과제 보고서

제목 : 파이썬 4차과제 – 레스토랑

과 목 명: 파이썬 통계분석
제출일자: 2022.12.01.
학 과: 정보통계학과
학 번: 2018015027
이 름: 김한탁

충북대학교
CHUNGBUK NATIONAL UNIVERSITY

1. 데이터를 불러와서 rest 데이터프레임 생성

답안)
1) rest 데이터프레임 생성

1-1) 코드 및 결과

```
In [3]:
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#실습1
rest=pd.read_csv("레스토랑.csv", encoding="CP949")
rest.head()
```

Out[3]:

	실내시설	메뉴	가격	서비스	편리성	인지도	위생	숙박	이용횟수	레스토랑	지불비용	인원	동반	형태	만족도	재구매	성별	연령	최종학력	소득
0	3	5	4	4	3	4	4	3	5	3	40	2	1	2	4	3	2	51	3	2
1	3	5	5	4	5	4	3	2	2	2	30	1	2	1	3	4	1	48	4	1
2	2	3	3	4	3	4	3	2	2	2	40	3	3	3	3	3	2	48	2	4
3	4	5	3	4	4	3	5	4	3	2	20	2	2	1	4	4	1	34	4	3
4	5	5	4	4	4	3	5	4	7	1	30	2	3	2	4	4	2	35	5	2

2) 데이터프레임 정보 확인

2-1) 코드 및 결과

```
In [2]:
rest.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 221 entries, 0 to 220
Data columns (total 20 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   실내시설    221 non-null    int64
 1   메뉴      221 non-null    int64
 2   가격      221 non-null    int64
 3   서비스     221 non-null    int64
 4   편리성     221 non-null    int64
 5   인지도     221 non-null    int64
 6   위생      221 non-null    int64
 7   숙박      221 non-null    int64
 8   이용횟수    221 non-null    int64
 9   레스토랑    221 non-null    int64
 10  지불비용    221 non-null    int64
 11  인원      221 non-null    int64
 12  동반      221 non-null    int64
 13  형태      221 non-null    int64
 14  만족도     221 non-null    int64
 15  재구매     221 non-null    int64
 16  성별      221 non-null    int64
 17  연령      221 non-null    int64
 18  최종학력    221 non-null    int64
 19  소득      221 non-null    int64
dtypes: int64(20)
memory usage: 34.7 KB
```

2-2) 결과해석

총 20개의 변수가 있으며, 자료의 수는 총 221개이다.
또한 20개의 자료형 모두 int형임을 확인할 수 있다.

2. 실내시설 ~ 숙박 : 8개의 변수를 토대로 계층적 군집분석을 수행하고 cluster 개수를 정하라.

답안)

1) 실내시설 ~ 숙박 8개의 변수를 독립변수로 선택

1-1) 코드 및 결과

```
In [5]:
#실습2
# 8개 변수(실내시설 ~ 숙박) 선택
rest_x=rest.iloc[:,0:8]
rest_x.head()
```

Out [5]:

	실내시설	메뉴	가격	서비스	편리성	인지도	위생	숙박
0	3	5	4	4	3	4	4	3
1	3	5	5	4	5	4	3	2
2	2	3	3	4	3	4	3	2
3	4	5	3	4	4	3	5	4
4	5	5	4	4	4	3	5	4

2) 계층적 군집분석 수행

2-1) 코드 및 결과

```
In [4]:
# 계층적 군집분석 model
# method=complete(완전결합방식)
from scipy.cluster.hierarchy import linkage, dendrogram
import matplotlib.pyplot as plt
clusters = linkage(y=rest_x, method='complete', metric='euclidean')
clusters
clusters.shape
```

Out [4]: (220, 4)

