Applied Machine Learning Term Project

June 07, 2021

20181353 Kim Yejin
20181331 Min Jiyeong

CONTENTS

INTRODUCTION

"How can we use sales data
for customer management and sales growth?"

INTRODUCTION

Data description

VS

Win!

시사저널e.

Home plus emart LOTTE Mart

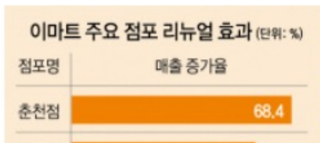
경제·금융 › 경제·금융일반

대형마트 매출 전략? 데이터에 물어봐

입력 2021-05-17 17:57:40 수정 2021.05.17 17:57:40 박민주 기자

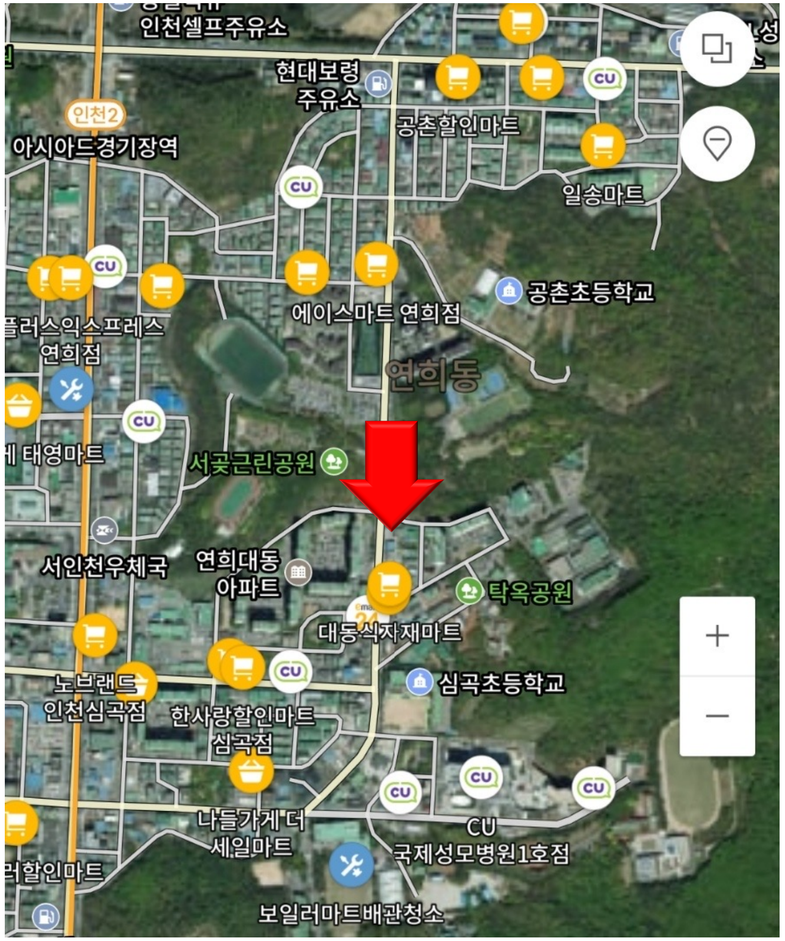
URL

고객 설문 기반으로 약점 보완하고
인근 상권·인구 분석해 공간 최적화
이마트 리뉴얼 점포 매출 두자릿수↑
홈플러스·롯데마트도 새단장 가속



INTRODUCTION

Data description



- There are apartment complexes, elementary schools, and hospitals nearby.
- Emart-24, CU, Homeplus express, Nobrand…
- We can get about 1000 sales data

```
In [1]: import os
        path_directory = 'C:/Users/USER/Desktop/기계학습응용/판매 데이터'
        file_list = os.listdir(path_directory)

In [3]: df=file_list

In [4]: len(df)
Out[4]: 975
```

Each data looks like in this way →

거래내역서_한혜미_2020-09-06(20210320) - 제한된 보기

파일 홈 삽입 그리기 페이지 레이아웃 수식 데이터 검토 보기 도움말

제한된 보기 정책 설정 때문에 이 파일 형식을 편집하는 것이 허용되지 않습니다. 자세한 내용을 보려면 클릭하십시오.

A1 No

	A	B	C	D	E	F	G	H	I	J
1	No	상품명	바코드	단가	수량	할인	금액	상품포인트	매출원가	비고
2	1	댓골두부찌	880833016	1000	2	0	2000	5	1600	
3	2	남양맛있는	880106922	4500	1	0	4500	13	3900	
4	3	한돈(앞다	209175	5615	1	0	5615	16	0	
5	4	왓따콜라	880106232	400	2	0	800	2	568	
6	5	바밤바	880971322	400	9	0	3600	10	2520	
7	6	해태자유시	880101920	800	3	0	2400	10	1579.8	
8		합계	6품목		18	0	18915	56	10168	
9										
10										
11										
12										
13										
14										
15										
16										
17										
18										
19										
20										
21										
22										
23										
24										
25										

INTRODUCTION

Goals

01

Classify product names according to item type using TF-IDF.

02

Discover the association rules between purchased products.

03

Improve product placement based on the relationship between products.

04

Predict future purchases and establish a product recommendation system.

“For customer management and sales growth”

METHOD

Product extraction & Classification

Our Dataframe!

	No	상품명	바코드	단가	수량	할인	금액	상품포인트	매출원가	비고
0	1	댓골두부찌개용300g	8[illegible]330	1000	2	0	2000	5	1600	
1	1	남양맛있는우유GT기획	8801069221397	4500	1	0	4500	13	3900	
2	1	한돈(앞다리살)찌개용	2001[illegible]	5615	1	0	1615	16	0	
3	1	왓따콜라	8801062321773	400	2	0	800	2	568	
4	1	바밤바	8809713220055	400	9	0	3600	10	2520	
...	...	...	...	...	...	...	...	...	...	...
3767	975	농심쌀새우깡80g	8801043036054	1150	1	0	1150	3	850	
3768	975	농심양파깡77g	8801043036498	1300	1	0	1300	3	965	
3769	975	농심매운새우깡90g	8801043036078	1150	1	0	1150	3	850	
3770	975	재사용봉투10리터	9144539122350	310	1	0	310	0	285	
3771	975	밀크플러스900ml*2	8806371306381	2980	1	0	2980	13	2500	

Example)

댓골두부찌개용300g

↓

[illegible]

↓

남양맛있는우유GT기획

↓

우유

1. Product extraction & Classification

: Classify product names according to item type using TF-IDF.

METHOD

Product extraction & Classification

Our Dataframe!

