# 마케터를 위한 데이터분석

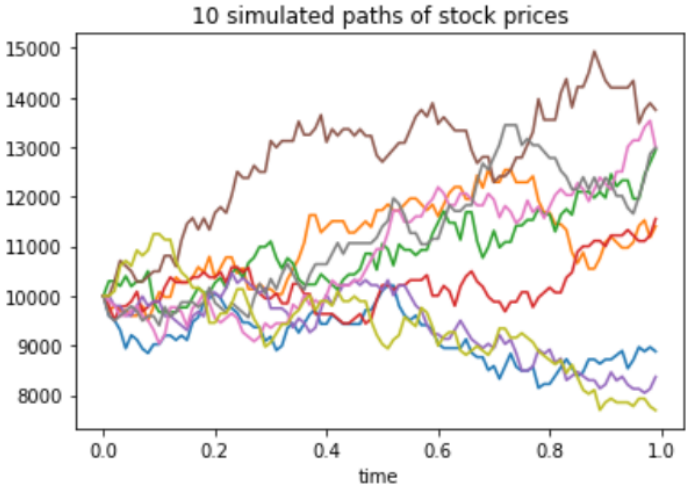
김우찬

1번

코드

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

s0, mu, sigma, cT=10000,0.05,0.2,0.01
M=100
i=10

S= np.zeros([i,M])
t=np.arange(0,1,0.01)

for y in range(0,i-1):
        S[y,0]=s0
        for x in range(0,M-1):
                epsilon=np.random.randn()
                S[y,x+1]=S[y,x]*np.exp((mu-
0.5*sigma**2)*cT+sigma*epsilon*np.sqrt(cT)).round(2)
        plt.plot(t,S[y])

plt.title('10 simulated paths of stock prices')
plt.xlabel('time')
plt.show()
```

코드의 실행결과



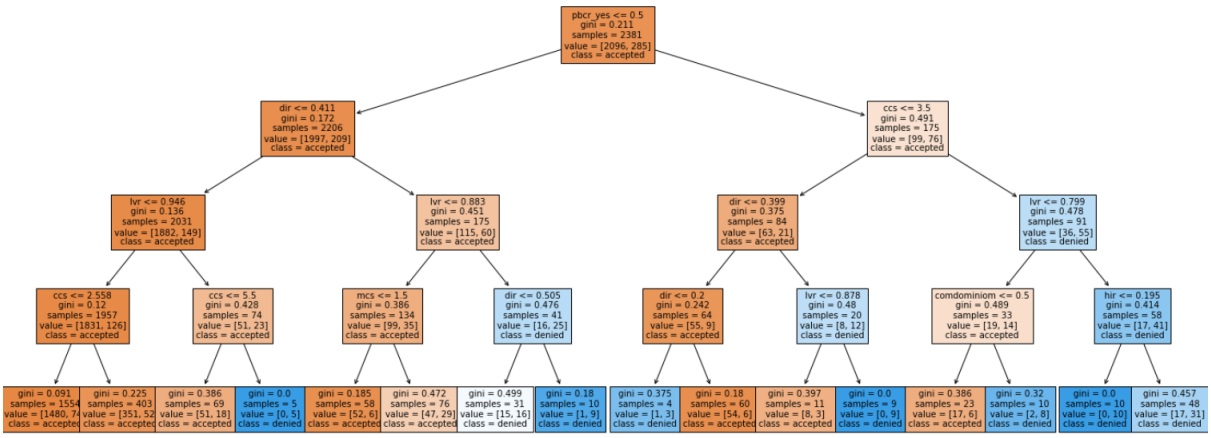
2번

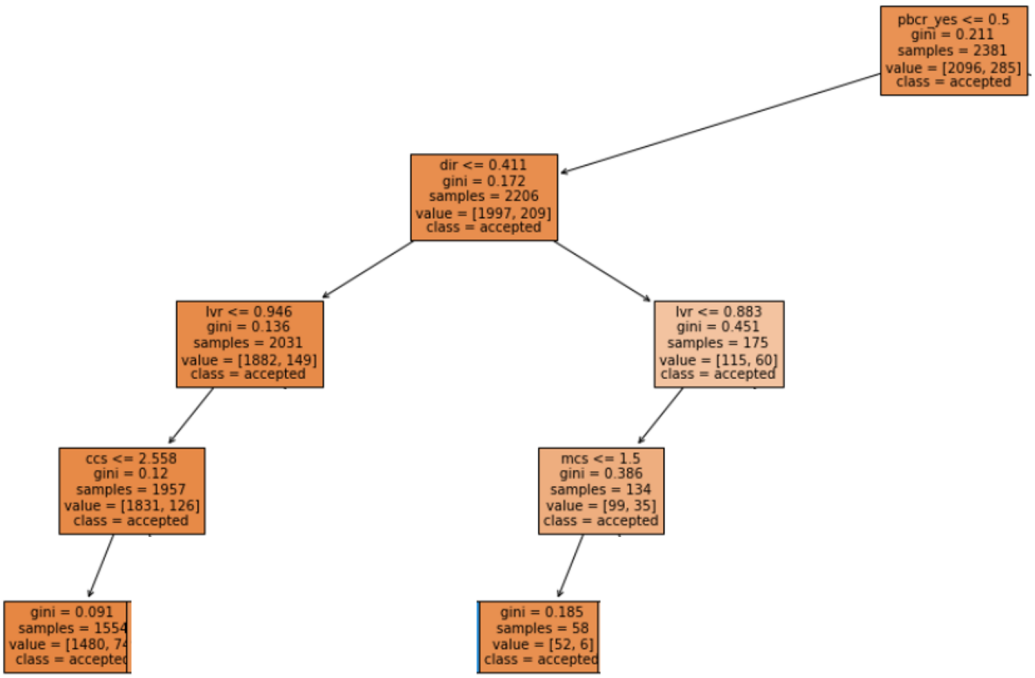
2-1)

코드

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
import matplotlib.pyplot as plt

df=pd.read_excel('C:/data/final/mortgage.xlsx')
df_1=df[['pbcr','self','single','black']].astype(str)
df_2=df.drop(['pbcr','self','single','black'],axis=1)
df_3=pd.concat([df_1,df_2],axis=1)
df_dummy=pd.get_dummies(data=df_3,columns=['pbcr','self','single','black'])

X=df_dummy.drop('deny',axis=1)
y=df_dummy['deny']


model=DecisionTreeClassifier(criterion='gini',max_depth=4)
result=model.fit(X,y)

fig=plt.figure(figsize=(25,10))
a=plot_tree(result,class_names=['accepted','denied'], feature_names=X.columns, filled=True, fontsize=10)
```