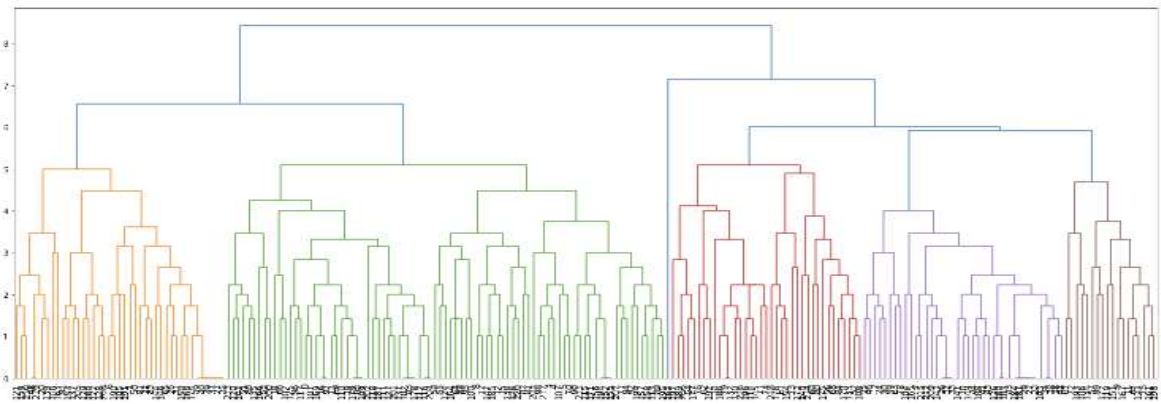
2-2) 결과해석

유클리드 거리를 이용한 완전결합방식을 통해, 계층적 군집분석을 수행하였다.

3) 덴드로그램의 시각화와 군집의 수 결정

3-1) 코드 및 결과

```
In [5]:
# 덴드로그램 시각화 : 군집수 결정
import matplotlib.pyplot as plt
plt.figure( figsize = (25, 10) )
dendrogram(clusters, leaf_rotation=90, leaf_font_size=12)
# leaf_rotation=90 : 글자 각도
# leaf_font_size=20 : 글자 사이즈
plt.show()
```



3-2) 결과해석

덴드로그램의 시각화를 통해, 결정지어지는 군집의 수는 크게 6개로 인식될 수 있다.

따라서 군집의 수를 6개로 결정하였다.

4) 계층적 군집분석 결과

4-1) 코드 및 결과

```
In [6]: # 클러스터링(군집) 결과
        from scipy.cluster.hierarchy import fcluster # 지정한 클러스터 자르기
        cut_tree = fcluster(clusters, t=5.5, criterion='distance')
        cut_tree
```

```
Out[6]: array([2, 1, 4, 2, 2, 2, 1, 4, 2, 5, 4, 5, 2, 2, 3, 2, 2, 2, 2, 2, 1, 1,
               4, 4, 4, 4, 1, 1, 4, 4, 4, 1, 4, 4, 4, 1, 4, 1, 4, 1, 1, 4, 4, 4,
               4, 4, 4, 5, 1, 1, 1, 2, 3, 2, 3, 2, 3, 5, 2, 4, 3, 3, 3, 4, 2, 1,
               2, 5, 3, 2, 2, 3, 5, 3, 3, 5, 3, 2, 4, 3, 3, 1, 2, 2, 3, 1, 2, 2,
               2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 5, 2, 2, 2, 4, 4, 1, 5, 2, 3, 2,
               2, 2, 2, 2, 3, 2, 2, 2, 2, 5, 2, 1, 1, 5, 2, 2, 2, 2, 1, 1, 3, 2,
               3, 3, 5, 2, 4, 1, 2, 1, 3, 4, 2, 4, 1, 3, 1, 2, 1, 4, 1, 5, 2, 1,
               3, 2, 2, 2, 2, 1, 1, 5, 6, 1, 2, 4, 2, 5, 1, 2, 3, 2, 5, 3, 1, 3,
               3, 3, 2, 2, 4, 4, 2, 4, 4, 2, 3, 2, 4, 3, 1, 1, 3, 5, 2, 2, 3, 3,
               5, 2, 2, 2, 2, 2, 2, 1, 3, 2, 1, 4, 1, 2, 4, 2, 3, 4, 3, 2, 3, 1,
               2], dtype=int32)
```

4-2) 결과해석

자료가 계층적 군집분석에 따라 6개의 군집으로 분류되었다.