	No	상품명	바코드	단가	수량	할인	금액	상품포인트	매출원가	비고
0	1	댓골두부찌개용300g	8808330161130	1000	2	0	2000	5	1600	
1	1	남양맛있는우유GT기획	8801069221397	4500	1	0	4500	13	3900	
2	1	한돈(앞다리살)찌개용	209175	5615	1	0	5615	16	0	
3	1	왓따콜라	8801062321773	400	2	0	800	2	568	
4	1	바밤바	8809713220055	400	9	0	3600	10	2520	
...	...	...	...	...	...	...	...	...	...	...
3767	975	농심쌀새우깡80g	8801043036054	1150	1	0	1150	3	850	
3768	975	농심양파깡77g	8801043036498	1300	1	0	1300	3	965	
3769	975	농심매운새우깡90g	8801043036078	1150	1	0	1150	3	850	
3770	975	재사용봉투10리터	9144539122350	310	1	0	310	0	285	
3771	975	밀크플러스900ml*2	8806371306381	2980	1	0	2980	13	2500	

Example)

댓골두부찌개용300g

↓

두부

남양맛있는우유GT기획

↓

우유

METHOD

Product extraction & Classification

• USE TF-IDF!!

: Statistical measure that evaluates how relevant a word is to a document in a collection of do
two metrics: how many times a word appears in a document, and the inverse document fr
documents.

It has many uses, most importantly in automated text analysis, and is very useful for scoring v
for Natural Language Processing (NLP).

• Original TF - IDF

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

• Modified TF – DF

term frequency * document frequency

'두부'

댓골두부찌개용300g
종가집콩이가득두부찌개용300g
풀무원소가찌개두부290g
강릉심층수순두부400g
풀무원순두부찌게양념장140g
백설두루두루두부800g
풀무원소가찌개두부200g
두부1모
강릉최가두부550g
댓골보글보글두부찌개용
풀무원소가부침두부290g
cj)몽글몽글순두부350g
댓골국산부침찌개두부300g
풀무원국산콩두부부침용300g
다담바지락순두부양념 140g
풀무원단단한두부1kg
댓골지글지글두부부침용500g
하늘찬연두부
하늘찬순두부400g
종가집콩이가득부침두부300g
두부
행복한국산콩찌게두부300
반모두부(부침/찌개)150g*2
종가집콩이가득큰두부800g
맛있는콩두부찌개용300g
풀무원소가순두부350g
CJ)행콩국산두부찌개180g
행복한콩두부부침용180g
맛있는콩두부부침용300g

APPLICATION

Product extraction & Classification

Step 1. Split the data into morpheme

```
from konlpy.tag import Okt

okt = Okt()
print(okt.pos('댓골두부찌개용'))
print(okt.pos('남양맛있는우유기획'))
```

```
[('댓', 'Modifier'), ('골', 'Noun'), ('두부', 'Noun'), ('찌개', 'Noun'), ('용', 'Noun')]
[('남양', 'Noun'), ('맛있는', 'Adjective'), ('우유', 'Noun'), ('기획', 'Noun')]
```

Step 2. Apply modified TF-DF algorithm

```
# modified TF-DF
## 문장 전체에서의 단어 t 출현 횟수
dt_dic = {}

## 한 문장 내부에서 단어 t 출현 횟수
result = []

for words in word_dic_list:
    used_word = {}
    data = np.zeros(word_dic['_id'])

    for id_ in words:
        data[id_]+=1
        used_word[id_]=1
    for id_ in used_word:
        if not(id_ in dt_dic):
            dt_dic[id_]=0
        dt_dic[id_]+=1
    data = data/len(words)
    result.append(data)

for i, doc in enumerate(result):
    for id, v in enumerate(doc):
        idf = np.log(dt_dic[id_]/len(word_dic_list)) +1
        doc[id_] = min([doc[id]*idf, 1.0])
    result[i] = doc

print(result[0])
```

Step 3. Extract the key-word

	원래이름	키워드
0	댓골두부찌개용	두부
1	남양맛있는우유기획	우유
2	한돈앞다리살찌개용	돈
3	왓따콜라	왓따콜
4	바밤바	바밤바
5	해태자유시간	자유시간
6	팽이버섯	팽이버섯
7	서울우유입	우유
8	종가집콩이가득두부찌개용	두부

Problem!!

‘왓따콜라’ → 음료
‘바밤바’ → 아이스크림
‘해태자유시간’ → 제과류

How to solve the problem?

```
['해태자유시간',
'롯데치토스매콤한맛',
'오리온초코송이',
'빙그레더위사냥엑티브',
'해태맛동산',
'크라운화이트하임',
'오리온다이제초코',
'오리온포카칩오리지날',
'허니버터칩',
'크라운초코하임',
'크라운콘초헤이즐넛',
'농심신라면멀티입',
'롯데스피아민트',
'롯데돼지콘',
'롯데더블비얀코',
'빙그레엑설런트',
'빙그레붕어싸만코',
'오리온땅콩강정',
'스윙칩고추장맛',
'롯데쥬시후레쉬껌',
```

APPLICATION

Product extraction & Classification

METHOD

Association Rules Analysis

The association rule is expressed in the form of ‘X→Y’

- Support(X→Y)

$= \Pr(X \cap Y)$

$= \frac{\text{Number of transactions involving products X and Y at the same time}}{\text{Total number of transactions}}$

- Confidence(X→Y)

$= \frac{\Pr(X \cap Y)}{\Pr(X)}$

$= \frac{\text{Number of transactions involving products X and Y at the same time}}{\text{Number of transactions with product X}}$

- Lift(X→Y) = Lift(Y→X)

$= \frac{\Pr(X \cap Y)}{\Pr(X) \times \Pr(Y)} = \frac{\text{Confidence}}{\Pr(Y)} = \frac{\text{Number of transactions involving products X and Y at the same time}}{\text{Number of transactions with product X}} \times \frac{\text{Total number of transactions}}{\text{Number of transactions with product Y}}$

We are related to each other!

2. Association Rules Analysis

- Discover the association rules between purchased products. & improve product placement based on the relationship between products.

METHOD

Association Rules Analysis

The association rule is expressed in the form of ‘X→Y’

- Support(X→Y)

$= \Pr(X \cap Y)$

$= \frac{\text{Number of transactions involving products X and Y at the same time}}{\text{Total number of transactions}}$

- Confidence(X→Y)

$= \frac{\Pr(X \cap Y)}{\Pr(X)}$

$= \frac{\text{Number of transactions involving products X and Y at the same time}}{\text{Number of transactions with product X}}$

- Lift(X→Y) = Lift(Y→X)

$= \frac{\Pr(X \cap Y)}{\Pr(X) \times \Pr(Y)} = \frac{\text{Confidence}}{\Pr(Y)} = \frac{\text{Number of transactions involving products X and Y at the same time}}{\text{Number of transactions with product X}} \times \frac{\text{Total number of transactions}}{\text{Number of transactions with product Y}}$

METHOD

Apriori algorithm

• Apriori algorithm is to calculate the association rule by finding only the frequent item-sets above the minimum support.

