        1.0
  (4, 3)        1.0
  (6, 3)        1.0
  (557, 3)      1.0
  (616, 3)      1.0
  (618, 3)      1.0
  (629, 3)      1.0
  (1343, 3)     1.0
```

trainMatrix

- 각 user가 구매한 item 하나를 제외한 모든 user-item 행렬
- sparse matrix 형태

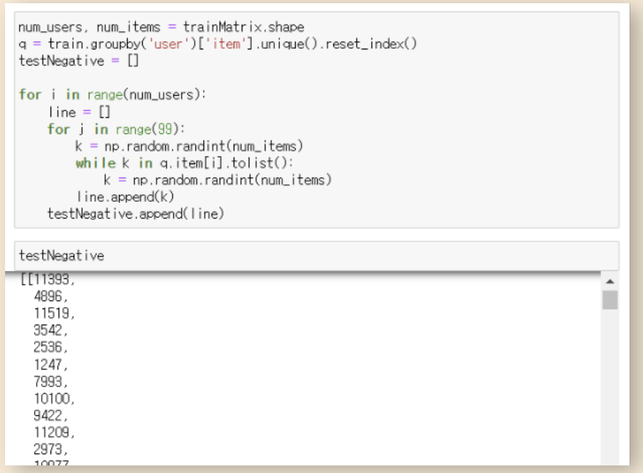
```
testRatings = (test.values.tolist())

testRatings

[16, 108],
[17, 114],
[18, 120],
[19, 197],
[20, 278],
[21, 286],
[22, 293],
[23, 297],
[24, 302],
[25, 312],
[26, 324],
[27, 332],
[28, 338],
[29, 345],
[30, 352],
[31, 358],
[32, 356],
```

testRatings

- user가 구매한 item 한 개
- 리스트 형태



```
num_users, num_items = trainMatrix.shape
g = train.groupby('user')['item'].unique().reset_index()
testNegative = []

for i in range(num_users):
    line = []
    for j in range(99):
        k = np.random.randint(num_items)
        while k in g.item[i].tolist():
            k = np.random.randint(num_items)
        line.append(k)
    testNegative.append(line)