3. 비계층적 군집분석(K-means)을 수행하고, 군집의 특징을 프로파일링하여 정리하라.(군집별로 연령, 지불비용, 소득, 만족도 등 유의한 차이가 있다는 결과가 나온 변수에 대해 정리)

답안)
1) k-means을 이용한 비계층적 군집분석

1-1) 코드 및 결과

```
In [7]: #실습3
        ## 비계층적 군집분석
        import pandas as pd
        from sklearn.cluster import KMeans
        import matplotlib.pyplot as plt
        # 비계층적 군집 분석 model : k-mean
        model= KMeans(n_clusters=6, random_state=0, algorithm='auto')
        model.fit(rest_x[0:8])
```

```
Out[7]: KMeans(n_clusters=6, random_state=0)
```

```
In [60]: pred=model.predict(rest_x)
         rest["cluster"]=pred
         pred
```

```
Out[60]: array([5, 3, 1, 0, 0, 0, 2, 4, 0, 1, 1, 4, 0, 5, 5, 4, 5, 0, 5, 4, 3, 3,
                1, 1, 1, 1, 1, 4, 2, 1, 1, 1, 4, 4, 4, 4, 4, 4, 4, 4, 4, 1, 2, 1,
                1, 5, 1, 1, 5, 4, 4, 4, 4, 1, 5, 0, 4, 1, 0, 1, 4, 1, 1, 1, 0, 3,
                0, 1, 4, 5, 0, 1, 1, 1, 1, 4, 5, 0, 1, 5, 4, 2, 0, 0, 4, 3, 4, 0,
                0, 0, 0, 5, 0, 4, 0, 0, 5, 5, 0, 1, 0, 5, 0, 2, 4, 2, 1, 0, 4, 4,
                4, 4, 5, 0, 1, 0, 5, 5, 5, 1, 4, 3, 3, 4, 0, 5, 5, 1, 3, 3, 5, 5,
                5, 4, 1, 4, 2, 3, 4, 3, 4, 2, 4, 4, 2, 1, 4, 4, 3, 1, 3, 1, 0, 3,
                1, 0, 0, 0, 0, 0, 3, 1, 4, 4, 5, 4, 3, 1, 4, 4, 4, 0, 1, 4, 5, 4,
                1, 1, 0, 4, 4, 1, 0, 4, 1, 0, 1, 5, 4, 1, 2, 2, 1, 1, 0, 0, 1, 4,
                4, 0, 1, 1, 5, 0, 5, 3, 1, 0, 3, 4, 3, 0, 4, 2, 0, 4, 5, 0, 2, 5,
                0])
```

1-2) 결과해석

실내시설 ~ 숙박의 8개 변수의 자료를 이용하여, 비계층적 군집분석 모델을 적합하고, 다시 자료에 대해 예측을 수행한 결과이다.

6개의 군집으로 분류된 결과를 cluster변수로 정하여 기존의 데이터에 추가하였다.

2) 군집 별 **연령**의 차이 확인 - 분산분석

2-1) 군집 간 등분산 검정

```
In [9]:
#실습 3-1 군집별 연령의 차이
from scipy import stats
stats.levene(rest["연령"][rest["cluster"]==0], rest["연령"][rest["cluster"]==1],
             rest["연령"][rest["cluster"]==2], rest["연령"][rest["cluster"]==3],
             rest["연령"][rest["cluster"]==4], rest["연령"][rest["cluster"]==5])
```

```
Out[9]: LeveneResult(statistic=6.820921871932536, pvalue=6.295240919554186e-06)
```

2-2) 일원배치 분산분석

```
In [10]:
# 분산분석 : bmi지수
import statsmodels.api as sm
import statsmodels.formula.api as smf
fit1=smf.ols('연령~cluster', rest).fit()
sm.stats.anova_lm(fit1, typ=1)
```

Out[10]:

	df	sum_sq	mean_sq	F	PR(>F)
cluster	1.0	266.579266	266.579266	2.819938	0.094526
Residual	219.0	20702.886797	94.533730	NaN	NaN

2-3) 군집 별 연령의 평균

```
In [11]:
rest.groupby('cluster').mean()[["연령"]]
```

Out[11]:

cluster	연령
0	32.978723
1	40.018868
2	38.000000
3	35.210526
4	38.000000
5	38.806452

