# 프로젝트 기반 빅데이터 서비스 솔루션 개발 전문 과정

**교과목명 : 머신러닝응용**

- 평가일 : 03.28
- 성명 : 권혁종
- 점수 : 90점

문제 : LMEMBERS의 상품구매데이터를 이용하여 개인맞춤 상품 추천솔루션을 구축 후 다양한 활용 방안을 시현하세요.

# 1. 풀이 개요 및 활용 방안

- **풀이 개요**
  - 상품분류는 '중분류명'을 기준으로 새로운 통합분류 체계를 구축
  - 6개월단위로 데이터 분석 진행
  - 6~3개월전까지 많이 구입한 물품 중 3개월 이내로 구입하지 않은 물품과 3개월안에 많이 구매한 물품을 동시에 추천
  - knn이웃 방식과 잠재요인협업 필터링 방식을 사용하여 추천아이템 선정
- **활용 방안**
  - knn 최근접이웃과 잠재요인 협업 필터링 두 가지를 합쳐서 하이브리드 추천알고리즘 생성
  - 고객이 3개월 이전에는 많이 구매했지만 최근 3개월동안은 구매하지 않은 물품 추천
  - 1차 프로젝트와 결합하여 관리대상고객 예측한 결과를 바탕으로 관리대상고객에게 상품을 추천하여 관리대상고객의 매출증대
  - 중분류 안에서 랜덤으로 소분류 하나를 선택하여 추천

# 2. 추천시스템 개발 코드

### 데이터 로딩 및 전처리

In [118]:

```
import pandas as pd
data = pd.read_csv('../../lmembers/data/purprod2.csv',encoding='ms949')
```

In [3]:

```
data3 = data.loc[data.구매일자>=20151001]
data6 = data.loc[(data.구매일자<20151001)&(data.구매일자>=20150601)]
```

In [75]:

```
cate = pd.read_csv('../../lmembers/data/prodcl2.csv')
```

In [76]:

```
data3_cate = pd.merge(data3,cate,on='소분류코드')
data6_cate = pd.merge(data6,cate,on='소분류코드')
```

In [77]:

```
# 중분류명별 구매횟수를 기준으로 아이템 추천하기 위하여 피벗 생성
pdf3_ori = pd.pivot_table(data3_cate,
                      index = ['고객번호'],              # 행위치에 들어갈 열
                      columns = ['상품분류명'],          # 열위치에 들어갈 열
                      values = ['영수증번호'],              # 데이터로 사용할 열
                      aggfunc = ['count'])         # 데이터 집계 함수
pdf6_ori = pd.pivot_table(data6_cate,
                      index = ['고객번호'],
                      columns = ['상품분류명'],
                      values = ['영수증번호'],
                      aggfunc = ['count'])
pdf3=pdf3_ori.copy()
pdf6=pdf6_ori.copy()
```

In [79]:

```
pdf6_ori
```

Out[79]:

| 상품분류명 | BABY/INFANT | BOYS 고가 | BOYS 저가 | DIY 욕실용품 | EDUCATION 저가 | Fast Food | GIRLS | LOCAL 김치채소 | LOCAL 잎채소 | SPORT |
|---|---|---|---|---|---|---|---|---|---|---|
| 고객번호 | | | | | | | | | | |
| 1 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 2 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 3 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 4 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 6 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 7.0 | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 19379 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 19380 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 19381 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 2.0 | |
| 19382 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 1.0 | |
| 19383 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |

19319 rows × 405 columns

## 2.1 아이템 기반 최근접 이웃 협업 필터링

In [80]:

```
# Nan값 0으로 대체 및 데이터프레임 정리
pdf3.fillna(0,inplace=True)
pdf3 = pdf3['count']['영수증번호']
pdf6.fillna(0,inplace=True)
pdf6 = pdf6['count']['영수증번호']
```