testNegative

[[11399,
  4896,
  11519,
  3542,
  2536,
  1247,
  7993,
  10100,
  9422,
  11209,
  2973,
```

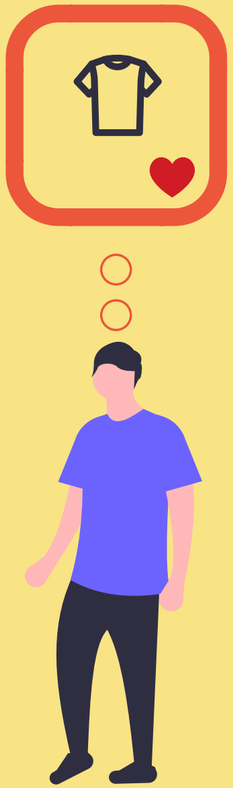
testNegative

- 각 user별 구매한 item을 제외한 99개의 item을 random negative sampling

03

추천시스템

- GMF
- MLP
- NCF



Neural Collaborative Filtering

- Collaborative Filtering, 즉 user와 item feature 간의 상호작용에 집중
- GMF와 MLP의 장점을 모두 살려 높은 레벨의 학습 진행 가능

Learning from Implicit Data

Implicit feedback의 장단점

- 장점 : 자동적으로 데이터가 수집되고 데이터가 풍부

- 단점 : negative feedback의 부재

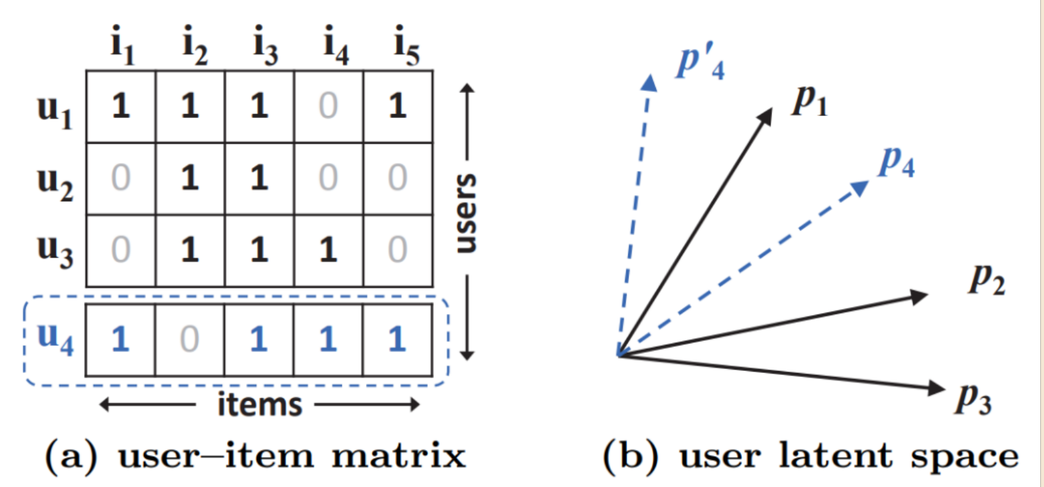
$$y_{ui} = \begin{cases} 1, & \text{if interaction (user } u\text{, item } i\text{) is observed;} \\ 0, & \text{otherwise.} \end{cases}$$

M, N : user와 item의 수

rating 데이터가 아닌 0과 1의 binary 데이터

positive vs. negative (X)
observed vs.unobserved (O)

Matrix Factorization



- User-Item Interaction Matrix의 합계점을 지적
- User와 item의 복잡한 관계를 low dimension에 표현하면서 문제 발생
- Dimension 크기를 키우면 overfitting 발생
- Dimension의 크기를 키우는 것이 아닌 user와 item factor 간의 상관관계를 풍부하게 표현

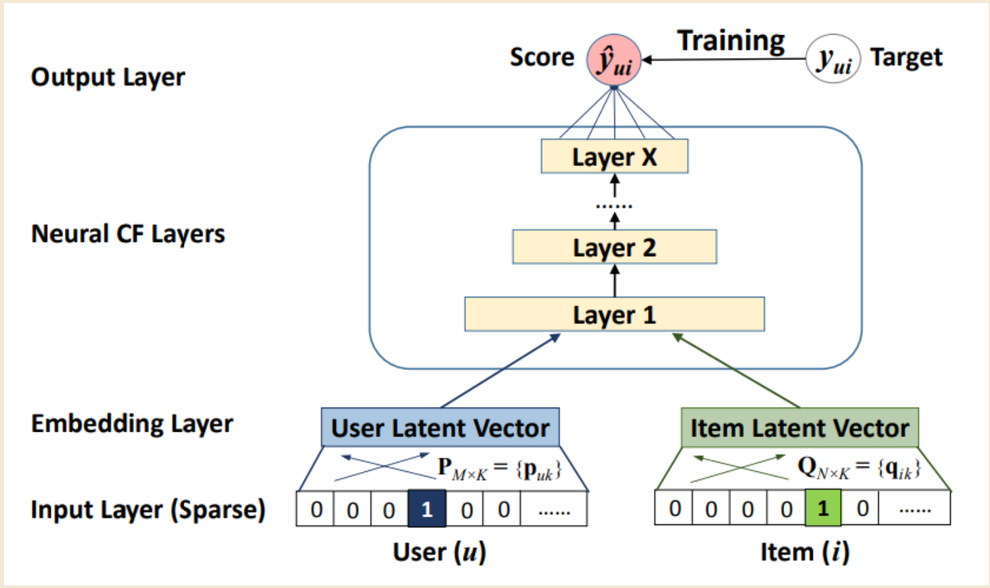
Neural Collaborative Filtering Framework

Input Layer(Sparse)
- user와 item을 one-hot vector로 표현

Embedding layer
- sparse one-hot vector를 통해 dense vector로 맵핑

Neural CF Layers
- user latent vector와 item latent vector를 concat하여 layer를 통과

Output Layer
- user u와 item i의 상관관계를 0과 1 사이의 점수로 나타낸다



Neural Collaborative Filtering Framework

Likelihood Function

$$p(\mathcal{Y}, \mathcal{Y}^- | \mathbf{P}, \mathbf{Q}, \Theta_f) = \prod_{(u,i) \in \mathcal{Y}} \hat{y}_{ui} \prod_{(u,j) \in \mathcal{Y}^-} (1 - \hat{y}_{uj}).$$

Negative Logarithm of the Likelihood

$$\begin{aligned} L &= - \sum_{(u,i) \in \mathcal{Y}} \log \hat{y}_{ui} - \sum_{(u,j) \in \mathcal{Y}^-} \log(1 - \hat{y}_{uj}) \\ &= - \sum_{(u,i) \in \mathcal{Y} \cup \mathcal{Y}^-} y_{ui} \log \hat{y}_{ui} + (1 - y_{ui}) \log(1 - \hat{y}_{ui}). \end{aligned}$$

- Label이 binary이기 때문에, 베르누이 분포를 사용
- Y는 y(u, i)=1인 집단, Y-는 y(u, i)=0인 집단
- Loss function은 binary cross entropy 사용
- L을 최소화하는 파라미터 탐색
- 학습은 SGD를 사용

Generalized Matrix Factoriaztion

$$\phi_1(\mathbf{p}_u, \mathbf{q}_i) = \mathbf{p}_u \odot \mathbf{q}_i,$$

$$\hat{y}_{ui} = a_{out}(\mathbf{h}^T(\mathbf{p}_u \odot \mathbf{q}_i)),$$

- 각 element의 weight를 학습

p, q : user와 item의 latent vector
a : non-linear activation function (sigmoid)
h^T : 내적할 때 가중치 역할

- non-linar activation function을 사용하여 user-item interaction을 더 풍부하게 표현

- y hat과 actual y 값 사이의 loss를 줄이는 방향으로 모델을 학습

- a가 1이고 h^T가 uniform vector이면 Matrix Factorization

Multi-Layer Perceptron

$$\mathbf{z}_1 = \phi_1(\mathbf{p}_u, \mathbf{q}_i) = \begin{bmatrix} \mathbf{p}_u \\ \mathbf{q}_i \end{bmatrix},$$
$$\phi_2(\mathbf{z}_1) = a_2(\mathbf{W}_2^T \mathbf{z}_1 + \mathbf{b}_2),$$
$$\ldots\ldots$$
$$\phi_L(\mathbf{z}_{L-1}) = a_L(\mathbf{W}_L^T \mathbf{z}_{L-1} + \mathbf{b}_L),$$
$$\hat{y}_{ui} = \sigma(\mathbf{h}^T \phi_L(\mathbf{z}_{L-1})),$$

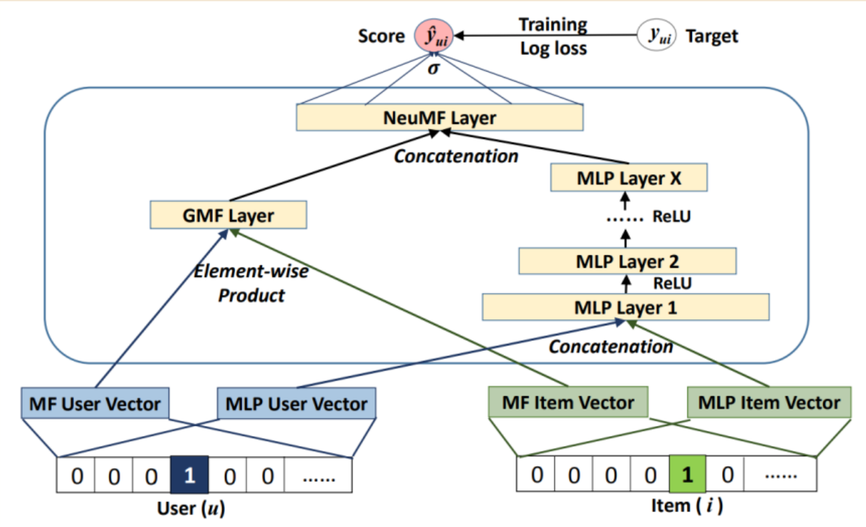
- GMF보다 더 쉽게 user-item interaction을 학습

Wx : weight matrix
bx : bias vector
ax : x번째 층 activation function

φ1 : user와 item의 latent vector를 concat
φL : weight matrix와 bias vector로 이루어진 식

- GMF와 동일한 구조

Fusion of GMF and MLP



- GMF와 MLP가 같은 embedding을 공유하지 않고 분리하여 각각 최적화
- Weight를 사용하여 GMF와 MLP의 output을 concat
- MF의 linearity와 MLP의 non linearity를 결합하여 장점만 선택
- GMF와 MLP의 장점을 모두 살린 네트워크 구조 사용

$$\phi^{GMF} = \mathbf{p}_u^G \odot \mathbf{q}_i^G,$$

$$\phi^{MLP} = a_L(\mathbf{W}_L^T(a_{L-1}(...a_2(\mathbf{W}_2^T \begin{bmatrix} \mathbf{p}_u^M \\ \mathbf{q}_i^M \end{bmatrix} + \mathbf{b}_2)...)) + \mathbf{b}_L),$$

$$\hat{y}_{ui} = \sigma(\mathbf{h}^T \begin{bmatrix} \phi^{GMF} \\ \phi^{MLP} \end{bmatrix}),$$

04 모델링

모델링 01

```python
def get_model(num_users, num_items, layers = [20,10], reg_layers=[0,0]):
    assert len(layers) == len(reg_layers)
    num_layer = len(layers) #Number of layers in the MLP
    # Input variables
    user_input = Input(shape=(1,), dtype='int32', name = 'user_input')
    item_input = Input(shape=(1,), dtype='int32', name = 'item_input')