2-4) 결과해석

수치형인 연령 변수에 따른 군집의 차이는 분산분석을 통해 확인할 수 있다.
따라서 분산분석을 수행하기 위해, 군집 간의 등분산 검정을 2-1)에서 수행하였고, p-value가 0.05보다 작아 군집들이 서로 다른 분산을 가진다고 판단되었다.
그럼에도 불구하고, 진행한 2-2)의 분산분석에서 F 통계량이 2.81이고, p-value가 0.09로 0.05보다 큰 것으로 나타나 군집 간 연령의 평균이 다르지 않은 것으로 확인된다.
실제로 군집 별 연령의 평균을 확인하면, 군집1이 40 정도의 값으로 평균이 가장 높고, 군집0이 33 정도로 평균이 가장 낮음을 알 수 있다.

3) 군집 별 **연령(대)**의 차이 확인 - 카이제곱 검정

3-1) 범주형으로 변환 후, 분할표

```
In [12]:
rest.loc[(rest["연령"]>=20) & (rest["연령"]<30), '연령대']='20대'
rest.loc[(rest["연령"]>=30) & (rest["연령"]<50), '연령대']='30대~40대'
rest.loc[(rest["연령"]>=50), '연령대']='50대 이상'
```

```
In [13]:
d_table=pd.crosstab(rest["cluster"], rest["연령대"])
d_table
```

Out[13]:

cluster	연령대 20대	30대~40대	50대 이상
0	10	37	0
1	11	27	15
2	4	7	2
3	5	13	1
4	14	34	10
5	5	20	6

3-2) 카이제곱 검정 결과

```
In [14]:
#카이제곱 검정결과 : 귀무가설 기각 (cluster에 따른 연령(대)의 차이가 있다.)
from scipy.stats import chi2_contingency
chi,p,df,expected=chi2_contingency(d_table)
# 카이제곱검정통계량과 유의확률
print("chisquare=%.3f" % chi, "P-value=%.3f" % p)
```

```
chisquare=19.486 P-value=0.035
```

3-3) 군집에 따른 연령(대)의 행별 비율표

```
In [15]:
# 행별 비율표(propotion table)
pd.crosstab(rest["cluster"], rest["연령대"], normalize='index').round(4)
```

Out[15]:

연령대	20대	30대~40대	50대 이상
cluster			
0	0.2128	0.7872	0.0000
1	0.2075	0.5094	0.2830
2	0.3077	0.5385	0.1538
3	0.2632	0.6842	0.0526
4	0.2414	0.5862	0.1724
5	0.1613	0.6452	0.1935

3-4) 결과해석

연령 변수를 범주형으로 변환한 연령대 변수를 생성하였다.

3-1)의 분할표를 확인해보면, 연령대 변수가 “20대”, “30대~40대”, “50대 이상”으로 3개의 범주의 가지고 있음을 확인할 수 있다.

다음으로 진행한 3-2)의 카이제곱 검정 결과는 크래머 법칙이 위배되지 않는다는 가정을 기반으로 얻어졌다. 카이제곱 검정 결과 카이제곱 통계량이 19.406의 값을 갖고, p-value가 0.035로 0.05보다 작아 군집에 따른 연령(대)의 분포 차이가 있는 것으로 나타났다.

마지막으로 군집에 따른 연령대의 행별 비율표를 살펴보면, 모든 군집이 30대~40대의 고객의 비율이 가장 높음을 알 수 있다. 각 군집에 대해 자세히 보면, 군집0은 50대 이상의 고객이 존재하지 않고, 20~40대의 고객으로만 구성되어 있음을 확인할 수 있다. 이와 비슷하게 군집3도 주로 20~40대의 고객으로 구성되어 있다.

반면 20대보다 50대 이상의 고객이 더 많은 군집도 존재하는데, 바로 군집1이다. 나머지 군집2, 4, 5는 20대와 50대 이상 고객의 구성비율이 비슷한 것으로 확인되었다.