코드의 실행결과



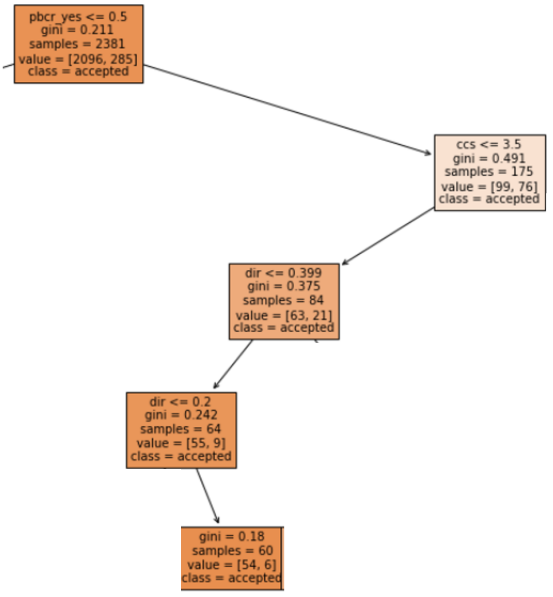



**대출신청이 승인된 고객들의 특성**

**(*accepted에 해당되는 leaf node 중 gini<0.2에 해당돼 뚜렷한 특성을 보여줄 수 있는 3개의 leaf node로 대출신청이 승인된 고객들의 특성을 분석했습니다.)**

(1) 신용불량기록(pbcr)이 없으며, 총소득 비율대비 부채상환액(dir)이 40%보다 낮으며, 부동산 평가액 대비 대출규모비율(lvr)이 94%를 넘지 않으며, 고객의 신용점수(ccs)가 2.5점 보다 낮은 고객의 주택담보대출신청은 대부분 승인됐다.