In [81]:

```
pdf3
```

Out[81]:

| 상품분류명 | BABY/INFANT | BOYS 고가 | BOYS 저가 | DIY 욕실용품 | EDUCATION 저가 | Fast Food | GIRLS | LOCAL 김치채소 | LOCAL 잎채소 | SPORT |
|---|---|---|---|---|---|---|---|---|---|---|
| 고객번호 | | | | | | | | | | |
| **1** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| **2** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| **3** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| **4** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| **5** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **19379** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| **19380** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| **19381** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| **19382** | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 6.0 | |
| **19383** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |

19319 rows × 405 columns

In [82]:

```
# 코사인유사도는 행기준으로 작동하기에 transpose 함수로 전치
pdf_T = pdf3.transpose()
pdf_T
```

Out[82]:

| 고객번호 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | ... | 19374 | 19375 | 19376 | 19377 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 상품분류명 | | | | | | | | | | | | | | | |
| BABY/INFANT | 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 |
| BOYS 고가 | 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 |
| BOYS 저가 | 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 |
| DIY욕실용품 | 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 |
| EDUCATION 저가 | 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 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 홈웨어 | 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 |
| 화장품선물세트 | 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 |
| 황태 | 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 |
| 휴대폰용품 | 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 |
| 휴지통 | 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 |

405 rows × 19319 columns

In [83]:

```
# 코사인 유사도 산출 (행인 아이템들을 기준으로 사용자간의 유사도)
from sklearn.metrics.pairwise import cosine_similarity
pdf_sim = cosine_similarity(pdf_T, pdf_T)
```

In [84]:

```
# cosine_similarity()로 반환된 Numpy 행렬을 중분류명으로 매핑해 DataFrame으로 변환
pdf_sim_df = pd.DataFrame(data=pdf_sim, index=pdf3.columns,
                          columns=pdf3.columns)
print(pdf_sim_df.shape)
pdf_sim_df.head(3)
```

(405, 405)

Out[84]:

| 상품분류명 | BABY/INFANT | BOYS 고가 | BOYS 저가 | DIY욕실용품 | EDUCATION 저가 | Fast Food | GIRLS | 간 |
|---|---|---|---|---|---|---|---|---|
| 상품분류명 | | | | | | | | |
| **BABY/INFANT** | 1.000000 | 0.197077 | 0.292339 | 0.034579 | 0.392382 | 0.011274 | 0.341021 | 0 |
| **BOYS 고가** | 0.197077 | 1.000000 | 0.205775 | 0.014724 | 0.229297 | 0.021223 | 0.211933 | 0 |
| **BOYS 저가** | 0.292339 | 0.205775 | 1.000000 | 0.044597 | 0.291195 | 0.006107 | 0.249573 | 0 |

3 rows × 405 columns

In [85]:

```
pdf_sim_df["황태"].sort_values(ascending=False)[:6]
```

Out[85]:

```
상품분류명
황태        1.000000
양념채소      0.328772
두부        0.323479
열매채소      0.323437
잎채소       0.318029
콩나물       0.307825
Name: 황태, dtype: float64
```

In [86]:

```
import numpy as np
# 실제 구매횟수와 코사인유사도를 이용하여 개인화된 예측구매횟수를 구함
def predict_rating(ratings_arr, item_sim_arr):
    ratings_pred = ratings_arr.dot(item_sim_arr)/ np.array([np.abs(item_sim_arr).sum(axis=1)])
    return ratings_pred
```

In [87]:

```
ratings_pred_knn = predict_rating(pdf3.values , pdf_sim_df.values)
ratings_pred_matrix_knn = pd.DataFrame(data=ratings_pred_knn, index= pdf3.index,
                                       columns = pdf3.columns)
ratings_pred_matrix_knn.head(3)
```

Out[87]:

| 상품분류명<br><br>고객번호 | BABY/INFANT | BOYS 고가 | BOYS 저가 | DIY욕실용품 | EDUCATION 저가 | Fast Food | GIRLS | LOCAL 김치채소 | L |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.227782 | 0.182452 | 0.197233 | 0.128494 | 0.240327 | 0.105566 | 0.225790 | 0.114363 | 0.1 |
| 2 | 0.358539 | 0.261698 | 0.292359 | 0.167424 | 0.370140 | 0.147067 | 0.347004 | 0.167453 | 0.1 |
| 3 | 0.146388 | 0.145322 | 0.156448 | 0.160200 | 0.135735 | 0.137162 | 0.148086 | 0.133797 | 0.1 |