    MLP_Embedding_User = Embedding(input_dim = num_users, output_dim = int(layers[0]/2), name = 'user_embedding',
                                   embeddings_initializer='uniform', embeddings_regularizer = l2(reg_layers[0]), input_length=1)
    MLP_Embedding_Item = Embedding(input_dim = num_items, output_dim = int(layers[0]/2), name = 'item_embedding',
                                   embeddings_initializer='uniform', embeddings_regularizer = l2(reg_layers[0]), input_length=1)

    # Crucial to flatten an embedding vector!
    user_latent = Flatten()(MLP_Embedding_User(user_input))
    item_latent = Flatten()(MLP_Embedding_Item(item_input))

    # The 0-th layer is the concatenation of embedding layers
    #vector = merge([user_latent, item_latent], mode = 'concat')
    vector = Concatenate()([user_latent, item_latent]) # NOTE: the layer is first constructed and then it's called on its input


    # MLP layers
    for idx in range(1, num_layer):
        layer = Dense(layers[idx], kernel_regularizer= l2(reg_layers[idx]), activation='relu', name = 'layer%d' %idx)
        vector = layer(vector)

    # Final prediction layer
    prediction = Dense(1, activation='sigmoid', kernel_initializer='lecun_uniform', name = 'prediction')(vector)

    model = Model(inputs=[user_input, item_input],
                  outputs=prediction)

    return model
```

| MLP 코드

```python
def get_model(num_users, num_items, latent_dim, regs=[0,0]):
    # Input variables
    user_input = Input(shape=(1,), dtype='int32', name = 'user_input')
    item_input = Input(shape=(1,), dtype='int32', name = 'item_input')

    MF_Embedding_User = Embedding(input_dim = num_users, output_dim = latent_dim, name = 'user_embedding',
                                  embeddings_initializer='uniform', embeddings_regularizer = l2(regs[0]), input_length=1)
    MF_Embedding_Item = Embedding(input_dim = num_items, output_dim = latent_dim, name = 'item_embedding',
                                  embeddings_initializer='uniform', embeddings_regularizer = l2(regs[1]), input_length=1)

    # Crucial to flatten an embedding vector!
    user_latent = Flatten()(MF_Embedding_User(user_input))
    item_latent = Flatten()(MF_Embedding_Item(item_input))

    # Element-wise product of user and item embeddings
    #predict_vector = merge([user_latent, item_latent], mode = 'mul')
    predict_vector = Concatenate()([user_latent, item_latent]) # NOTE: the layer is first constructed and then it's called on its input

    # Final prediction layer
    #prediction = Lambda(lambda x: K.sigmoid(K.sum(x)), output_shape=(1,))(predict_vector)
    prediction = Dense(1, activation='sigmoid', kernel_initializer='lecun_uniform', name = 'prediction')(predict_vector)

    model = Model(inputs=[user_input, item_input],
                  outputs=prediction)

    return model
```

| GMF 코드

```python
def get_model(num_users, num_items, mf_dim=10, layers=[10], reg_layers=[0], reg_mf=0):
    assert len(layers) == len(reg_layers)
    num_layer = len(layers) #Number of layers in the MLP
    # Input variables
    user_input = Input(shape=(1,), dtype='int32', name = 'user_input')
    item_input = Input(shape=(1,), dtype='int32', name = 'item_input')

    # Embedding layer
    MF_Embedding_User = Embedding(input_dim = num_users, output_dim = mf_dim, name = 'mf_embedding_user',
                                  embeddings_initializer='uniform', embeddings_regularizer = l2(reg_mf), input_length=1)

    MF_Embedding_Item = Embedding(input_dim = num_items, output_dim = mf_dim, name = 'mf_embedding_item',
                                  embeddings_initializer='uniform', embeddings_regularizer = l2(reg_mf), input_length=1)

    MLP_Embedding_User = Embedding(input_dim = num_users, output_dim = int(layers[0]/2), name = "mlp_embedding_user",
                                   embeddings_initializer='uniform', embeddings_regularizer = l2(reg_layers[0]), input_length=1)
    MLP_Embedding_Item = Embedding(input_dim = num_items, output_dim = int(layers[0]/2), name = 'mlp_embedding_item',
                                   embeddings_initializer='uniform', embeddings_regularizer = l2(reg_layers[0]), input_length=1)

    # MF part
    mf_user_latent = Flatten()(MF_Embedding_User(user_input))
    mf_item_latent = Flatten()(MF_Embedding_Item(item_input))
    mf_vector = Multiply()([mf_user_latent, mf_item_latent]) # NOTE: the layer is first constructed and then it's called on its input
    #mf_vector = merge([mf_user_latent, mf_item_latent], mode = 'mul') # element-wise multiply

    # MLP part
    mlp_user_latent = Flatten()(MLP_Embedding_User(user_input))
    mlp_item_latent = Flatten()(MLP_Embedding_Item(item_input))
    mlp_vector = Concatenate()([mlp_user_latent, mlp_item_latent]) # NOTE: the layer is first constructed and then it's called on its input
    #mlp_vector = merge([mlp_user_latent, mlp_item_latent], mode = 'concat')
    for idx in range(1, num_layer):
        layer = Dense(layers[idx], kernel_regularizer = l2(reg_layers[idx]), activation='relu', name="layer%d" %idx)
        mlp_vector = layer(mlp_vector)

    # Concatenate MF and MLP parts
    #mf_vector = Lambda(lambda x: x * alpha)(mf_vector)
    #mlp_vector = Lambda(lambda x : x * (1-alpha))(mlp_vector)

    predict_vector = Concatenate()([mf_vector, mlp_vector]) # NOTE: the layer is first constructed and then it's called on its input
    #predict_vector = merge([mf_vector, mlp_vector], mode = 'concat') <- fade-out

    # Final prediction layer
    prediction = Dense(1, activation='sigmoid', kernel_initializer='lecun_uniform', name = "prediction")(predict_vector)

    model = Model(inputs=[user_input, item_input],
                  outputs=prediction)