(2) 신용불량기록(pbcr)이 없으며, 총소득 비율대비 부채상환액(dir)이 40%보다 높으며, 부동산 평가액 대비 대출규모비율(lvr)이 88%를 넘지 않으며, 고객의 주택담보대출 신용점수(mcs)가 1.5점 보다 낮은 고객의 주택담보대출신청은 대부분 승인됐다.




(3) 신용불량기록(pbcr)이 있으며, 고객신용점수(ccs)가 3.5점 보다 낮으며, 총

소득 비율대비 부채상환액(dir)이 20%보다 낮은 고객의 주택담보대출신청은 대부분 승인됐다.

2-2)

코드

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics

X=df_dummy.drop('deny',axis=1)
y=df_dummy['deny']

X_train, X_test, y_train, y_test=train_test_split(X,y,test_size=0.1)
model=DecisionTreeClassifier(criterion='gini',max_depth=4)
result=model.fit(X_train,y_train)

predictions=result.predict(X_test)

accuracy=metrics.accuracy_score(y_test,predictions)
accuracy
```

코드의 실행결과

```
accuracy=metrics.accuracy_score(y_test,predictions)
accuracy
```

```
0.8870292887029289
```

2-3)

**(*2-1문제의 답에 기반해서 작성했습니다.)**

신용불량기록(pbcr)이 없고, 총소득 비율대비 부채상환액(dir)이 낮으며, 신용점수(ccs)가 1에 가까운 고객에게는 먼저 주택담보대출을 허가해 준다. 또한, 비록 총소득 비율대비 부채상환액(dir)이 좀 높더라도 부동산 평가액 대비 대출규모비율(lvr)이 약 90%를 넘지 않으며 주택담보대출 신용점수(mcs)가 1점에 가까운 고객에게도 주택담보대출을 허가해 준다. 마지막으로, 비록 신용불량기록이 있더라도 고객신용점수(ccs)가 3.5보다 낮으며 총소득 비율대비 부채상환액(dir)이 20%보다 낮은 고객에게도 주택담보대출을 허가해 준다.

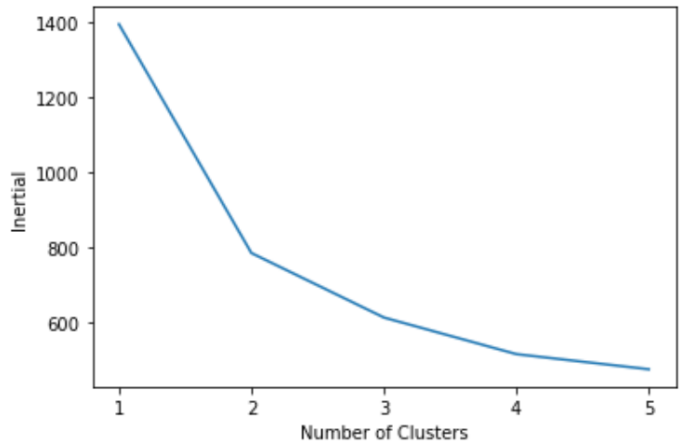
3번

코드#1 (Elbow point)

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

df=pd.read_excel("C:/data/final/Cars.xlsx")
df.dropna(axis=0,inplace=True)
```

```
Brand=df['Brand']
df.drop('Brand',axis=1,inplace=True)

scaler=StandardScaler()
df_scaled=scaler.fit_transform(df)

inertias=[]
for k in range(1,6):
        model=KMeans(n_clusters=k,random_state=0)
        result=model.fit(df_scaled)
        inertias.append(result.inertia_)

plt.plot(range(1,6),inertias)
plt.xticks(range(1,6))
plt.xlabel("Number of Clusters")
plt.ylabel("Inertial")
plt.show()
```

코드#1 실행결과(Elbow point)




: 3개 이후에 inertia의 감소폭이 현저히 적기에 Cluster 3개가 Elbow point입니다.
따라서, KMeans Clustering에서 cluster의 개수는 3개로 둡니다.