3 rows × 405 columns

In [88]:

```
from sklearn.metrics import mean_squared_error

# 고객이 구매한 물품에 대해서만 예측 성능 평가 MSE 를 구함.
def get_mse(pred, actual):
    # Ignore nonzero terms.
    pred = pred[actual.nonzero()].flatten()
    actual = actual[actual.nonzero()].flatten()
    return mean_squared_error(pred, actual)

print('아이템 기반 모든 인접 이웃 MSE: ', get_mse(ratings_pred_knn, pdf3.values ))
```

```
아이템 기반 모든 인접 이웃 MSE:  20.957257690837952
```

In [89]:

```
def predict_rating_topsim(ratings_arr, item_sim_arr, n=20):
    # 사용자-아이템 평점 행렬 크기만큼 0으로 채운 예측 행렬 초기화
    pred = np.zeros(ratings_arr.shape)

    # 사용자-아이템 평점 행렬의 열 크기만큼 Loop 수행.
    for col in range(ratings_arr.shape[1]):
        # 유사도 행렬에서 유사도가 큰 순으로 n개 데이터 행렬의 index 반환
        top_n_items = [np.argsort(item_sim_arr[:, col])[:-n-1:-1]]
        # 개인화된 예측 평점을 계산
        for row in range(ratings_arr.shape[0]):
            pred[row, col] = item_sim_arr[col, :][top_n_items].dot(ratings_arr[row, :][top_n_items].
            pred[row, col] /= np.sum(np.abs(item_sim_arr[col, :][top_n_items]))
    return pred
```

In [90]:

```
ratings_pred_knn = predict_rating_topsim(pdf3.values , pdf_sim_df.values, n=20)
print('아이템 기반 인접 TOP-20 이웃 MSE: ', get_mse(ratings_pred_knn, pdf3.values ))


# 계산된 예측 구매 데이터는 DataFrame으로 재생성
ratings_pred_matrix_knn = pd.DataFrame(data=ratings_pred_knn, index= pdf3.index,
                                       columns = pdf3.columns)
```

```
C:₩Users₩kpo01₩AppData₩Local₩Temp/ipykernel_18684/2382315358.py:11: FutureWarning: U
sing a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tupl
e(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array i
ndex, `arr[np.array(seq)]`, which will result either in an error or a different resu
lt.
  pred[row, col] = item_sim_arr[col, :][top_n_items].dot(ratings_arr[row, :][top_n_i
tems].T)
C:₩Users₩kpo01₩AppData₩Local₩Temp/ipykernel_18684/2382315358.py:12: FutureWarning: U
sing a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tupl
e(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array i
ndex, `arr[np.array(seq)]`, which will result either in an error or a different resu
lt.
  pred[row, col] /= np.sum(np.abs(item_sim_arr[col, :][top_n_items]))

아이템 기반 인접 TOP-20 이웃 MSE:  14.357487301579964
```

In [91]:

```
user_rating_id = pdf3.loc[5064, :]
user_rating_id[ user_rating_id > 0].sort_values(ascending=False)[:10]
```

Out[91]:

```
상품분류명
쿠키케이크     26.0
견과류        10.0
맥주         10.0
라면          9.0
두채류         8.0
우유          8.0
병통조림        8.0
스낵          7.0
청소욕실용품      6.0
베이커리        6.0
Name: 5064, dtype: float64
```