4) 군집 별 **지불비용**의 차이-분산분석

4-1) 군집 간 등분산 검정

```
In [16]:
#실습 3-2 군집별 지불비용의 차이
from scipy import stats
stats.levene(rest["지불비용"][rest["cluster"]==0], rest["지불비용"][rest["cluster"]==1],
             rest["지불비용"][rest["cluster"]==2], rest["지불비용"][rest["cluster"]==3],
             rest["지불비용"][rest["cluster"]==4], rest["지불비용"][rest["cluster"]==5])
```

```
Out[16]: LeveneResult(statistic=1.7263568569962784, pvalue=0.12973109484576517)
```

4-2) 일원배치 분산분석

```
In [17]:
# 분산분석 : bmi지수
import statsmodels.api as sm
import statsmodels.formula.api as smf
fit1=smf.ols('지불비용~cluster', rest).fit()
sm.stats.anova_lm(fit1, typ=1)
```

Out[17]:

	df	sum_sq	mean_sq	F	PR(>F)
cluster	1.0	8.329369e+01	83.293687	0.013211	0.908599
Residual	219.0	1.380774e+06	6304.905269	NaN	NaN

4-3) 군집 별 지불비용의 평균

```
In [18]: rest.groupby('cluster').mean()[["지불비용"]]
```

Out[18]:

cluster	지불비용
0	50.659574
1	69.528302
2	106.153846
3	50.894737
4	50.327586
5	74.032258

4-4) 결과해석

연령 변수와 마찬가지로 수치형인 지불비용 변수에 따른 군집의 차이는 분산분석을 통해 확인할 수 있다.
따라서 분산분석을 수행하기 위해, 군집 간의 등분산 검정을 4-1)에서 수행하였고, p-value가 0.05보다 커 군집들이 등분산으로 가정된다.
그러므로 4-2)의 분산분석에서 F 통계량이 0.01이고, p-value가 0.908로 0.05보다 커 군집 간 지불비용의 평균이 다르지 않은 것으로 확인할 수 있다.
실제로 군집 별 지불비용의 평균을 확인하면, 군집2의 지불비용 평균이 106정도로 다른 군집보다 매우 큰 값을 가지며, 군집 0, 3, 4가 지불비용 평균 50으로 비슷하게 작은 값을 갖는 것으로 나타났다.
군집 2의 평균이 다른 군집의 평균보다 1.5배에서 2배 정도 차이가 있음에도, 분산분석의 결과가 군집별 지불비용의 평균비용의 차이가 없다고 나타난 것은 기술 통계량만으로는 확인하기 어려운 결과일 것이다.

5) 군집 별 **지불비용**의 차이-카이제곱 검정

5-1) 범주형으로 변환 후, 분할표

```
In [19]: rest.loc[(rest["지불비용"]<50), '지불비용1']='50만원 미만'
         rest.loc[(rest["지불비용"]>=50) & (rest["지불비용"]<100), '지불비용1']="50만원 이상~100만원 미만"
         rest.loc[(rest["지불비용"]>=100), '지불비용1']='100만원이상'
```

```
In [20]: d_table=pd.crosstab(rest["cluster"], rest["지불비용1"])
         d_table
```

Out[20]:

cluster	100만원이상	50만원 미만	50만원 이상~100만원 미만
0	8	28	11
1	11	26	16
2	3	6	4
3	2	11	6
4	9	35	14
5	8	9	14

5-2) 카이제곱 검정 결과

```
In [21]: #카이제곱 검정결과 : 귀무가설 채택 (cluster에 따른 지불비용의 차이가 없다.)
         from scipy.stats import chi2_contingency
         chi,p,df,expected=chi2_contingency(d_table)
         # 카이제곱검정통계량과 유의확률
         print("chisquare=%.3f" % chi, "P-value=%.3f" % p)
```

```
chisquare=10.695 P-value=0.382
```

5-3) 군집에 따른 지불비용1의 행별 비율표

```
In [22]:
# 행별 비율표(propotion table)
pd.crosstab(rest["cluster"], rest["지불비용1"], normalize='index').round(4)
```

Out [22]:

지불비용1	100만원이상	50만원 미만	50만원 이상~100만원 미만
cluster			
0	0.1702	0.5957	0.2340
1	0.2075	0.4906	0.3019
2	0.2308	0.4615	0.3077
3	0.1053	0.5789	0.3158
4	0.1552	0.6034	0.2414
5	0.2581	0.2903	0.4516

5-4) 결과해석

지불비용 변수를 범주형으로 변환한 지불비용1 변수를 생성하였다.