코드#2(Clustering)

(*코딩#1에 이어서 쓴 코딩입니다.)

```
model=KMeans(n_clusters=3,random_state=0)
result=model.fit(df_scaled)
cluster=result.labels_
df['cluster']=cluster
df.head()
```

코드#2 실행결과(Clustering)

| | Price | MPGcity | MPGhighway | Cylinders | EngineSize | Horsepower | RPM | Revpermile | Fueltankcapacity | Passengers | Length |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 15.9 | 25 | 31 | 4 | 1.8 | 140 | 6300 | 2890 | 13.2 | 5 | 177 |
| **1** | 33.9 | 18 | 25 | 6 | 3.2 | 200 | 5500 | 2335 | 18.0 | 5 | 195 |
| **2** | 29.1 | 20 | 26 | 6 | 2.8 | 172 | 5500 | 2280 | 16.9 | 5 | 180 |
| **3** | 37.7 | 19 | 26 | 6 | 2.8 | 172 | 5500 | 2535 | 21.1 | 6 | 193 |
| **4** | 30.0 | 22 | 30 | 4 | 3.5 | 208 | 5700 | 2545 | 21.1 | 4 | 186 |

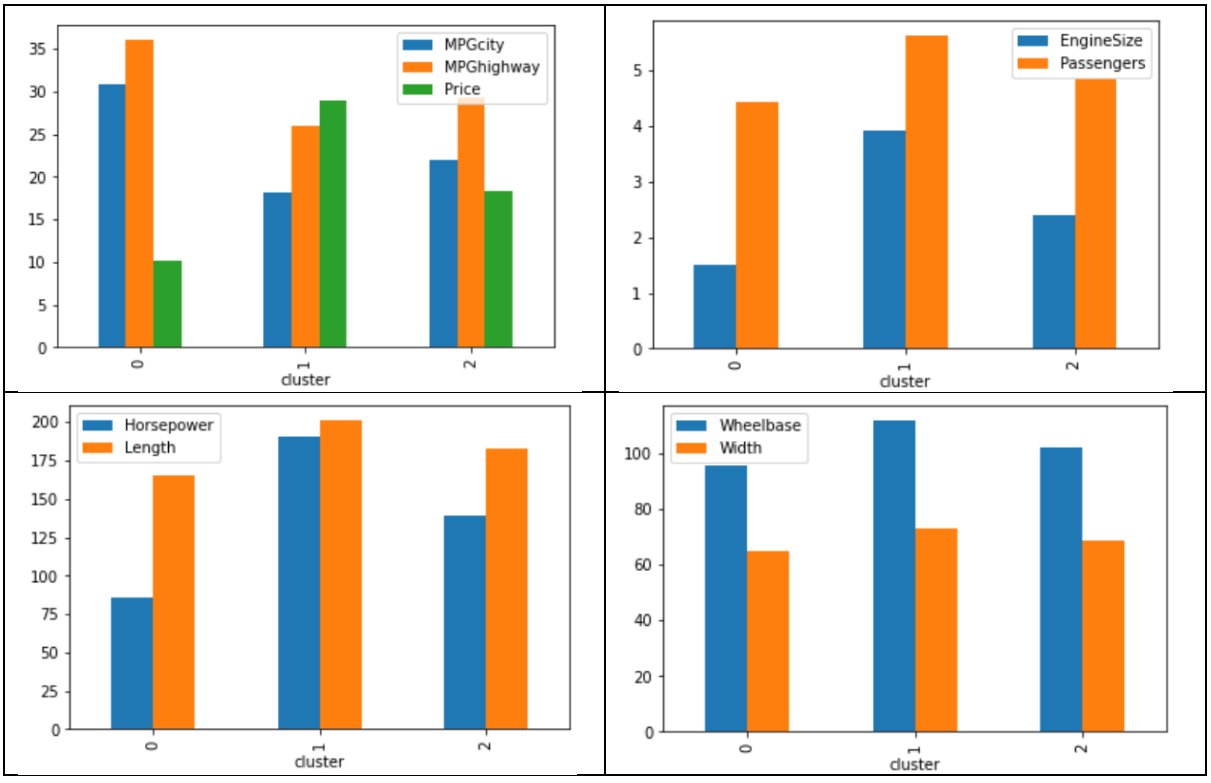
| Length | Wheelbase | Width | Turncircle | Rearseatroom | Luggageroom | Weight | cluster |
|---|---|---|---|---|---|---|---|
| 177 | 102 | 68 | 37 | 26.5 | 11.0 | 2705 | 2 |
| 195 | 115 | 71 | 38 | 30.0 | 15.0 | 3560 | 1 |
| 180 | 102 | 67 | 37 | 28.0 | 14.0 | 3375 | 2 |
| 193 | 106 | 70 | 37 | 31.0 | 17.0 | 3405 | 1 |
| 186 | 109 | 69 | 39 | 27.0 | 13.0 | 3640 | 2 |