In [92]:

```
def get_not_buy_list(ratings_matrix, custid):
    # custid로 입력받은 사용자의 모든 구매정보 추출하여 Series로 반환함.
    # 반환된 user_rating 은 중분류명을 index로 가지는 Series 객체임.
    user_rating = ratings_matrix.loc[custid,:]

    # user_rating이 0보다 크면 기존에 구매한 물품임. 대상 index를 추출하여 list 객체로 만듬
    already_buy = user_rating[ user_rating > 0].index.tolist()

    # 모든 중분류명을 list 객체로 만듬.
    buy_list = ratings_matrix.columns.tolist()

    # 간편한 for문으로 already_buy에 해당하는 물품은 buy_list에서 제외함.
    notbuy_list = [ item for item in buy_list if item not in already_buy]

    return notbuy_list
```

In [93]:

```
def recomm_item_by_custid(pred_df, custid, notbuy_list, top_n=10):
    # 구매예측 DataFrame에서 사용자id index와 notbuy_list로 들어온 중분류명 컬럼을 추출하여 가장 예측
    recomm_items = pred_df.loc[custid, notbuy_list].sort_values(ascending=False)[:top_n]
    return recomm_items

# 사용자가 구매하지 않은 물품명 추출
notbuy_list_knn = get_not_buy_list(pdf3, 5064)

# 아이템 기반의 인접 이웃 협업 필터링으로 물품 추천
recomm_items_knn = recomm_item_by_custid(ratings_pred_matrix_knn, 5064, notbuy_list_knn, top_n=20)

# 구매 데이타를 DataFrame으로 생성.
recomm_items_knn = pd.DataFrame(data=recomm_items_knn.values,index=recomm_items_knn.index,columns=[
recomm_items_knn
```

Out[93]:

| 상품분류명 | pred_score |
|---|---|
| 아이스크림 | 5.149578 |
| 전통과자 | 4.946039 |
| 발효유 | 4.754618 |
| 즉석식품 | 4.564400 |
| 소주 | 4.475109 |
| 과채음료 | 4.342535 |
| 씨리얼 | 4.339017 |
| 냉동MS | 4.222804 |
| 두유 | 4.195458 |
| 기타음주류 | 4.189140 |
| 전통주 | 4.100823 |
| 냉장간식MS | 4.003405 |
| 냉장음료 | 3.872684 |
| 생수 | 3.741460 |
| 기타조리식품 | 3.674888 |
| 유아용품 | 3.656081 |
| 헤어용품 | 3.585563 |
| 주방주거세제 | 3.543273 |
| 냉장농산 | 3.394480 |
| 냉장기타MS | 3.392750 |

## 2.2 잠재요인 협업 필터링

In [94]:

```
import numpy as np
from sklearn.metrics import mean_squared_error

def get_rmse(R, P, Q, non_zeros):
    error = 0
    # 두개의 분해된 행렬 P와 Q.T의 내적 곱으로 예측 R 행렬 생성
    full_pred_matrix = np.dot(P,Q.T)

    # 실제 R 행렬에서 NULL이 아닌 값의 위치 인덱스 추출하여 실제 R 행렬과 예측 행렬의 RMSE 추출
    x_non_zero_ind = [non_zero[0] for non_zero in non_zeros]
    y_non_zero_ind = [non_zero[1] for non_zero in non_zeros]
    R_non_zeros = R[x_non_zero_ind, y_non_zero_ind]

    full_pred_matrix_non_zeros = full_pred_matrix[x_non_zero_ind,y_non_zero_ind]

    mse = mean_squared_error(R_non_zeros, full_pred_matrix_non_zeros)
    rmse = np.sqrt(mse)

    return rmse
```