• Apriori principle: If an item-set is frequent, then all of its subsets must also be frequent.

• Anti-monotonicity: The support of an item-set never exceeds the support of its subsets.

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

APPLICATION

Apriori algorithm

Step1. Convert the data frame into a list.

```
data_arr=[[]for i in range(976)]
num=0

for i in data['최종상품명'] :
    data_arr[data['No'].iloc[num]].append(i)
    num+=1
```

```
data_arr.pop(0)
num=0

for i in data_arr :
    data_arr[num] = list(set(data_arr[num]))
    num+=1

data_arr
```

```
[['우유', '두부', '음료', '제과류', '돼지고기', '아이스크림'],
 ['우유', '두부', '음료', '소시지', '버섯', '돼지고기'],
 ['라면', '우유', '무', '제과류', '아이스크림'],
 ['딸기', '아이스크림'],
 ['애호박', '우유', '두부', '음료', '무', '식초'],
 ['우유', '제과류', '음료', '아이스크림'],
 ['계란', '우유', '제과류', '탕'],
 ['우유', '물티슈', '당근', '무', '탕', '조명'],
 ['비피더스', '제과류', '애호박', '밀가루'],
 ['라면', '우유', '버섯', '무', '제과류', '탕', '비피더스'],
 ['음료', '귤'],
 ['우유', '물티슈', '음료', '떡', '제과류', '만두', '탕'],
 ['우유', '제과류'],
 ['부탄가스'],
 ['부추', '물티슈', '만두'],
 ['양파', '딸기', '두부', '오일세럼', '돼지고기', '소스'],
 ['비피더스', '우유', '마요네즈'],
 ['라면', '깻잎', '배추'],
 ['바나나'],
```

APPLICATION

Apriori algorithm

Step2. Transform the list using “mlxtend”.

```
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules

te = TransactionEncoder()
te_ary = te.fit(data_arr).transform(data_arr)
df = pd.DataFrame(te_ary, columns=te.columns_)
df
```

	가지	간장	간청각	갈비	감자	감자전분	강낭콩	건포도	계란	고구마	...	햇반	행주	호떡	호박	홍어	홍합	황태	후추	휘이설퐁	휴지
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
970	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
971	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
972	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
973	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
974	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False

APPLICATION

Apriori algorithm

Step3. Apply the Apriori algorithm.

```
frequent_itemsets = apriori(df, min_support=0.02, use_colnames=True)
frequent_itemsets
```

	support	itemsets
0	0.053333	(계란)
1	0.047179	(고추)
2	0.025641	(귤)
3	0.063590	(나물)
4	0.033846	(담배)
5	0.027692	(당근)
6	0.056410	(대파)
7	0.075897	(돼지고기)
8	0.107692	(두부)
9	0.035897	(딸기)
10	0.093333	(라면)
11	0.031795	(마늘)
12	0.028718	(막걸리)
13	0.082051	(맥주)
14	0.144615	(무)
15	0.030769	(물티슈)
16	0.029744	(미역)
17	0.026667	(밀가루)
18	0.044103	(배추)
19	0.053333	(버섯)
20	0.020513	(사과)
21	0.029744	(삼겹살)
22	0.040000	(상추)
23	0.025641	(소세지)
24	0.023590	(소스)
25	0.111795	(소주)
26	0.026667	(숙주나물)
27	0.048205	(아이스크림)
28	0.036923	(애호박)
29	0.047179	(양파)
30	0.048205	(어묵)
31	0.024615	(오이)
32	0.023590	(오징어)
33	0.153846	(우유)
34	0.217436	(음료)
35	0.232821	(제과류)
36	0.026667	(치즈)
37	0.052308	(탕)
38	0.028718	(한우)
39	0.023590	(라면, 음료)
40	0.026667	(라면, 제과류)
41	0.023590	(맥주, 소주)
42	0.025641	(맥주, 음료)
43	0.023590	(맥주, 제과류)
44	0.023590	(소주, 무)
45	0.024615	(무, 음료)
46	0.032821	(무, 제과류)
47	0.020513	(소주, 우유)
48	0.027692	(소주, 음료)
49	0.024615	(소주, 제과류)
50	0.022564	(제과류, 아이스크림)
51	0.031795	(우유, 음료)
52	0.036923	(우유, 제과류)
53	0.061538	(제과류, 음료)

39	0.023590	(라면, 음료)	
40	0.026667	(라면, 제과류)	
41	0.023590	(맥주, 소주)	
42	0.025641	(맥주, 음료)	
43	0.023590	(맥주, 제과류)	
44	0.023590	(소주, 무)	
45	0.024615	(무, 음료)	
46	0.032821	(무, 제과류)	③
47	0.020513	(소주, 우유)	
48	0.027692	(소주, 음료)	⑤
49	0.024615	(소주, 제과류)	
50	0.022564	(제과류, 아이스크림)	
51	0.031795	(우유, 음료)	④
52	0.036923	(우유, 제과류)	②
53	0.061538	(제과류, 음료)	①