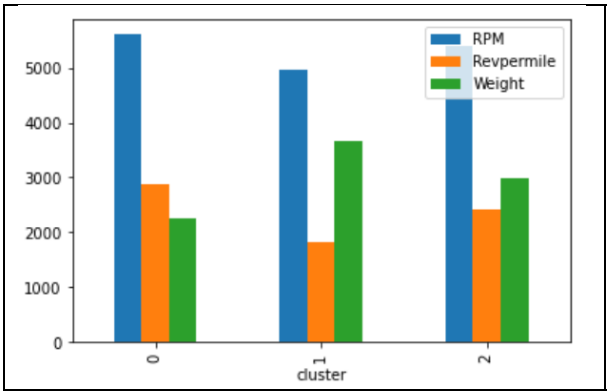
## 코드#3

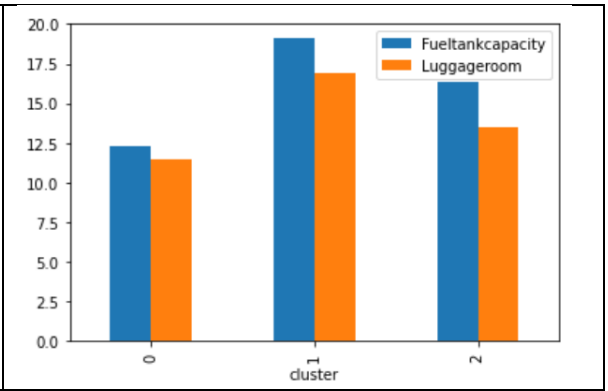
(*군집별로 자료를 시각화 할 때 사용하는 코딩은 틀이 같고, values=[‘ ’]에서 [‘ ’]에 들어가는 변수만 바꾸면 됩니다. 따라서, 밑에 제시된 ['Price','MPGcity','Passengers']의 values를 제외하고 군집별로 자료를 시각화 하는 코딩문 기입은 생략했습니다. 또한, 데이터를 막대그래프로 그릴 때 수치가 비슷한 항목끼리 묶어서 출력했습니다.)

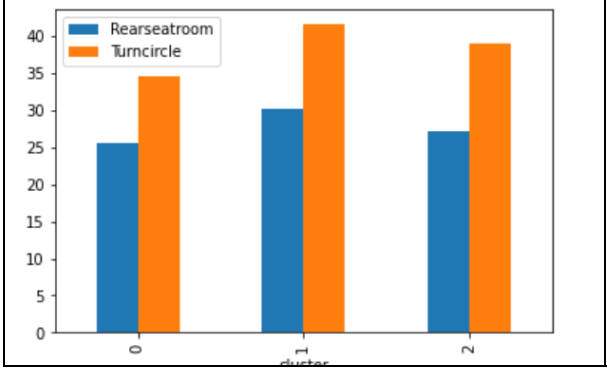
```
table=df.pivot_table(index='cluster',values=['Price','MPGcity',
'MPGhighway'],aggfunc='mean')
table.plot(kind='bar')
```

## 코드#3 실행결과






중고차 유형별 특성의 평균차이

| 특성 | Cluster0 | Cluster1 | Cluster2 |
|---|---|---|---|
| 가격 관련 특성 | | | |
| Price | $10,000 | $30,000 | $17,000 |
| 길이/크기 관련 특성 | | | |
| Length | 160 | 200 | 175 |
| Wheelbase | 90 | 110 | 100 |
| Whidth | 65 | 70 | 68 |
| Turncirle | 35 | 40 | 38 |
| Revpermile | 2,900 | 1,800 | 2,200 |
| 적재량 관련 특성 | | | |
| Fueltankcapacity | 12.5 | 19 | 17 |
| Luggaggeroom | 12 | 17 | 13 |
| Rearseatroom | 25 | 30 | 26 |
| Passengers | 4.5 | 5.5 | 5 |
| Weight | 2,100 | 3,600 | 2,900 |
| 마력 관련 특성 | | | |
| EngineSize | 1.5 | 4 | 2.5 |
| RPM | 5,500 | 4,900 | 5,200 |
| Horsepower | 85 | 185 | 135 |
| 연비 관련 특성 | | | |
| MPGcity | 30 | 17 | 23 |
| MPGhighway | 36 | 25 | 30 |

Cluster0: 소형차에 속하며 적재량과 마력은 낮지만, 연비가 좋고 가격은 저렴한 유형이다.
Cluster1: 대형차에 속하며 적재량과 마력이 높지만, 연비가 좋지 않고 가격은 비싼 유형이다.
Cluster2: 중형차에 속하며 적재량과 마력과 연비와 가격이 모두 높지도 낮지도 않은 유형이다.