    return model
```

| NeuMF 코드

```
df = df.query('리뷰개수 >= 5').reset_index(drop= True)
df
```

	유저명	아이템명	평점	리뷰개수
0	LV 5 june.kr	[BE]TMR-PL001 [조거팬츠]	5	9
1	LV 5 june.kr	[BK]TMR-PL001 [조거팬츠]	5	9
2	LV 5 june.kr	[OR]TMR-HL001 [후드티]	5	9
3	LV 5 june.kr	[WT]TMR-TH001 [반팔 레이어드 티셔츠]	5	9
4	LV 5 june.kr	[LBE]TMR-PL001 [조거팬츠]	5	9
...	...	...	...	...
25011	LV 5 뉴비_3d39d4dd	타이다이 오버핏 반팔 슬리브 - 블랙&베이지 (FU-140)	5	6
25012	LV 5 뉴비_3d39d4dd	Logo T-shirts (GK2TSU091GN)	5	6
25013	LV 5 뉴비_3d39d4dd	유틸리티 쇼츠 [라이트 그레이]	5	6
25014	LV 5 뉴비_3d39d4dd	벡터 오버사이즈 빅 로고 티셔츠 - 네이비 / GL1269	5	6
25015	LV 5 뉴비_3d39d4dd	밴딩 와이드 슬랙스 P-035_BG	5	6

- 2021.01 - 2022.06 까지의 리뷰 데이터
- **리뷰 개수 5개 이상**인 user만 사용
- **HR 0.2**로 낮은 성능을 보임

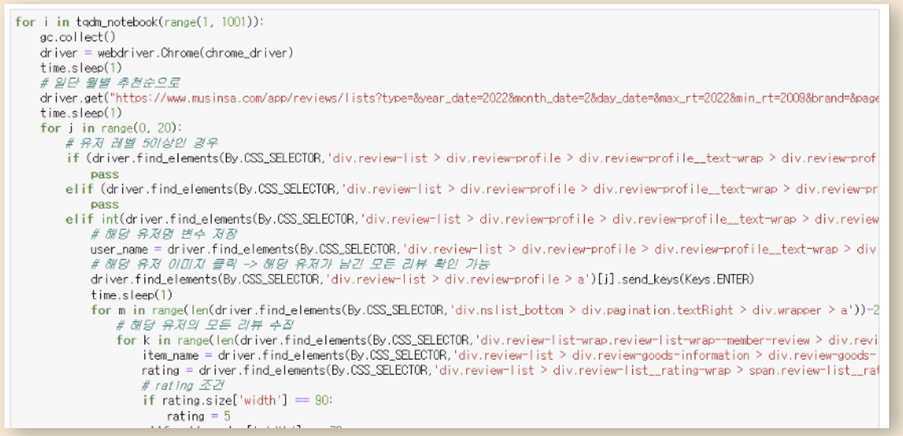
리뷰 개수 3개 이상으로 조정

```
df = df.query('리뷰개수 >= 3').reset_index(drop= True)
df
```

	유저명	아이템명	평점	리뷰개수
0	LV 5 june.kr	[BE]TMR-PL001 [조거팬츠]	5	9
1	LV 5 june.kr	[BK]TMR-PL001 [조거팬츠]	5	9
2	LV 5 june.kr	[OR]TMR-HL001 [후드티]	5	9
3	LV 5 june.kr	[WT]TMR-TH001 [반팔 레이어드 티셔츠]	5	9
4	LV 5 june.kr	[LBE]TMR-PL001 [조거팬츠]	5	9
...	...	...	...	...
25011	LV 5 뉴비_3d39d4dd	타이다이 오버핏 반팔 슬리브 - 블랙&베이지 (FU-140)	5	6
25012	LV 5 뉴비_3d39d4dd	Logo T-shirts (GK2TSU091GN)	5	6
25013	LV 5 뉴비_3d39d4dd	유틸리티 쇼츠 [라이트 그레이]	5	6
25014	LV 5 뉴비_3d39d4dd	벡터 오버사이즈 빅 로고 티셔츠 - 네이비 / GL1269	5	6

- **리뷰 개수 3개 이상**인 user만 사용
- **HR 0.3**으로 성능이 조금 좋아짐

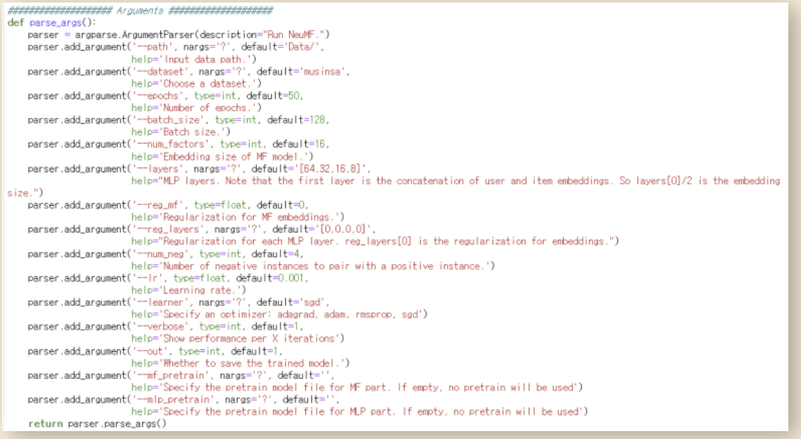
데이터를 늘리기 위한 추가 크롤링 진행



```
for i in tqdm_notebook(range(1, 1001)):
    gc.collect()
    driver = webdriver.Chrome(chrome_driver)
    time.sleep(1)
    # 일단 월별 추천순으로
    driver.get("https://www.musinsa.com/app/reviews/lists?type=&year_date=2022&month_date=2&day_date=&max_rt=2022&min_rt=2009&brand=&page
    time.sleep(1)
    for j in range(0, 20):
        # 유저 레벨 5이상인 경우
        if (driver.find_elements(By.CSS_SELECTOR,'div.review-list > div.review-profile > div.review-profile__text-wrap > div.review-prof
            pass
        elif (driver.find_elements(By.CSS_SELECTOR,'div.review-list > div.review-profile > div.review-profile__text-wrap > div.review-pr
            pass
        elif int(driver.find_elements(By.CSS_SELECTOR,'div.review-list > div.review-profile > div.review-profile__text-wrap > div.review
            # 해당 유저명 변수 저장
            user_name = driver.find_elements(By.CSS_SELECTOR,'div.review-list > div.review-profile > div.review-profile__text-wrap > div
            # 해당 유저 이미지 클릭 -> 해당 유저가 남긴 모든 리뷰 확인 가능
            driver.find_elements(By.CSS_SELECTOR,'div.review-list > div.review-profile > a')[j].send_keys(Keys.ENTER)
            time.sleep(1)
            for m in range(len(driver.find_elements(By.CSS_SELECTOR,'div.nslist_bottom > div.pagination.textRight > div.wrapper > a'))-2
                # 해당 유저의 모든 리뷰 수집
                for k in range(len(driver.find_elements(By.CSS_SELECTOR,'div.review-list-wrap.review-list-wrap--member-review > div.revi
                    item_name = driver.find_elements(By.CSS_SELECTOR,'div.review-list > div.review-goods-information > div.review-goods-
                    rating = driver.find_elements(By.CSS_SELECTOR,'div.review-list > div.review-list__rating-wrap > span.review-list__rat
                    # rating 조건
                    if rating.size['width'] == 90:
                        rating = 5
```

| 추가 크롤링 코드

- ✓ 옷 뿐만 아니라 **전체 분류** 사용
- ✓ **Level 5 이상**의 user data