In [95]:

```
# 행렬 분해
def matrix_factorization(R, K, steps=200, learning_rate=0.01, r_lambda=0.01):
    num_users, num_items = R.shape
    np.random.seed(1)
    P = np.random.normal(scale=1./K, size = (num_users,K))
    Q = np.random.normal(scale=1./K, size = (num_items,K))

    break_count = 0

    # R > 0인 행 위치, 열 위치, 값을 non_zeros 리스트 객체에 저장
    non_zeros = [(i,j,R[i,j]) for i in range(num_users) for j in range(num_items) if R[i,j] > 0]

    # SGD 기법으로 P와 Q 매트릭스를 계속 업데이트
    for step in range(steps):  # steps는 SGD의 반복횟수
        for i, j, r in non_zeros:
            # 실제 값과 예측 값의 차이인 오류 값 구함
            eij = r - np.dot(P[i,:],Q[j,:].T)

            P[i,:] = P[i,:] + learning_rate*(eij * Q[j,:] - r_lambda*P[i,:])
            Q[j,:] = Q[j,:] + learning_rate*(eij * P[i,:] - r_lambda*Q[j,:])

        rmse = get_rmse(R,P,Q, non_zeros)
        if ( step % 10) == 0: # 10회 반복할 때마다 오류 값 출력
            print(f'iteration step: {step}, rmse: {rmse}')
    return P, Q
```

In [96]:

```
# 구매횟수간 편차가 너무 커서 오류가 나서 log변환시킨값으로 예측 수행
pdf_log = pdf3.apply(lambda x: np.log1p(x))
pdf_log
```

Out[96]:

| 상품분류명 | BABY/INFANT | BOYS 고가 | BOYS 저가 | DIY 욕실용품 | EDUCATION 저가 | Fast Food | GIRLS | LOCAL 김치채소 | LOCAL 잎채소 | SPO |
|---|---|---|---|---|---|---|---|---|---|---|
| 고객번호 | | | | | | | | | | |
| **1** | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.00000 | |
| **2** | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.00000 | |
| **3** | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.00000 | |
| **4** | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.00000 | |
| **5** | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.00000 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **19379** | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.00000 | |
| **19380** | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.00000 | |
| **19381** | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.00000 | |
| **19382** | 0.0 | 0.693147 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.94591 | |
| **19383** | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.00000 | |

19319 rows × 405 columns

In [ ]:

```
pdf_log['휴지통']
```

In [97]:

```
P, Q = matrix_factorization(pdf_log.values, steps=200, K=50,learning_rate=0.001, r_lambda = 0.001)
pred_matrix = np.dot(P, Q.T)
```

```
iteration step: 0, rmse: 1.3281824895693333
iteration step: 10, rmse: 0.5534272250348119
iteration step: 20, rmse: 0.49231016490718316
iteration step: 30, rmse: 0.4710300257898537
iteration step: 40, rmse: 0.45810641087919307
iteration step: 50, rmse: 0.44156358010044916
iteration step: 60, rmse: 0.4214403615160305
iteration step: 70, rmse: 0.4010169534263127
iteration step: 80, rmse: 0.3805314015128914
iteration step: 90, rmse: 0.3600181674105359
iteration step: 100, rmse: 0.34037454025766334
iteration step: 110, rmse: 0.3224466714550372
iteration step: 120, rmse: 0.30653860354828166
iteration step: 130, rmse: 0.29261609853555903
iteration step: 140, rmse: 0.2805218688080491
iteration step: 150, rmse: 0.2700800559677009
iteration step: 160, rmse: 0.261125423036753
iteration step: 170, rmse: 0.2534885309437512
iteration step: 180, rmse: 0.2469880081160605
iteration step: 190, rmse: 0.24144287512886073
```

In [98]:

```
ratings_pred_matrix = pd.DataFrame(data=pred_matrix, index= pdf_log.index,
                                   columns = pdf_log.columns)

ratings_pred_matrix
```

Out[98]:

| 상품분류명<br>고객번호 | BABY/INFANT | BOYS 고가 | BOYS 저가 | DIY욕실용품 | EDUCATION 저가 | Fast Food | GIRLS | LOCAL 김치채소 |
|---|---|---|---|---|---|---|---|---|
| **1** | 0.711405 | 0.529906 | 0.639420 | 0.342873 | 0.675612 | 0.289166 | 0.701957 | 0.434854 |
| **2** | 0.986341 | 0.956986 | 0.880773 | 0.513446 | 0.894798 | 0.494093 | 0.953585 | 0.697234 |
| **3** | 1.123722 | 1.340591 | 1.037659 | 0.581880 | 0.984399 | 0.517136 | 1.040721 | 0.802160 |
| **4** | 0.912388 | 1.071613 | 0.930572 | 0.501660 | 0.929509 | 0.473222 | 1.053776 | 0.668318 |
| **5** | 0.847401 | 1.208503 | 0.884575 | 0.467232 | 0.861449 | 0.458737 | 0.897928 | 0.674778 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **19379** | 1.128274 | 1.193035 | 1.011524 | 0.662635 | 1.039798 | 0.649553 | 1.089176 | 0.873512 |
| **19380** | 1.290005 | 1.795983 | 1.246063 | 0.662792 | 1.269828 | 0.696445 | 1.584362 | 0.866004 |
| **19381** | 0.836009 | 0.957734 | 0.929333 | 0.568237 | 0.925742 | 0.573156 | 1.063935 | 0.866982 |
| **19382** | 0.758193 | 1.009897 | 0.852987 | 0.565118 | 0.768947 | 0.568543 | 0.972288 | 0.826281 |
| **19383** | 1.212370 | 1.356234 | 1.115543 | 0.764688 | 1.151411 | 0.745033 | 1.245374 | 1.030091 |

19319 rows × 405 columns

In [99]:

```
def recomm_movie_by_userid(pred_df, userId, unseen_list, top_n=10):
    # 예측 평점 DataFrame에서 사용자id index와 unseen_list로 들어온 영화명 컬럼을 추출하여 가장 예측

    recomm_movies = pred_df.loc[userId, unseen_list].sort_values(ascending=False)[:top_n]
    return recomm_movies
```

In [100]:

```
# get_not_buy_list 재활용하여 사지않은 물품 반환
notbuy_list = get_not_buy_list(pdf3, 5064)
notbuy_list
```

Out[100]:

```
['BABY/INFANT',
 'BOYS 고가',
 'BOYS 저가',
 'DIY욕실용품',
 'EDUCATION 저가',
 'Fast Food',
 'GIRLS',
 'LOCAL김치채소',
 'LOCAL잎채소',
 'SPORTS/OUTDOOR',
 'TRAVEL',
 '가공식품',
 '가구',
 '가방',
 '간편조리',
 '간편조리행사',
 '감',
 '감귤',
```

In [101]:

```
# 아이템 기반의 인접 이웃 협업 필터링으로 상품 추천
recomm_items_SGD = recomm_movie_by_userid(ratings_pred_matrix, 5064, notbuy_list, top_n=20)
recomm_items_SGD
```

Out[101]:

```
상품분류명
수입과자          2.149221
주유소          1.769743
전통주          1.671960
기타조리식품        1.632907
닭고기          1.623105
균일가          1.579312
요구르트          1.562248
냉장식사HMR       1.507150
발효유          1.484595
바나나          1.443609
브랜드빵          1.432905
어묵맛살          1.424901
아이스크림          1.421456
과채음료          1.400103
치즈           1.371746
고급의류          1.369399
이유식/유아간식       1.347832
```

In [102]:

```
# 평점 데이타를 DataFrame으로 생성.
recomm_items_SGD = pd.DataFrame(data=recomm_items_SGD.values,index=recomm_items_SGD.index,columns=[
recomm_items_SGD
```

Out[102]:

| 상품분류명 | pred_score |
|---|---|
| 수입과자 | 2.149221 |
| 주유소 | 1.769743 |
| 전통주 | 1.671960 |
| 기타조리식품 | 1.632907 |
| 닭고기 | 1.623105 |
| 균일가 | 1.579312 |
| 요구르트 | 1.562248 |
| 냉장식사HMR | 1.507150 |
| 발효유 | 1.484595 |
| 바나나 | 1.443609 |
| 브랜드빵 | 1.432905 |
| 어묵맛살 | 1.424901 |
| 아이스크림 | 1.421456 |
| 과채음료 | 1.400103 |
| 치즈 | 1.371746 |
| 고급의류 | 1.369399 |
| 이유식/유아간식 | 1.347832 |
| 면류 | 1.338552 |
| 냉장음료 | 1.336842 |
| 명품 | 1.304456 |