4번

코드

```
import pandas as pd
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules

df=pd.read_excel("C:/data/final/supermarket.xlsx")
basket=df.groupby(['Member_number','Product'])['Product'].count()
basket_1=basket.unstack(level=1)
basket_1.fillna(0,inplace=True)
basket_2=basket_1.applymap(lambda x:1 if x>=1 else 0)
frequent_itemsets=apriori(basket_2, min_support=0.02, use_colnames=True)
rules=association_rules(frequent_itemsets, metric="lift", min_threshold=1.7)
rules
```

코드의 실행결과

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction |
|---|---|---|---|---|---|---|---|---|---|
| 0 | (yogurt, bottled water) | (whole milk, other vegetables) | 0.066444 | 0.191380 | 0.022063 | 0.332046 | 1.735009 | 0.009346 | 1.210593 |
| 1 | (whole milk, other vegetables) | (yogurt, bottled water) | 0.191380 | 0.066444 | 0.022063 | 0.115282 | 1.735009 | 0.009346 | 1.055201 |
| 2 | (yogurt, other vegetables) | (whole milk, sausage) | 0.120318 | 0.106978 | 0.023089 | 0.191898 | 1.793806 | 0.010217 | 1.105085 |
| 3 | (whole milk, sausage) | (yogurt, other vegetables) | 0.106978 | 0.120318 | 0.023089 | 0.215827 | 1.793806 | 0.010217 | 1.121796 |
| 4 | (whole milk, sausage) | (yogurt, rolls/buns) | 0.106978 | 0.111339 | 0.022832 | 0.213429 | 1.916929 | 0.010921 | 1.129791 |
| 5 | (sausage, rolls/buns) | (yogurt, whole milk) | 0.082350 | 0.150590 | 0.022832 | 0.277259 | 1.841148 | 0.010431 | 1.175261 |
| 6 | (yogurt, whole milk) | (sausage, rolls/buns) | 0.150590 | 0.082350 | 0.022832 | 0.151618 | 1.841148 | 0.010431 | 1.081648 |
| 7 | (yogurt, rolls/buns) | (whole milk, sausage) | 0.111339 | 0.106978 | 0.022832 | 0.205069 | 1.916929 | 0.010921 | 1.123396 |

코드결과에 따른 Display 전략

**(*A를 산 사람은 B를 살 확률이 높다는 문장은 B를 산 사람은 A를 살 확률이 높다는 뜻도 동시에 됩니다.)**

(1) yogurt와 bottled water을 같이 사는 사람은 milk와 vegetables도 동시에 같이 살 확률이 높다. 따라서, yogurt와 bottled water을 진열하는 곳 근처에 milk와 vegetables도 같이 진열하면 좋다.
(2) yogurt와 vegetables를 같이 사는 사람은 milk와 sausage도 같이 살 확률이 높다. 따라서, yogurt와 vegetables을 진열하는 곳 근처에 milk와 sausage도 같이 진열하면 좋다.
(3) milk와 sausage를 같이 사는 사람은 yogurt와 rolls/buns도 같이 살 확률

이 높다. 따라서, milk와 sausage를 진열하는 곳 근처에 yogurt와 rolls/buns도 같이 진열하면 좋다.

(4) sausage와 rolls/buns를 같이 사는 사람은 yogurt와 milk도 같이 살 확률이 높다. 따라서, sausage와 rolls/buns을 진열하는 곳 근처에 yogurt와 milk도 같이 진열하면 좋다.

(5) yogurt와 milk를 사는 사람은 sausage와 rolls/buns를 살 확률이 높다. 따라서, yogurt와 milk을 진열하는 곳 근처에 sausage와 rolls/buns도 같이 진열하면 좋다.