```
################ Arguments ################
def parse_args():
    parser = argparse.ArgumentParser(description="Run NeuMF.")
    parser.add_argument('--path', nargs='?', default='Data/',
                        help='Input data path.')
    parser.add_argument('--dataset', nargs='?', default='musinsa',
                        help='Choose a dataset.')
    parser.add_argument('--epochs', type=int, default=50,
                        help='Number of epochs.')
    parser.add_argument('--batch_size', type=int, default=128,
                        help='Batch size.')
    parser.add_argument('--num_factors', type=int, default=16,
                        help='Embedding size of MF model.')
    parser.add_argument('--layers', nargs='?', default='[64,32,16,8]',
                        help="MLP layers. Note that the first layer is the concatenation of user and item embeddings. So layers[0]/2 is the embedding
size.")
    parser.add_argument('--reg_mf', type=float, default=0,
                        help='Regularization for MF embeddings.')
    parser.add_argument('--reg_layers', nargs='?', default='[0,0,0,0]',
                        help="Regularization for each MLP layer. reg_layers[0] is the regularization for embeddings.")
    parser.add_argument('--num_neg', type=int, default=4,
                        help='Number of negative instances to pair with a positive instance.')
    parser.add_argument('--lr', type=float, default=0.001,
                        help='Learning rate.')
    parser.add_argument('--learner', nargs='?', default='sgd',
                        help='Specify an optimizer: adagrad, adam, rmsprop, sgd')
    parser.add_argument('--verbose', type=int, default=1,
                        help='Show performance per X iterations')
    parser.add_argument('--out', type=int, default=1,
                        help='Whether to save the trained model.')
    parser.add_argument('--mf_pretrain', nargs='?', default='',
                        help='Specify the pretrain model file for MF part. If empty, no pretrain will be used')
    parser.add_argument('--mlp_pretrain', nargs='?', default='',