APPLICATION

Apriori algorithm

Step4. Generate the rules.

```
network_df=association_rules(frequent_itemsets, metric="lift", min_threshold=1)
network_df
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(라면)	(음료)	0.093333	0.217436	0.023590	0.252747	1.162399	0.003296	1.047255
1	(음료)	(라면)	0.217436	0.093333	0.023590	0.108491	1.162399	0.003296	1.017002
2	(라면)	(제과류)	0.093333	0.232821	0.026667	0.285714	1.227187	0.004937	1.074051
3	(제과류)	(라면)	0.232821	0.093333	0.026667	0.114537	1.227187	0.004937	1.023947
4	(맥주)	(소주)	0.082051	0.111795	0.023590	0.287500	2.571674	0.014417	1.246604
5	(소주)	(맥주)	0.111795	0.082051	0.023590	0.211009	2.571674	0.014417	1.163447
6	(맥주)	(음료)	0.082051	0.217436	0.025641	0.312500	1.437205	0.007800	1.138275
7	(음료)	(맥주)	0.217436	0.082051	0.025641	0.117925	1.437205	0.007800	1.040669
8	(맥주)	(제과류)	0.082051	0.232821	0.023590	0.287500	1.234857	0.004487	1.076743
9	(제과류)	(맥주)	0.232821	0.082051	0.023590	0.101322	1.234857	0.004487	1.021443
10	(소주)	(우)	0.111795	0.144615	0.023590	0.211009	1.459106	0.007422	1.084150
11	(우)	(소주)	0.144615	0.111795	0.023590	0.163121	1.459106	0.007422	1.061330
12	(소주)	(우유)	0.111795	0.153846	0.020513	0.183486	1.192661	0.003314	1.036301
13	(우유)	(소주)	0.153846	0.111795	0.020513	0.133333	1.192661	0.003314	1.024852
14	(소주)	(음료)	0.111795	0.217436	0.027692	0.247706	1.139216	0.003384	1.040238
15	(음료)	(소주)	0.217436	0.111795	0.027692	0.127358	1.139216	0.003384	1.017835
16	(제과류)	(아이스크림)	0.232821	0.048205	0.022564	0.096916	2.010498	0.011341	1.053939
17	(아이스크림)	(제과류)	0.048205	0.232821	0.022564	0.468085	2.010498	0.011341	1.442297
18	(우유)	(제과류)	0.153846	0.232821	0.036923	0.240000	1.030837	0.001105	1.009447
19	(제과류)	(우유)	0.232821	0.153846	0.036923	0.158590	1.030837	0.001105	1.005638
20	(제과류)	(음료)	0.232821	0.217436	0.061538	0.264317	1.215610	0.010915	1.063725
21	(음료)	(제과류)	0.217436	0.232821	0.061538	0.283019	1.215610	0.010915	1.070013

APPLICATION

Apriori algorithm

Step5. Filter the rules.

Ramen → Drink

Ramen → Confectionery

Beer ⇄ Soju

Beer → Drink

Beer → Confectionery

Soju → Radish

Soju → Drink

Ice cream → Confectionery

Milk → Snack

Confectionery ⇄ Drink

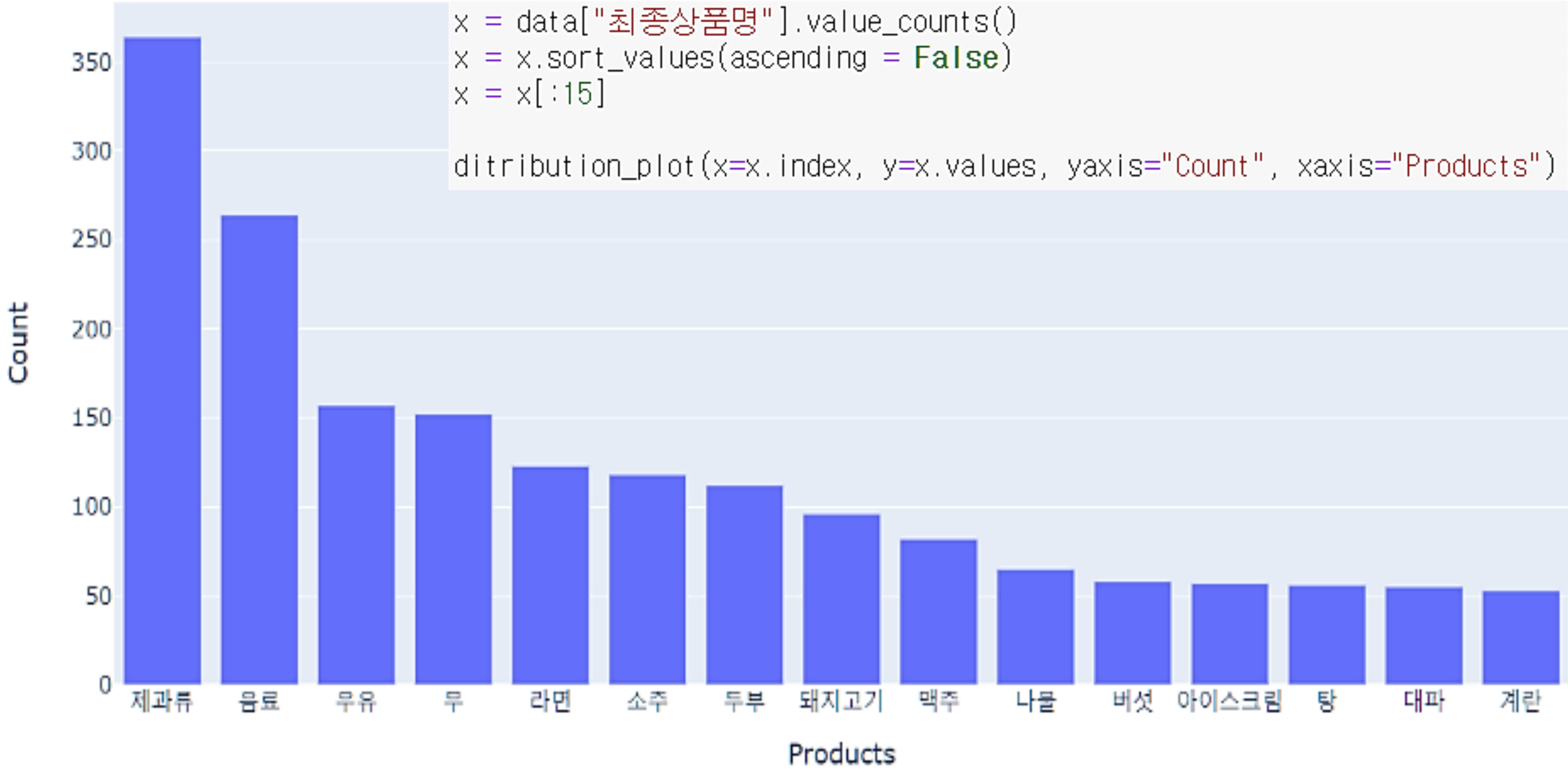
```
network_df[(network_df['lift'] >= 1) & (network_df['confidence'] >= 0.2) ]
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(라면)	(음료)	0.093333	0.217436	0.023590	0.252747	1.162399	0.003296	1.047255
2	(라면)	(제과류)	0.093333	0.232821	0.026667	0.285714	1.227187	0.004937	1.074051
4	(소주)	(맥주)	0.111795	0.082051	0.023590	0.211009	2.571674	0.014417	1.163447
5	(맥주)	(소주)	0.082051	0.111795	0.023590	0.287500	2.571674	0.014417	1.246604
7	(맥주)	(음료)	0.082051	0.217436	0.025641	0.312500	1.437205	0.007800	1.138275
9	(맥주)	(제과류)	0.082051	0.232821	0.023590	0.287500	1.234857	0.004487	1.076743
10	(소주)	(무)	0.111795	0.144615	0.023590	0.211009	1.459106	0.007422	1.084150
14	(소주)	(음료)	0.111795	0.217436	0.027692	0.247706	1.139216	0.003384	1.040238
16	(아이스크림)	(제과류)	0.048205	0.232821	0.022564	0.468085	2.010498	0.011341	1.442297
18	(우유)	(제과류)	0.153846	0.232821	0.036923	0.240000	1.030837	0.001105	1.009447
20	(음료)	(제과류)	0.217436	0.232821	0.061538	0.283019	1.215610	0.010915	1.070013
21	(제과류)	(음료)	0.232821	0.217436	0.061538	0.264317	1.215610	0.010915	1.063725