In [103]:

```
recomm_items_knn
```

Out[103]:

| 상품분류명 | pred_score |
|---|---|
| 아이스크림 | 5.149578 |
| 전통과자 | 4.946039 |
| 발효유 | 4.754618 |
| 즉석식품 | 4.564400 |
| 소주 | 4.475109 |
| 과채음료 | 4.342535 |
| 씨리얼 | 4.339017 |
| 냉동MS | 4.222804 |
| 두유 | 4.195458 |
| 기타음주류 | 4.189140 |
| 전통주 | 4.100823 |
| 냉장간식MS | 4.003405 |
| 냉장음료 | 3.872684 |
| 생수 | 3.741460 |
| 기타조리식품 | 3.674888 |
| 유아용품 | 3.656081 |
| 헤어용품 | 3.585563 |
| 주방주거세제 | 3.543273 |
| 냉장농산 | 3.394480 |
| 냉장기타MS | 3.392750 |

# 3. 활용 방안

## 3.1 최근접이웃&잠재요인 협업필터링을 합쳐서 하이브리드 추천알고리즘 생성

In [105]:

```
# knn 과 잠재요인 협업 필터링 두가지에 공통된 요소를 추천
recom_items = pd.merge(recomm_items_SGD,recomm_items_knn,on='상품분류명')
recom_items
```

Out[105]:

| 상품분류명 | pred_score_x | pred_score_y |
|---|---|---|
| 전통주 | 1.671960 | 4.100823 |
| 기타조리식품 | 1.632907 | 3.674888 |
| 발효유 | 1.484595 | 4.754618 |
| 아이스크림 | 1.421456 | 5.149578 |
| 과채음료 | 1.400103 | 4.342535 |
| 냉장음료 | 1.336842 | 3.872684 |

In [106]:

```
recom_items['recom_pred'] = recom_items.iloc[:,1]*recom_items.iloc[:,0]
recom_items = recom_items.sort_values(by='recom_pred',ascending=False)
```

In [107]:

```
recom_items
```

Out[107]:

| 상품분류명 | pred_score_x | pred_score_y | recom_pred |
|---|---|---|---|
| 아이스크림 | 1.421456 | 5.149578 | 7.319901 |
| 발효유 | 1.484595 | 4.754618 | 7.058685 |
| 전통주 | 1.671960 | 4.100823 | 6.856412 |
| 과채음료 | 1.400103 | 4.342535 | 6.079996 |
| 기타조리식품 | 1.632907 | 3.674888 | 6.000750 |
| 냉장음료 | 1.336842 | 3.872684 | 5.177167 |

## 3.2 3개월 이전에는 많이 구매했지만, 최근 3개월동안은 구매하지 않은 물품 추천

In [108]:

```
# 6~3개월전까지 많이 구입한 물품 중 3개월 이내로 구입하지 않은 물품을 추천
def recom_notbuy(custid,pdf3,pdf6):
    user_rating = pdf6.loc[custid,:]
    already_buy = user_rating[ user_rating > 0].sort_values(ascending=False)
    notbuy_list = get_not_buy_list(pdf3, custid)
    for x in already_buy.index:
        if x in notbuy_list:
            already_buy.drop(index=x ,axis=0,inplace=True)
    recom_list = already_buy
    return recom_list
```

In [109]:

```
recom_notbuy(546,pdf3,pdf6)
```

Out[109]:

```
상품분류명
농산물        26.0
명품          12.0
가공식품        11.0
젓갈/반찬        8.0
일식           6.0
기초화장품         5.0
주방용품         4.0
구두           3.0
수산품          2.0
차/커피         2.0
건강기능식품       1.0
Name: 546, dtype: float64
```