                        help='Specify the pretrain model file for MLP part. If empty, no pretrain will be used')
    return parser.parse_args()
```

| 파라미터

- ✓ **리뷰 개수** 필터링 → 10개/20개
- ✓ **파라미터** 조정

모델	HR	NDCG	전체 user-item	리뷰 개수	필터링 후 user-item	비고
musinsa_NeuMF_8_[64,32,16,8]_1659608666.h5	**0.4399**	**0.3019**	(130645 , 88205)	**20개 이상**	(1033, 17724)	NCF 논문 데이터의 경우 각 user 당 20개 이상의 item을 가짐
musinsa_NeuMF_8_[64,32,16,8]_1659351220.h5	0.4307	0.2797	(110575 , 84923)	10개 이상	(3316, 33753)	
musinsa_NeuMF_8_[64,32,16,8]_1659251634.h5	0.4277	0.2684	(97293 , 78293)	10개 이상	(2508, 30432)	
musinsa_NeuMF_8_[64,32,16,8]_1658423553.h5	0.3344	0.1946	(60340, 48953)	10개 이상	(1951, 27542)	

05

결론

- 결과 출력
- 의의 및 한계

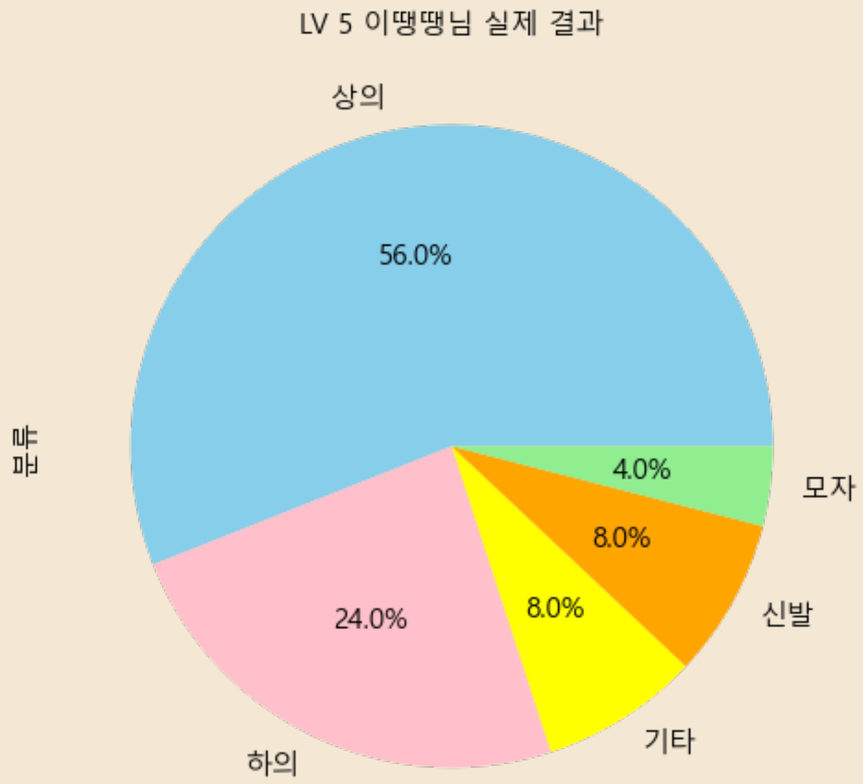
LV5. 이땡땡

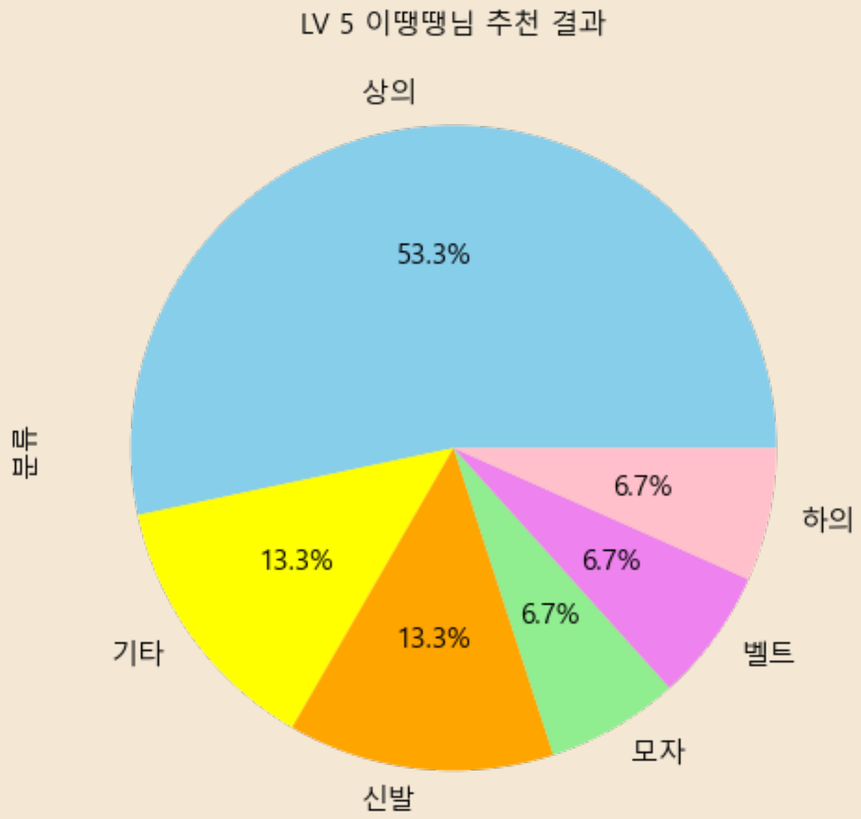
	유저명	아이템명	user_encoded	item_encoded
12892	LV 5 이땡땡님	썸머 더블턱 와이드 슬랙스[블랙]	398	9299
12893	LV 5 이땡땡님	KODACOLOR 워싱 니트 숏팬츠 CHARCOAL	398	9300
12894	LV 5 이땡땡님	KODACOLOR 반팔티셔츠 CHARCOAL	398	9301
12895	LV 5 이땡땡님	[SET] 로럴골든 하프 맨투맨 셋업_화이트멜란지	398	9302
12896	LV 5 이땡땡님	[SET] 쓰리피스 컴피 후드 아노락 셋업_블랙	398	1763
12897	LV 5 이땡땡님	SELLER LOGO SS TEE WHITE(CV2BMMT500A)	398	9303
12898	LV 5 이땡땡님	커브드 와이드 데님 팬츠_인디고	398	1318
12899	LV 5 이땡땡님	올드스쿨(캔버스) - 블랙:트루 화이트 / VN000ZDF1WX1	398	2186
12900	LV 5 이땡땡님	슬리드 옥스포드 오버셔츠(화이트)	398	1213
12901	LV 5 이땡땡님	시티보이 빅오버 옥스포드 셔츠_Royal Blue	398	451
12902	LV 5 이땡땡님	TAG SWEATPANTS - MELANGE GREY	398	9304
12903	LV 5 이땡땡님	[2PACK] 7.5oz 헤비웨이트 무지 오버핏 레이어드 반팔티	398	4998
12904	LV 5 이땡땡님	라이트웨이트 크루 삭스 7팩 [모노]	398	619
12905	LV 5 이땡땡님	SIGNATURE SMALL LOGO FULL ZIP-UP HOODIE - MELA...	398	9305
12906	LV 5 이땡땡님	[SET] YKK 2-WAY 테크 숏 자켓 셋업_차콜	398	6685
12907	LV 5 이땡땡님	볼드 엔아이 - (반다나) 드레스 블루:트루 화이트 / VN0A5DYAI9X1	398	8594
12908	LV 5 이땡땡님	덕아웃 로고 숄 볼캡 - 블랙:칠리페퍼 / VN0A2XAIA2T1	398	9306
12909	LV 5 이땡땡님	반스 클래식 반소매 티셔츠 - 그레이 / VN000GGG1RQ1	398	9307
12910	LV 5 이땡땡님	반스 클래식 반소매 티셔츠 - 블랙/ VN000GGGY281	398	3632
12911	LV 5 이땡땡님	Wide String Cargo Slacks Pants Black	398	784
12912	LV 5 이땡땡님	[2PACK] 베이식 무지 레이어드 반팔 티셔츠 일반 기장	398	3921
12913	LV 5 이땡땡님	SS미니멀 스탠다드 카고조거 슬랙스	398	322
12914	LV 5 이땡땡님	NJ1DN55A 남성 1996 에코 놉시 자켓	398	9308
12915	LV 5 이땡땡님	원턱 와이드 스웨트팬츠 블랙	398	2185
12916	LV 5 이땡땡님	베이식 긴팔 티셔츠 [화이트]	398	30

| 실제 구매 목록