APPLICATION

Top 15 Selling Products

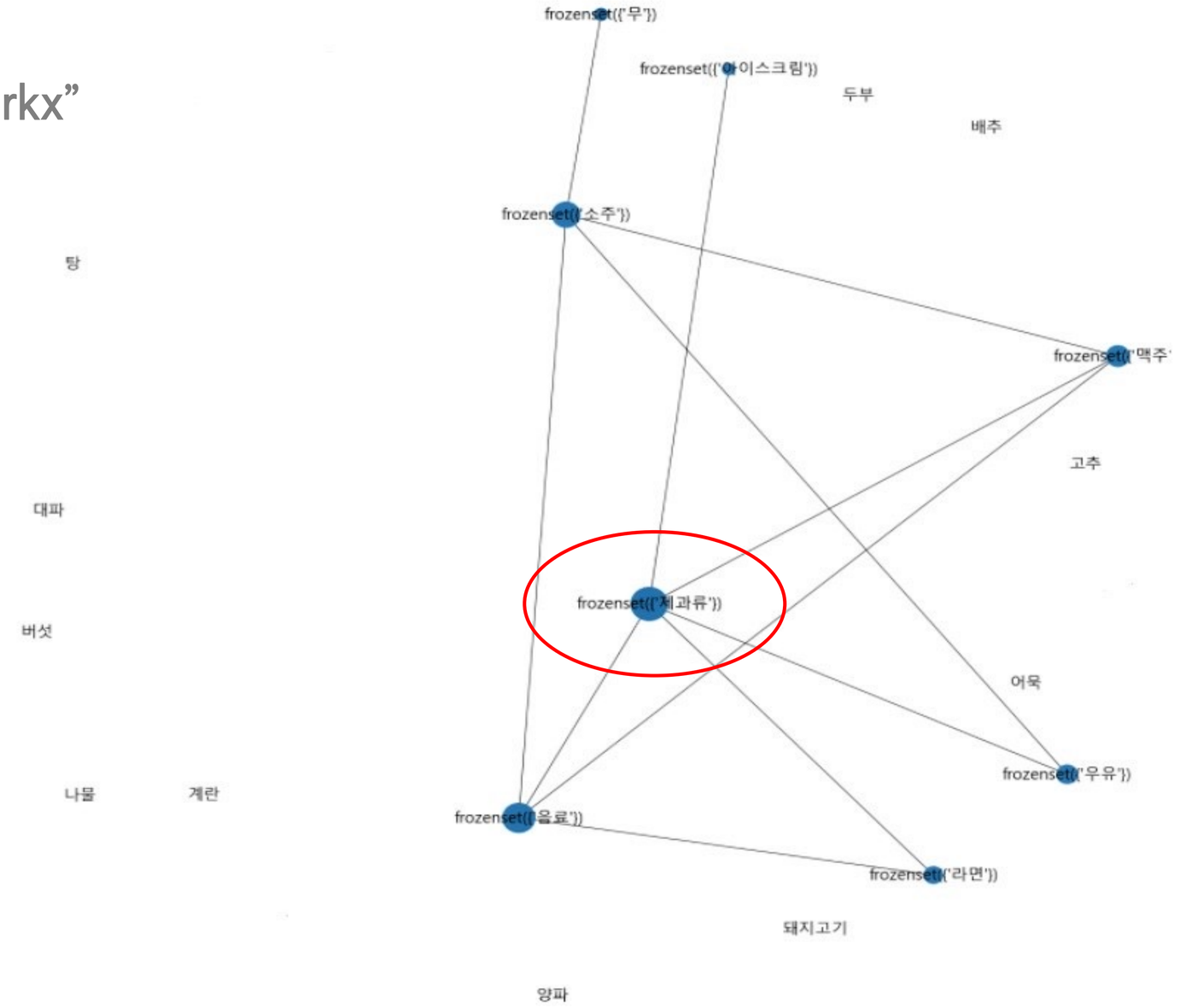
```
x = data["최종상품명"].value_counts()
x = x.sort_values(ascending = False)
x = x[:15]

ditribution_plot(x=x.index, y=x.values, yaxis="Count", xaxis="Products")
```



APPLICATION

Network Visualization using “networkx”

- Circle: node of the network graph
- Line: relationship of the network graph
- Based on **association analysis results** and **total sales of each product !**



METHOD

recommendation

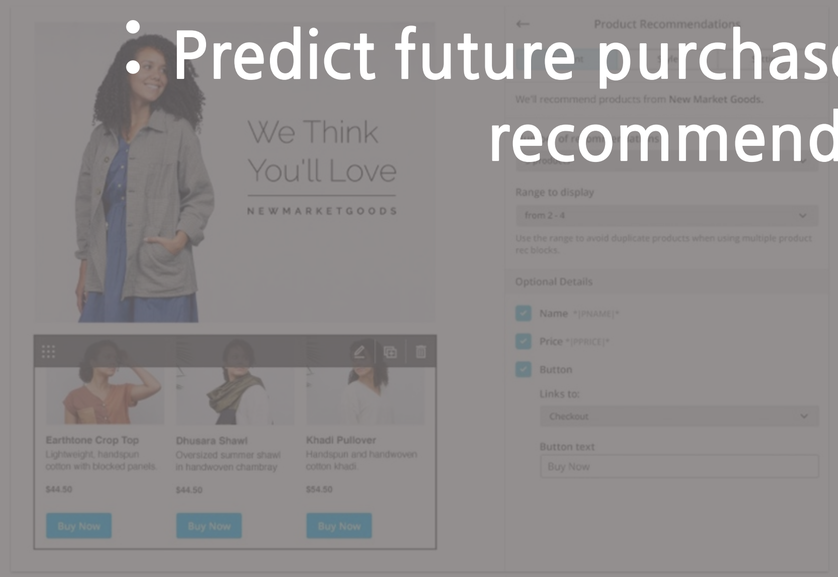
• recommendation system

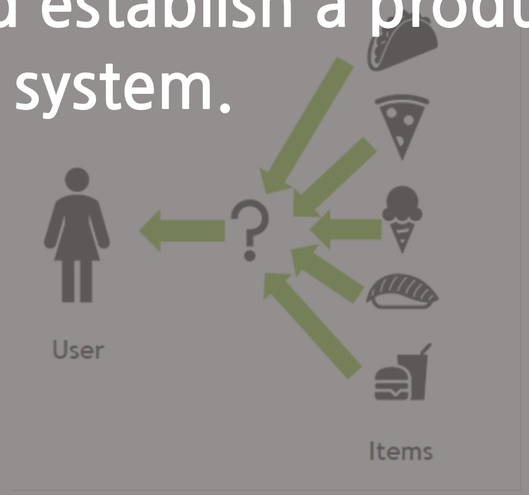
: aim to predict users' interests and recommend product items that quite likely are interesting for them. They are among the most powerful machine learning systems that retailers implement in order to drive sales.