## 3.3 1차 프로젝트와 결합하여 관리대상고객 예측한 결과를 바탕으로 관리대상고객에게 상품을 추천하여 관리대상고객의 매출증대

In [112]:

```
def recommend_list(custid,pdf3,pdf6,top = 50,N=5):
    notbuy_list = get_not_buy_list(pdf3, custid)
    recomm_items_knn = recomm_item_by_custid(ratings_pred_matrix_knn, custid, notbuy_list, top_n=top
    recomm_items_knn = pd.DataFrame(data=recomm_items_knn.values,index=recomm_items_knn.index,columr
    recomm_items_SGD = recomm_movie_by_userid(ratings_pred_matrix, custid, notbuy_list, top_n=top)
    recomm_items_SGD = pd.DataFrame(data=recomm_items_SGD.values,index=recomm_items_SGD.index,columr
    recom_items = pd.merge(recomm_items_SGD,recomm_items_knn,on='상품분류명')
    recom_items['recom_pred'] = recom_items.iloc[:,1]*recom_items.iloc[:,0]
    recom_items = recom_items.sort_values(by='recom_pred',ascending=False)
    recom_list = recom_notbuy(custid,pdf3,pdf6)
    print(f'3개월전에 {recom_list.index.values[:N]}물품들을 구매하셨는데 지금은 필요하지 않으신가요?
    print()
    print(f'{custid}고객님이 구매하신 상품을 구매한 고객들이 많이 구매한 {recom_items.index.values[:
```

In [135]:

```
recommend_list(1234,pdf3,pdf6)
```

3개월전에 ['열매채소' '양념채소' '잎채소' '우유' '닭고기']물품들을 구매하셨는데 지금
은 필요하지 않으신가요?

1234고객님이 구매하신 상품을 구매한 고객들이 많이 구매한 ['종량제봉투' '샐러드채소'
'어묵맛살' '브랜드빵' '냉동만두']은/는 어떠세요?

## 3.4 중분류 안에서 랜덤으로 소분류 하나를 선택하여 추천

In [123]:

```
import numpy as np
def recommend_more_list(custid,pdf3,pdf6,top = 50,N=5):
    global cate
    notbuy_list = get_not_buy_list(pdf3, custid)
    recomm_items_knn = recomm_item_by_custid(ratings_pred_matrix_knn, custid, notbuy_list, top_n=top
    recomm_items_knn = pd.DataFrame(data=recomm_items_knn.values,index=recomm_items_knn.index,columr
    recomm_items_SGD = recomm_movie_by_userid(ratings_pred_matrix, custid, notbuy_list, top_n=top)
    recomm_items_SGD = pd.DataFrame(data=recomm_items_SGD.values,index=recomm_items_SGD.index,columr
    recom_items = pd.merge(recomm_items_SGD,recomm_items_knn,on='상품분류명')
    recom_items['recom_pred'] = recom_items.iloc[:,1]*recom_items.iloc[:,0]
    recom_items = recom_items.sort_values(by='recom_pred',ascending=False)
    recom_list = recom_notbuy(custid,pdf3,pdf6)
    more_list = []
    more_list2 = []
    for i in recom_list.index.values[:N]:
        more_list.append(np.random.choice(cate.소분류명[cate.상품분류명==i]))
    for i in recom_items.index.values[:N]:
        more_list2.append(np.random.choice(cate.소분류명[cate.상품분류명==i]))
    print(f'3개월전에 {recom_list.index.values[:N]}물품들을 구매하셨는데 혹시 {more_list}는 어떠신가
    print()
    print(f'{custid}고객님이 구매하신 상품을 구매한 고객들이 많이 구매한 {more_list2}은/는 어떠세요?
```

In [136]:

```
recommend_more_list(1234,pdf3,pdf6)
```

3개월전에 ['열매채소' '양념채소' '잎채소' '우유' '닭고기']물품들을 구매하셨는데 혹시
['콩류', '건고추', '상추', '과일맛우유', '닭정육']는 어떠신가요?

1234고객님이 구매하신 상품을 구매한 고객들이 많이 구매한 ['종량제봉투', '즙용양채',
'맛살', '일반빵류', '교자만두']은/는 어떠세요?