itemId	prediction
(VEGAN)WARM UP SHORT PUFFER JACKET BLACK	0.999656
칼라 프린지 트리 하프 슬리브 니트_그린	0.970211
sw1403 램스울 라운드 니트-네이비	0.964336
TC1-SH08 프렌치 스트라이프 셔츠-그레이 그린	0.959872
핸드워시 50ml	0.954526
Heavy Corduroy Shirt S85 Ivory	0.937486
오픈카라 피케 루즈핏 반팔티 (NAVY BLUE)	0.909393
페이퍼 코튼 오버핏 반팔 셔츠 [CREAM BEIGE]	0.900607
블랙 미니멀 트위스트 버클 벨트	0.884704
아이폰 라리가 A1 케이스 - 아푼텐	0.873532
2PACK 30수 코튼 루즈핏 유넥 오프화이트	0.858881
N-COVER 볼캡 NY (BLACK)	0.858239
1968 HIMALAYA JEANS [WIDE STRAIGHT]	0.853933
[80YS] Summer Vacation_7(은주) 티셔츠	0.838624
로버츠 드라이버 슈즈 - 브라운	0.836679

| 추천 결과





- 상의는 비슷한 비율로 추천
- 하의가 적게 추천

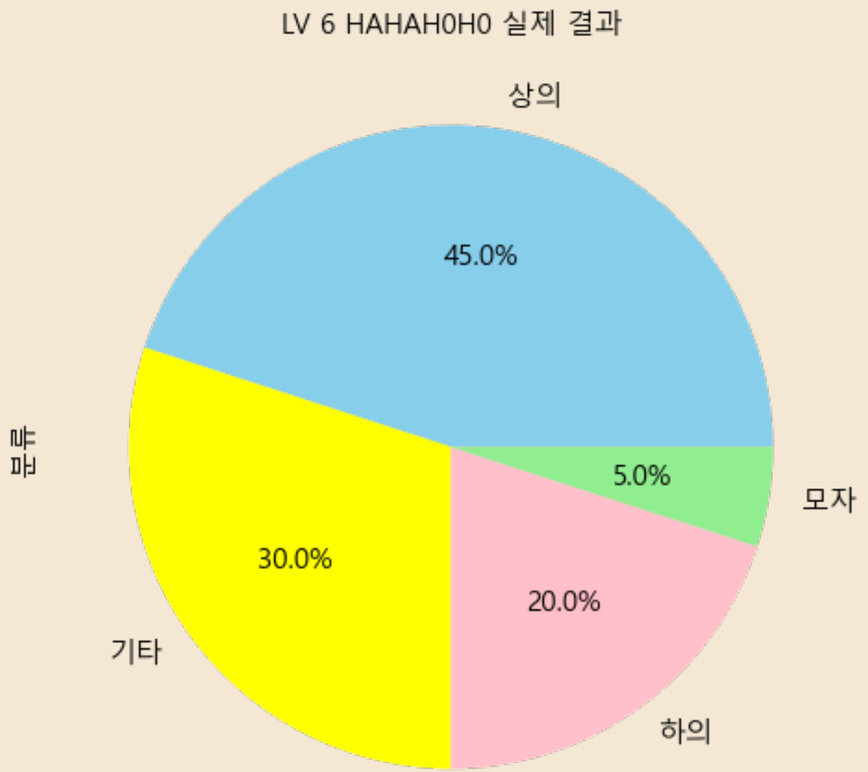
LV6. HAHAH0H0

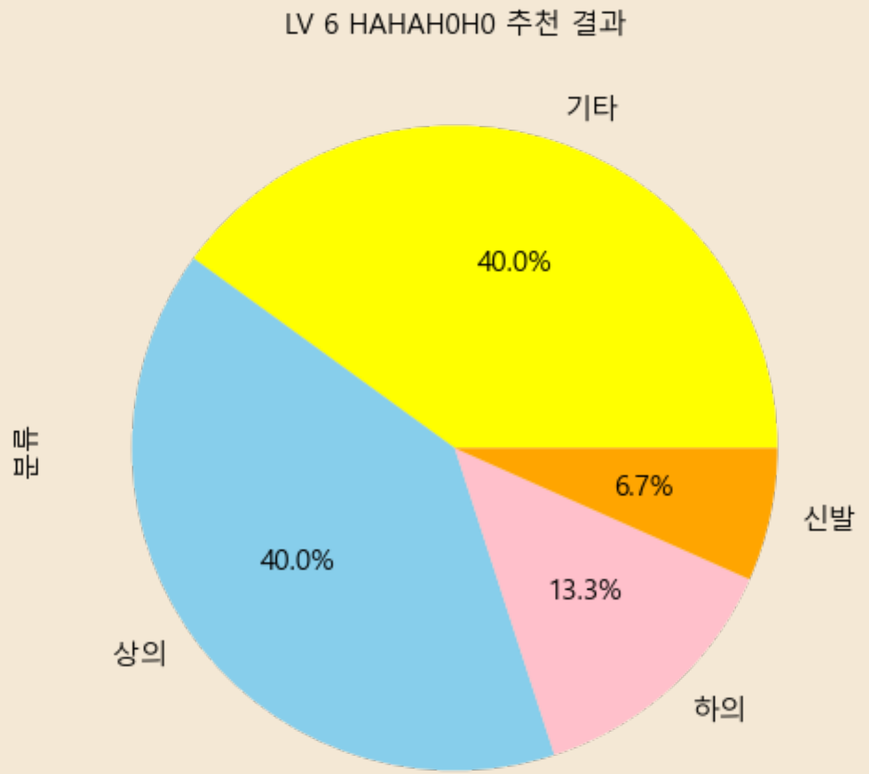
	유저명	아이템명	user_encoded	item_encoded
21560	LV 6 HAHAH0H0	[2세트]두가지연출 러블리 세라 니삭스	659	14669
21561	LV 6 HAHAH0H0	SMALL CLASSIC LOGO CAP pink	659	14670
21562	LV 6 HAHAH0H0	Contrast Stitch Carpenter Pants Beige	659	14671
21563	LV 6 HAHAH0H0	지오메트리컬 사이드 다트 팬츠 블랙	659	14672
21564	LV 6 HAHAH0H0	COLOR POINT COATING JUMPER IN WHITE	659	14673
21565	LV 6 HAHAH0H0	BABY SPORTY TOTE BAG IN RED	659	14674
21566	LV 6 HAHAH0H0	MATIN KIM BASIC WALLET IN BLACK	659	14675
21567	LV 6 HAHAH0H0	와이드 절개 벌룬팬츠 코튼 카키	659	6571
21568	LV 6 HAHAH0H0	유스풀 벌룬 맨투맨 네이비 SJMT1330	659	14676
21569	LV 6 HAHAH0H0	[GF70SOG] 리버스 위브 크루 - 옥스포드 그레이(스티치)	659	14677
21570	LV 6 HAHAH0H0	[오리지널스]22SS BALLOON FIT THREE TUCK SELVAGE DEN...	659	14678
21571	LV 6 HAHAH0H0	스테디 피시테일 자켓 블랙 JDOT1320	659	14679
21572	LV 6 HAHAH0H0	90 이얼스 캣 케이스	659	4341
21573	LV 6 HAHAH0H0	[35차 재입고!!]변색/알러지없는 써지컬스틸 슬림 원버튼 링 귀걸이	659	3128
21574	LV 6 HAHAH0H0	엘리스마샤 X 엘르 케렌 3Colors 숄더백 크로스백	659	14680
21575	LV 6 HAHAH0H0	608 유니섹스 비건 레더 점퍼_블랙	659	14681
21576	LV 6 HAHAH0H0	Squad Sweatshirt(Soy Cream)	659	14682
21577	LV 6 HAHAH0H0	[EZwithPIECE] DAISY SMART GLOVES (BLACK)	659	14683
21578	LV 6 HAHAH0H0	mini round quilting bag_baby pink	659	14684
21579	LV 6 HAHAH0H0	헤비웨이트 속기모 리얼 와이드 히든 밴딩 슬랙스 [블랙]	659	14685

| 실제 구매 목록

itemId	prediction
MNC 트레이닝 스웨트 팬츠 그레이	0.998288
CLASSIC LOGO TEE white	0.957477
[쿨탠다드] 팔토시 2팩 [화이트/블랙]	0.957141
TSHIRT BIG APPLE_IVORY	0.956645
향수공병 휴대용 향수리필 스프레이 용기 5ml	0.954893
[락피쉬웨더웨어x마르디메크르디] 오리지널 플랫폼 슬리퍼 - BLACK	0.938926
제로 선 클린 세트	0.937364