Companies using recommender systems focus on increasing sales as a result of very personalized offers and an enhanced customer experience.

3. Recommendation

: Predict future purchases and establish a product recommendation system.





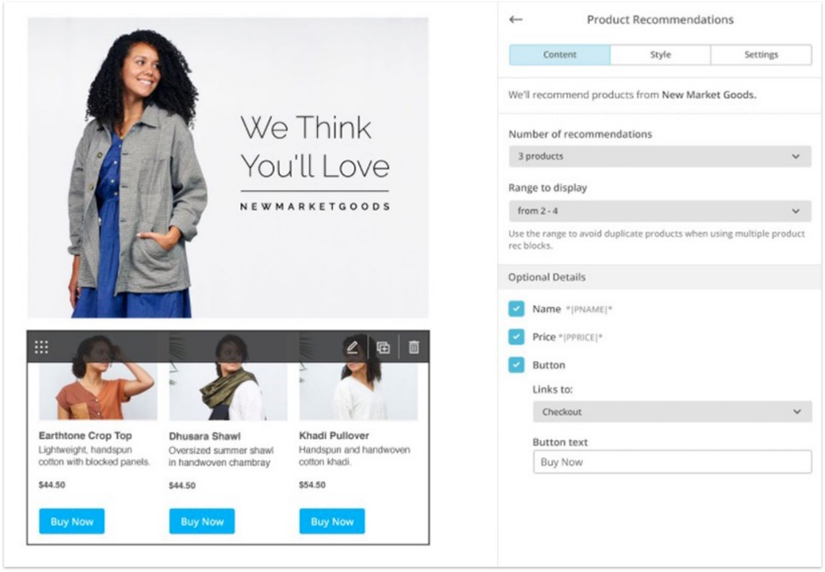
METHOD

Recommendation

• Recommendation system

: Aim to predict users' interests and recommend product items that quite likely are interesting for them. They are among the most powerful machine learning systems that retailers implement in order to drive sales.

Companies using recommender systems focus on increasing sales as a result of very personalized offers and an enhanced customer experience.



METHOD

Recommendation

•surprise package

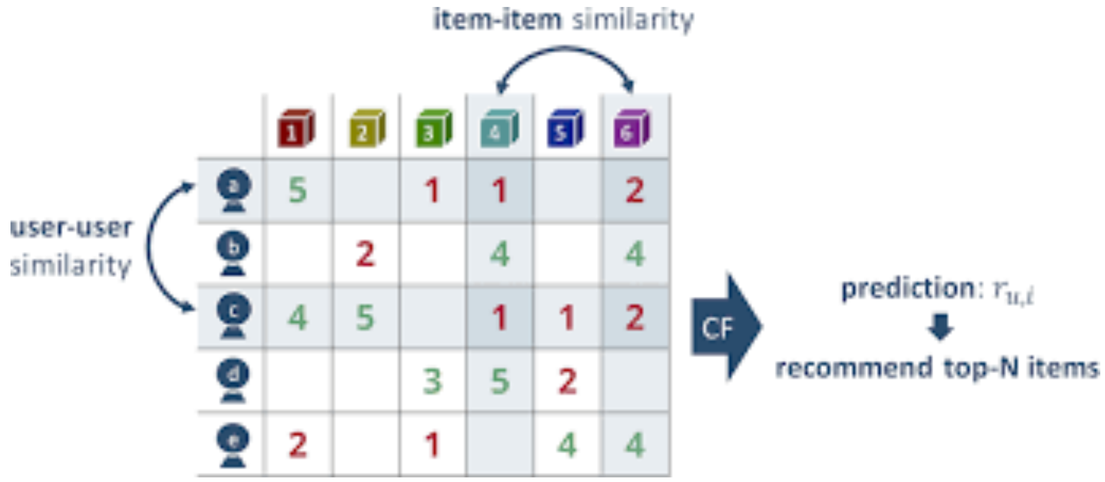
:Python scikit for building and analyzing recommender systems.

surprise

A Python scikit for recommender systems.

•Collaborative filtering

: Method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating)



APPLICATION

Recommendation

```
# ALS
print('Using ALS')
bsl_options_1 = {'method': 'als'}
algo_1 = BaselineOnly(bsl_options=bsl_options_1)
```

```
Using ALS
```

```
cross_validate(algo_1, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

```
Estimating biases using als...
Estimating biases using als...
Estimating biases using als...
Estimating biases using als...
Estimating biases using als...
Evaluating RMSE, MAE of algorithm BaselineOnly on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9644  0.8862  1.1312  1.0812  1.4787  1.1084  0.2042
MAE (testset)     0.4141  0.4106  0.4460  0.4144  0.4955  0.4361  0.0323
Fit time          0.00    0.00    0.00    0.00    0.00    0.00    0.00
Test time         0.00    0.00    0.00    0.00    0.00    0.00    0.00
```

```
# SGD
print('Using SGD')
bsl_options_2 = {'method': 'sgd'}
algo_2 = BaselineOnly(bsl_options=bsl_options_2)
```

```
Using SGD
```

```
cross_validate(algo_2, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

```
Estimating biases using sgd...
Estimating biases using sgd...
Estimating biases using sgd...
Estimating biases using sgd...
Estimating biases using sgd...
Evaluating RMSE, MAE of algorithm BaselineOnly on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.9110  1.3562  1.2990  1.1264  0.8632  1.1112  0.1985
MAE (testset)     0.4288  0.4948  0.4691  0.3902  0.3938  0.4354  0.0412
Fit time          0.01    0.01    0.01    0.01    0.01    0.01    0.00
Test time         0.00    0.00    0.00    0.00    0.00    0.00    0.00
```

```
# KNN
print('using KNN')
algo_3 = KNNBasic()
```

```
using KNN
```

```
cross_validate(algo_3, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

