선택 문제 1, 2

<개요>

회사에서 새롭게 런칭한 디지털 콘텐츠 상품을 고객들을 대상으로 판매하는 방안을 데이터를 바탕으로 수립하고자 합니다. 구체적으로는, 1) 고객들의 온라인 리뷰 데이터를 바탕으로 적절한 고객群을 발견하여 고객들에 대한 이해를 하고, 2) 고객들이 생성했던 제품 구매 Transaction 자료를 바탕으로 판매 패턴을 발견합니다. 3) 이후, 새로운 제품의 판매를 위한 프로모션 데이터를 바탕으로 효과적인 홍보 방안을 찾고자 합니다. 마지막으로 4) 프로모션 후 생성된 텍스트 데이터를 바탕으로 제품 판매 활동에 대한 평가를 하고자 합니다.

<대상 데이터를 확인해보세요>

온라인 리뷰 데이터: data_1_2.csv
제품 구매 이력 데이터: data_3.csv
ABC 社 판촉 데이터: data_4.csv
고객 피드백 데이터: data_5.csv, data_5_2.csv

<선택문제 1: 비지도학습>

1-1. 고객群발견을 해보세요.

1~5 번 변수에 대하여, K-means 클러스터링을 통해 고객 군집을 발견하세요. 이때 데이터 크기를 최소-최대 정규화를 적용하여 클러스터링하세요. 반복문을 사용하시어, K 가 2,4,6,8 일때의 Between Sum of Squares(모형의 inertia)를 비교해보세요. 그리고 k=8 에서의 각 군집의 특성을 파악해보세요

```
from pandas import read_csv
data = read_csv("data_1_2.csv")
csv_data = data.iloc[:,0:5]
print(csv_data.shape)

from sklearn import preprocessing
min_max_scaler = preprocessing.MinMaxScaler()
scaled = min_max_scaler.fit_transform(csv_data)


from sklearn.cluster import KMeans
for numcl in [2,4,6,8]:
        kmeans = KMeans(n_clusters=numcl)
        kmeans.fit(scaled)
        print(kmeans.inertia_)

kmeans = KMeans(n_clusters=8)
```

```
kmeans.fit(scaled)
kmeans.cluster_centers_
```

1-2. 기존 제품의 판매 패턴을 mlxtend 모듈을 사용해서 발견해보세요.

최소 지지도는 0.01, 최소 리프트는 2 로 지정해보세요. 결과를 리프트가 3 이상이고, 컨피던스가 0.9 이상인 패턴을 발견해보세요.

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

df = pd.read_csv('data_3.csv')
df.head()

df['artist'] = df['artist'].str.strip()      #Description 컬럼의 문자열 전후 공백 정리
df.dropna(axis=0, subset=['user'], inplace=True)      #InvoiceNo 컬럼의 NA 처리
df['user'] = df['user'].astype('str')                #InvoiceNo 을 문자열로 변경

basket=df.groupby(['user',
'artist'])['qty'].sum().unstack().reset_index().fillna(0).set_index('user')

def encode_units(x):
    if x <= 0:
        return 0
    if x >= 1:
        return 1

basket_sets = basket.applymap(encode_units)
basket_sets.columns

frequent_itemsets = apriori(basket_sets, min_support=0.01, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=2)
rules.head()

rules[ (rules['lift'] >= 3) & (rules['confidence'] >= 0.9) ]
```

<선택문제 2: 회귀와 분류>

2-1. 효과적인 홍보를 위한 의사결정을 위해, 판촉 자료에 대해 회귀분석하세요. 회귀분석 결과를 바탕으로 유효한 홍보 매체를 발견해보세요.

Y 변수는 sales, X 변수는 SMS, TV, News 를 선택하세요. 자료를 train, test 로 7:3 으로 나누어서, train 으로 발견한 모형을 test 에 적용하여 rmse 를 구하세요.
from statsmodels.tools.eval_measures import rmse
rmse(예측값, testing 셋의 y)

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from statsmodels.tools.eval_measures import rmse
from sklearn import linear_model

df_adv = pd.read_csv('data_4.csv')

X = df_adv.loc[:,['SMS', 'TV', 'News']]
y = df_adv['sales']

X_train, X_test, Y_train, Y_test = train_test_split( X, y, test_size=0.3,
random_state=42)

regr = linear_model.LinearRegression()
regr.fit(X_train, Y_train)
regr.score(X_train, Y_train)    #R^2
regr.coef_
rmse(regr.predict(X_test), Y_test)
```

2-2. 판촉에 대한 고객의 피드백을 바탕으로, 고객 피드백을 통해 판촉에 대한 긍부정 반응을 분류하는 모형을 만드려고 합니다.

data_5.csv 에서는 고객의 피드백이 텍스트로 기록되어 있습니다. 텍스트마이닝을 거쳐, 중빈도(10~30)에 해당하는 단어들로 구성된 DTM 이 data_5_2.csv 로 제공됩니다. 해당 파일의 첫 컬럼은 고객의 피드백이 긍정(1)인지 부정(0)인지 기록되어 있습니다. 이 데이터를 파티셔닝한 후에, 고객 피드백이 긍정인지를 분류하는 모형으로 적용해보세요. 수업시간에 학습하신 분류모형 중 최적의 모형을 찾아보세요.

```
import pandas as pd
import numpy as np
from sklearn import preprocessing, model_selection
from sklearn import tree, neighbors
from sklearn.svm import SVC

data = pd.read_csv("data_5_2.csv")
variables = data.columns[1:58]
csv_data = data.values
csv_data.shape
y   = csv_data[:, 0 ]
X = csv_data[:, 1:58 ]

#Partitioning
X_train, X_test, y_train, y_test = model_selection.train_test_split( X, y, test_size=0.3,
random_state=42 )

dt1 = tree.DecisionTreeClassifier()
dt1 = dt1.fit(X_train, y_train)
pred = dt1.predict( X_test)
np.mean( pred == y_test )

knn = neighbors.KNeighborsClassifier()
knn.fit(X_train, y_train)
knn.kneighbors(X_test, return_distance=False)
predicted = knn.predict(X_test)
np.mean( predicted == y_test )

clf = SVC()
clf.fit(X_train, y_train)
clf.n_support_
predicted = clf.predict(X_test)
np.mean( predicted == y_test )
```