5-1)의 분할표를 확인해보면, 지불비용1 변수가 “50만원 미만”, “50만원 이상~100만원 미만”, “100만원 이상”으로 3개의 범주의 가지고 있음을 확인할 수 있다.

다음으로 진행한 5-2)의 카이제곱 검정 결과는 크래머 법칙이 위배되지 않는다는 가정을 기반으로 얻어졌다.

카이제곱 검정 결과, 카이제곱 통계량이 10.695의 값을 갖고, p-value가 0.382로 0.05보다 커 군집에 따른 지불 비용1의 분포에 차이가 없는 것으로 나타났다.

마지막으로 군집에 따른 지불비용의 범주(지불비용1)의 행별 비율표를 살펴보면, 군집5를 제외한 군집들에서 50만원 미만, 50만원 이상~100만원 미만, 100만원 이상 순서로 비율이 높게 나타났다.

반면에 군집5는 50만원 이상~100만원 미만의 비율이 가장 높았고, 100만원 이상과 50만원 미만의 두 개의 범주는 비율이 비슷하게 나타났다.

위의 각 군집들의 지불비용의 평균과 함께 고려하였을 때, 군집 5의 지불비용 평균이 74로 두 번째로 크고 50만원 이상~100만원 미만의 지불비용의 비율이 높은 점으로 군집 5의 레스토랑 서비스 비용이나 메뉴 가격이 다른 군집에 비해 높은 가격을 형성하고 있을 것으로 예측할 수 있다.

6) 군집 별 **소득**의 차이-카이제곱 검정

6-1) 범주 변환 후, 분할표

```
In [24]:
#실습 3-3 군집별 소득의 차이
#소득 1, 2 통합 -> 300만 미만
#소득 3, 4 통합 -> 300 ~ 500만
#소득 5, 6 통합 -> 500만 초과
rest.loc[(rest["소득"]==1)|(rest["소득"]==2), '소득1']='300만 미만'
rest.loc[(rest["소득"]==3)|(rest["소득"]==4), '소득1']='300~ 500만'
rest.loc[(rest["소득"]==5)|(rest["소득"]==6), '소득1']='500만 초과'
```

```
In [25]:
d_table=pd.crosstab(rest["cluster"], rest["소득1"])
d_table
```

Out [25]:

소득1	300~ 500만	300만 미만	500만 초과
cluster			
0	12	34	1
1	20	4	29
2	1	7	5
3	7	9	3
4	13	24	21
5	10	16	5

6-2) 카이제곱 검정 결과

```
In [26]:
#카이제곱 검정결과 : 귀무가설 기각 (cluster에 따른 소득의 차이가 있다.)
from scipy.stats import chi2_contingency
chi,p,df,expected=chi2_contingency(d_table)
# 카이제곱검정통계량과 유의확률
print("chisquare=%.3f" % chi, "P-value=%.3f" % p)
```

```
chisquare=59.445 P-value=0.000
```

6-3) 군집에 따른 소득(1)의 행별 비율표

```
In [27]:
# 행별 비율표(propotion table)
pd.crosstab(rest["cluster"], rest["소득1"], normalize='index').round(4)
```

Out [27]:

소득1	300~500만	300만 미만	500만 초과
cluster			
0	0.2553	0.7234	0.0213
1	0.3774	0.0755	0.5472
2	0.0769	0.5385	0.3846
3	0.3684	0.4737	0.1579
4	0.2241	0.4138	0.3621
5	0.3226	0.5161	0.1613

6-4) 결과해석

소득1 변수는 기존 소득 변수(1-200만미만, 2-200~300만, 3-300~400만, 4-400~500만, 5-500~600만, 6-60만 초과)을 새로운 범주로 분류한 것이다.