```
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Computing the msd similarity matrix...
Done computing similarity matrix.
Evaluating RMSE, MAE of algorithm KNNBasic on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.8907  0.9864  1.3134  1.1875  1.6239  1.2004  0.2585
MAE (testset)     0.3180  0.3580  0.4198  0.4377  0.4664  0.4000  0.0543
Fit time          0.02    0.02    0.02    0.02    0.02    0.02    0.00
Test time         0.03    0.03    0.03    0.04    0.04    0.03    0.00
```

Parameter tuning

mean_test_mae	std_test_mae	rank_test_mae	mean_fit_time	std_fit_time	mean_test_time	std_test_time	params	param_bsl_options	param_k
0.446746	0.027134	11	0.016296	0.000465	0.034929	0.000845	{'bsl_options': {'method': 'als', 'reg': 1}, ...	{'method': 'als', 'reg': 1}	2
0.437805	0.027972	5	0.015498	0.000409	0.036686	0.000291	{'bsl_options': {'method': 'als', 'reg': 1}, ...	{'method': 'als', 'reg': 1}	3
0.437538	0.029731	3	0.016364	0.000572	0.037462	0.001363	{'bsl_options': {'method': 'als', 'reg': 1}, ...	{'method': 'als', 'reg': 1}	4
0.434426	0.030408	1	0.015795	0.000438	0.038035	0.000671	{'bsl_options': {'method': 'als', 'reg': 1}, ...	{'method': 'als', 'reg': 1}	5
0.446746	0.027134	12	0.015470	0.000401	0.034581	0.000476	{'bsl_options': {'method': 'als', 'reg': 2}, ...	{'method': 'als', 'reg': 2}	2
0.437805	0.027972	6	0.016161	0.000614	0.035929	0.000749	{'bsl_options': {'method': 'als', 'reg': 2}, ...	{'method': 'als', 'reg': 2}	3
0.437538	0.029731	4	0.016446	0.001233	0.038909	0.001422	{'bsl_options': {'method': 'als', 'reg': 2}, ...	{'method': 'als', 'reg': 2}	4
0.434426	0.030408	2	0.015936	0.000824	0.037961	0.000421	{'bsl_options': {'method': 'als', 'reg': 2}, ...	{'method': 'als', 'reg': 2}	5

APPLICATION

Recommendation

특정 유저에 대해 추천하는 process

```
item_name = df['최종상품명'].unique()
```

```
# user1에 대해 상품 추천
uid=1

buy_item = df[df['No']==uid]['최종상품명'].unique()
total_items = df['최종상품명'].unique()

unbuy_item = [item for item in total_items if item not in buy_item]

print(uid, '번째 유저가 산 물건의 갯수는 ',len(buy_item),'개 입니다.')
```

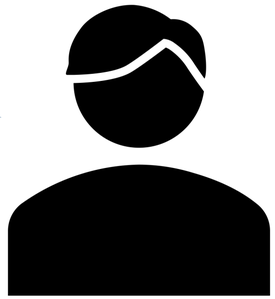
```
# user가 사지 않은 상품에 대해 살 갯수 예측
pred={}

for item in unbuy_item:

    recom = algo.predict(uid,item)
    pred[recom[1]]=recom[3]
```

```
pred_sorted = sorted(pred.items(),
                     reverse=True,
                     key=lambda item:item[1])
```