Vivid Flow Ring_7Color	0.926831
밸런스턱 와이드 치노 팬츠[블랙]	0.905889
가죽 마우스패드	0.899625
NN2PM54J 플랩 크로스 백 미니	0.894715
슈퍼브레이크 BLACK	0.879949
모헤어 보더 풀오버니트 - 레드	0.872231
우먼스 스텐다드 핏 그래픽 티셔츠 라벤더	0.864246
에센셜 쿨 맥스 3-PACK 드로우즈 블랙	0.861604

| 추천 결과





- ✓ 상의, 하의, 기타 모두 비슷한 비율로 추천
- ✓ 모자 대신 신발이 추천

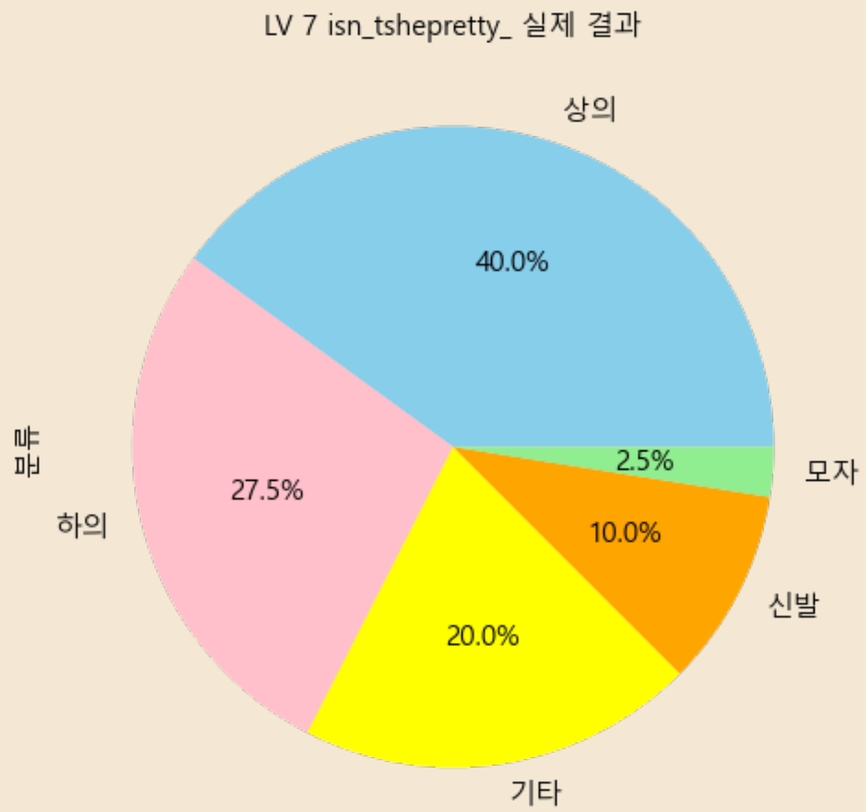
LV7. isn_tshepretty_

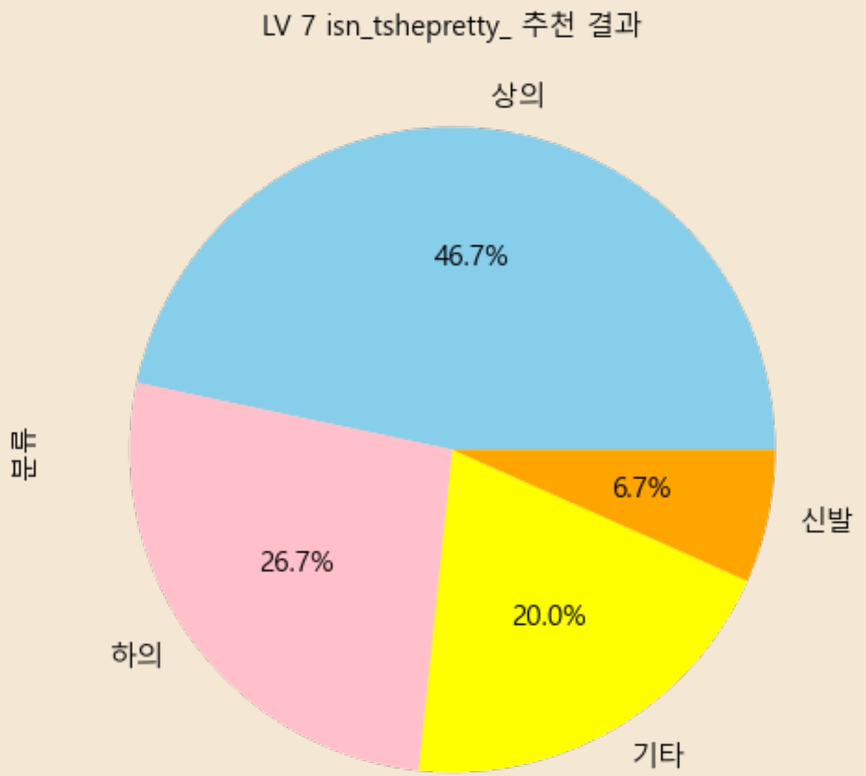
	유저명	아이템명	user_encoded	item_encoded
29486	LV 7 isn_tshepretty_	RUNNING RETRIEVER SHORT SLEEVE [NAVY]	884	18847
29487	LV 7 isn_tshepretty_	B Logo T-Shirts / WHITE MELANGE	884	18848
29488	LV 7 isn_tshepretty_	B Logo T-Shirts / BLACK	884	18849
29489	LV 7 isn_tshepretty_	[SET] 헤비웨이트 CGP 아치 로고 트레이닝 셋업_그레이	884	5357
29490	LV 7 isn_tshepretty_	컨버스 X 꼼데 가르송 플레이 척 70[행사용-판매종료]	884	18850
...	...	...	...	...
29548	LV 7 isn_tshepretty_	와이드 스웨트팬츠 블랙	884	9969
29549	LV 7 isn_tshepretty_	오버사이즈 트렌치 코트 BLACK	884	5241
29550	LV 7 isn_tshepretty_	Fleece Trucker Jacket - KHAKI BROWN	884	18879
29551	LV 7 isn_tshepretty_	[비건레더] 크루얼티프리 카라넥 MA-1 레더 점퍼 IRO112 아이보리	884	5737
29552	LV 7 isn_tshepretty_	라이프 포토그래피 트랙 자켓	884	18880

| 실제 구매 목록

itemId	prediction
와이드 스트링 하프 셔츠 _Off White	0.998846
LMC IDEAL TRACK SETUP gray	0.887792
여성 하이드로목 WHITE(화이트) 컬러 MLW1C2QA1801	0.881251
와이드 이지 밴딩 크롭 슬랙스 [더스티 베이지]	0.863966
크림 퍼퓸 더 퍼스트 무드	0.829631
오버핏 옥스포드 셔츠_SPYWC12C02	0.817937
프림백-아이보리	0.759933
[비건스웨이드] 라운드 스타디움 스웨이드 자켓 IRO131 아이보리	0.694951
클래식 핏 폴로 프렙스터 코듀로이 - 화이트	0.694588
1960 s/s tee navy	0.653418
와이드 히든 밴딩 슬랙스 [미디엄 그레이]	0.605268
울 컴포트 싱글 자켓 차콜	0.574127
Hailey Velvet Ribbon Hair String	0.569491
시그니처 후드 #2 오트밀	0.548454
S.S PRINT T-SHIRT - WHITE	0.541270

| 추천 결과





- 상의, 하의, 기타, 신발 모두 비슷한 비율로 추천
- 구매 내역에는 모자가 있지만 추천 결과에는 모자가 없음

딥러닝을 활용한 추천시스템 구현

여러 추천시스템 모델 중 딥러닝을 활용한 추천시스템을 구현

이는 필터링 모델에 비해 개인에게 맞는 item을 추천할 수 있음

과적합 문제

모델의 성능은 HR = 0.4399, NDCG = 0.3019로, 논문 모델의 성능(HR = 0.692, NDCG = 0.425)에 비해 부족함

이는 모델링에 사용한 데이터가 각 user별 item 수가 적고 편차가 커서 과적합 문제가 발생하는 것을 알 수 있음

데이터 셋의 한계

논문에 나온 데이터는 user 당 최소 20개 이상의 item을 가지고 있음

크롤링한 데이터의 경우 리뷰 개수의 한계가 있어 각 user별 item 수가 적은 데이터를 얻을 수 밖에 없음

한 user 당 리뷰하는 item 수가 늘어난 데이터 셋을 만든다면 과적합 문제 해결과 모델 성능을 개선할 수 있을 것이라 판단됨

Q & A

추천시스템 1조

THANK YOU

추천시스템 1조