6-1)의 분할표를 확인해보면, 소득1 변수가 “300만 미만”, “300~500만 미만, “500만 초과”로 3개의 범주의 가지고 있음을 확인할 수 있다. 기존의 범주형 변수인 소득을 새로운 범주로 분류하여 소득1 변수를 생성한 이유는 카이제곱 검정을 수행하기 위해 크래머 법칙을 고려하였기 때문이다.

다음으로 진행된 6-2)의 카이제곱 검정 결과, 카이제곱 통계량이 59.445의 값을 갖고, p-value가 0.00으로 0.05보다 매우 작아 군집에 따른 소득의 분포에 차이가 있는 것으로 보인다.

마지막으로 군집에 따른 소득의 범주(소득1)의 행별 비율표를 살펴보면, 군집2를 제외한 군집들에서 300만원 미만의 소득이 가장 높은 비율을 나타내고 있다.

자세히 확인하면 군집 0, 3, 5는 300만 미만, 300~500만 미만, 500만 초과 순으로 비율이 높은 것으로 나타났고, 군집 2, 4는 300만 미만, 500초과, 300~500만 미만 순으로 비율이 높은 것으로 나타났다.

군집 1은 500만 초과, 300~500만의 범주가 군집1 전체의 90%정도를 차지하는 것으로 보아, 다른 군집보다도 고소득자가 많다는 것을 알 수 있다.

7) 군집 별 **만족도**의 차이-카이제곱 검정

7-1) 분할표

```
In [28]:
#실습 3-4 군집별 만족도의 차이
d_table=pd.crosstab(rest["cluster"], rest["만족도"])
d_table
```

Out [28]:

만족도	2	3	4	5
cluster				
0	0	11	33	3
1	4	31	18	0
2	0	2	10	1
3	0	7	11	1
4	0	24	32	2
5	0	9	21	1

7-2) 카이제곱 검정 결과

```
In [29]: #카이제곱 검정결과 : 귀무가설 채택 (cluster에 따른 만족도에 차이가 있다.)
         from scipy.stats import chi2_contingency
         chi,p,df,expected=chi2_contingency(d_table)
         # 카이제곱검정통계량과 유의확률
         print("chisquare=%.3f" % chi, "P-value=%.3f" % p)

         chisquare=35.417 P-value=0.002
```

7-3) 군집에 따른 만족도의 행별 비율표

```
In [30]: # 비행별 비율표(propotion table)
         pd.crosstab(rest["cluster"], rest["만족도"], normalize='index').round(4)
```

Out[30]:

만족도	2	3	4	5
cluster				
0	0.0000	0.2340	0.7021	0.0638
1	0.0755	0.5849	0.3396	0.0000
2	0.0000	0.1538	0.7692	0.0769
3	0.0000	0.3684	0.5789	0.0526
4	0.0000	0.4138	0.5517	0.0345
5	0.0000	0.2903	0.6774	0.0323

7-4) 결과해석

만족도 변수는 1-매우 불만족~5-매우 만족을 나타낸다, 그러나 7-1)을 확인하였을 때, 실제로 1번은 나타나지 않았음을 알 수 있다.

다음으로 진행한 7-2)의 카이제곱 검정 결과는 크래머 법칙이 위배되지 않는다는 가정을 기반으로 얻어졌다.

카이제곱 검정 결과 카이제곱 통계량이 35.417의 값을 갖고, p-value가 0.002로 0.05보다 작아 군집에 따른 만족도의 분포 차이가 있는 것으로 보인다.

마지막으로 군집에 따른 만족도의 행별 비율표를 살펴보면, 군집1을 제외한 모든 군집이 3, 4, 5번의 범주에서만 나타났음을 확인할 수 있다. 또한 이들 군집은 모두 4, 3, 5 순으로 높은 비율을 나타내고 있다.

반면 군집1는 2, 3, 4번의 범주만 나타났으며, 다른 군집과 다르게 3. 4, 2 순으로 높은 비율이 나타났다.

이를 통해 군집0, 2, 3, 4, 5의 고객들은 레스토랑에 대한 전반적인 만족도가 양호한 편으로 보이고, 군집 1 또한 양호한 것으로 보이나, 불만을 나타낸 일부 고객이 존재하고 보통의 비율이 가장 많이 나타난 것을 통해 나머지 군집에 비해 만족도가 뛰어나다고 할 순 없을 것으로 보인다.