```
# 상품추천
for key, value in pred_sorted[:10]:
    print(key,'를 ',value,'개 살 것으로 예상됩니다.')
```

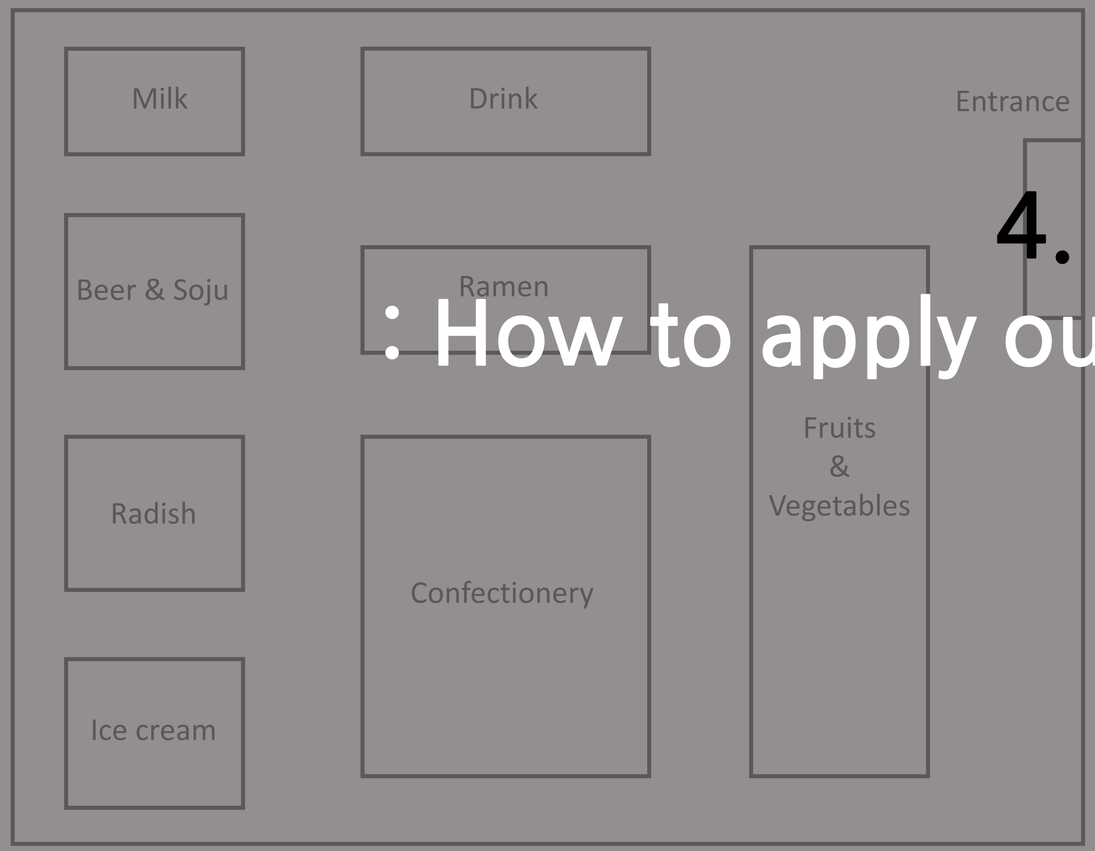
```
붕 를  6.3814796922147 32 개 살 것으로 예상됩니다.
식품 를  5.243953088459811 개 살 것으로 예상됩니다.
속옷 를  4.422806698022963 개 살 것으로 예상됩니다.
귤 를  3.173309571337812 개 살 것으로 예상됩니다.
참치 를  2.7846524359282867 개 살 것으로 예상됩니다.
비트 를  2.7355514809268993 개 살 것으로 예상됩니다.
굴 를  2.6081415462868778 개 살 것으로 예상됩니다.
누룽지 를  2.5357365305955932 개 살 것으로 예상됩니다.
스티커 를  2.508117562623645 개 살 것으로 예상됩니다.
```

안녕하세요, 고객님.
고객님의 과거 데이터를 기반으로
물, 식품, 속옷, 귤, 참치를 추천드립니다!



RESULT

Product Placement & Sales Strategies



< Improved product placement based on the network graph >

Sales Strategies

1. Focus on the best-selling products and prepare enough stock for these items.

2. Display the products at 130-140cm on the floor where the eyes are focused.

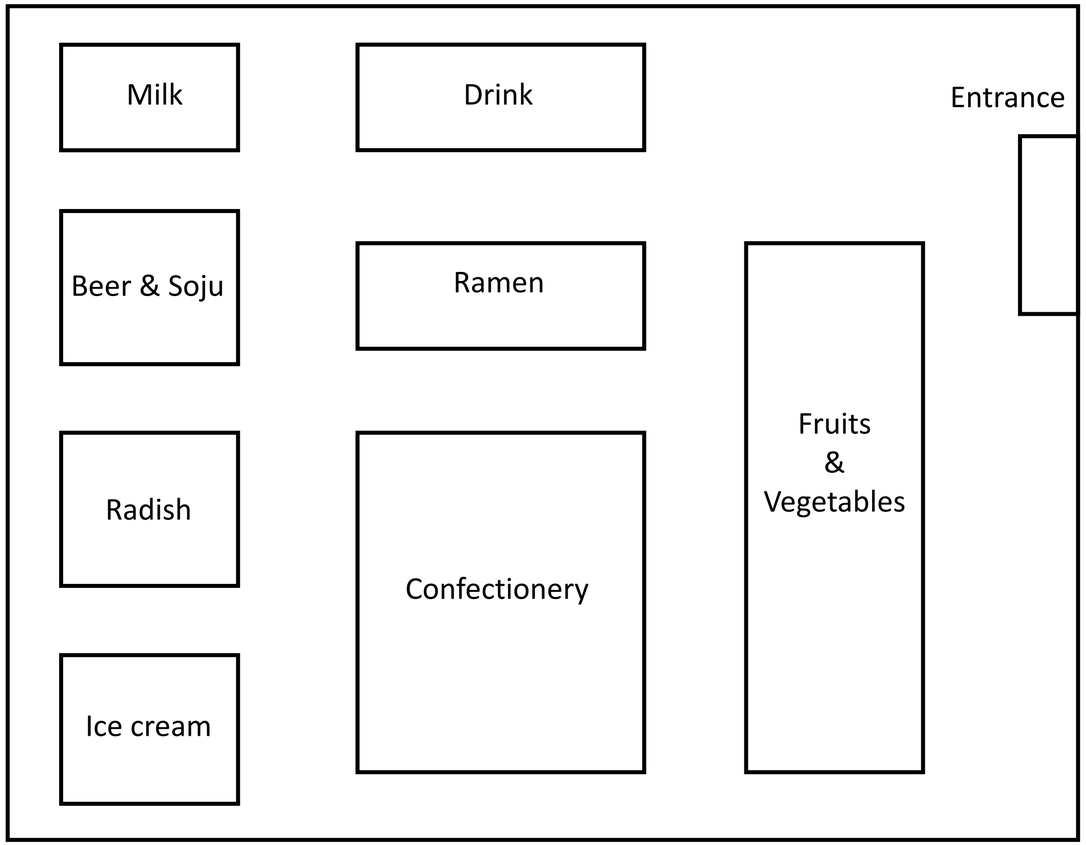
3. Display the best-selling products in the back of the store to attract customers to every corner of the store.

4. Display the products that are difficult to manage inventory such as fruits and vegetables near the entrance and increase the sales rate by attracting interest.

4. Result
: How to apply our results in real world

RESULT

Product Placement & Sales Strategies



< Improved product placement based on the network graph >

Sales Strategies

1. Focus on the best-selling products and prepare enough stock for these items.

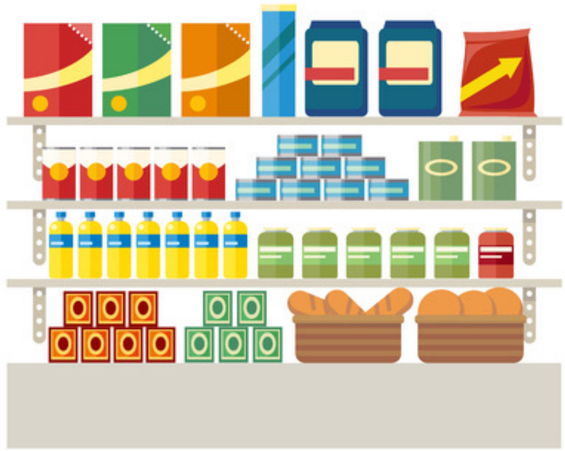
2. Display the products at 130-140cm on the floor where the eyes are focused.

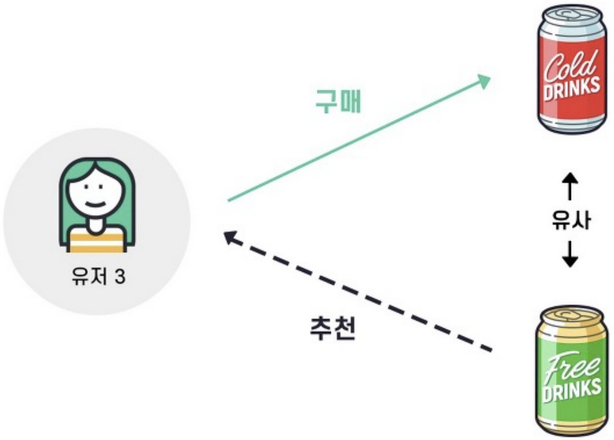
3. Display the best-selling products in the back of the store to attract customers to every corner of the store.

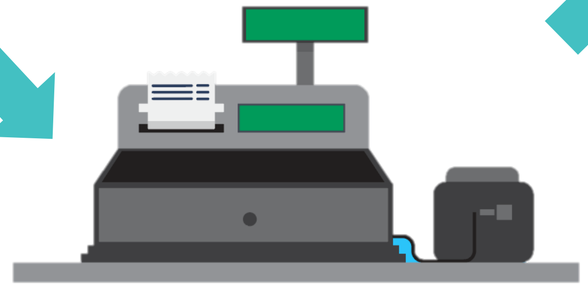
4. Display the products that are difficult to manage inventory such as fruits and vegetables near the entrance and increase the sales rate by attracting interest.

RESULT

Product Recommendation







THANK YOU
Do you have any question ?