4. 군집분석 결과 생성된 cluster변수를 target으로 하고, 실내시설~숙박(8개 변수)을 독립변수로 하여 판별분석과 SVM의분류정확도를 비교해 보아라. (train, test자료로 분리하지 않고 전체 자료를 이용하여 분석)

답안)

1) 선형 판별분석

1-1) 선형 판별분석 결과

```
In [45]: ## 전체 자료에 대한 판별분석
         x = rest.iloc[:, 0:8]
         y = rest["cluster"]
         lda = LinearDiscriminantAnalysis()
         lda=lda.fit(x, y)
         print( "R² =", lda.score(x, y))

         R² = 0.9230769230769231

In [34]: # 모형예측(분류)
         y_fit=lda.predict(x)
         y_pred= lda.predict(x)
         confusion_matrix(y, y_pred)

Out[34]: array([[46,  0,  0,  1,  0,  0],
                [ 0, 49,  1,  0,  2,  1],
                [ 0,  0, 12,  0,  1,  0],
                [ 0,  2,  0, 16,  1,  0],
                [ 2,  0,  0,  0, 56,  0],
                [ 2,  3,  0,  1,  0, 25]], dtype=int64)
```

1-2) 결과분석

x 변수는 독립변수로 실내시설 ~ 숙박의 8개 변수이고, y 변수는 target이 되는 변수로 군집분석 결과 생성된 cluster 변수이다. x, y 변수를 선형 판별모형에 적합하여, y의 자료를 분류한 분류정확도가 0.92정도로 나타났고, 오차행렬(행- 실제, 열- 예측)을 통해 모든 군집에서 모형을 통한 분류가 정확하지 않다는 것을 확인할 수 있다.

2) 선형 판별분석의 분류 정확도

2-1) 모형평가 종합정리

```
In [35]: # 모형평가(분류정확도_보고서) 종합정리
         from sklearn.metrics import classification_report
         print(classification_report(y, y_pred))

                       precision    recall  f1-score   support

                    0       0.92      0.98      0.95        47
                    1       0.91      0.92      0.92        53
                    2       0.92      0.92      0.92        13
                    3       0.89      0.84      0.86        19
                    4       0.93      0.97      0.95        58
                    5       0.96      0.81      0.88        31

             accuracy                           0.92       221
            macro avg       0.92      0.91      0.91       221
         weighted avg       0.92      0.92      0.92       221
```

2-2) 결과해석

1에서 확인한 것과 동일하게 분류정확도는 0.92 정도로 나타났고, 군집5의 정밀도(precision)가 0.96으로 가장 크고, 군집3의 정밀도가 0.89로 가장 낮음을 알 수 있다. 이것은 예측한 것 중 정답의 비율이 군집5에서 가장 높으며, 군집3에서 가장 낮다는 것을 의미한다.

또한 군집4의 재현율(recall)이 0.97로 가장 크고, 군집5의 재현율이 0.81로 가장 낮게 나타난 점을 통해 찾아야 할 대상을 실제로 잘 찾은 군집이 군집4이고, 잘 찾지 못한 군집이 군집5임을 알 수 있다.

따라서 정밀도와 재현율은 높을수록 좋으므로, 정밀도와 재현율의 평균인 f1-score가 높을수록 성능이 좋다고 할 수 있을 것이다. 위의 표에서 군집4에서 f1-score가 0.95로 가장 높으므로, 따라서 선형 판별모형이 군집4를 분류하는 성능이 가장 좋다고 볼 수 있다.

3) SVM 결과 - 선형 커널

3-1) 선형 커널을 이용한 SVM 결과

```
In [51]:
# SVM - 분류정확도 분석(선형)
svclassifier = SVC(kernel='linear')
svclassifier.fit(x, y)
y_pred = svclassifier.predict(x)
print('score : ',svclassifier.score(x,y))
print(confusion_matrix(y, y_pred))
print(classification_report(y, y_pred))
```

```
score :  0.9728506787330317
[[45  0  0  1  1  0]
 [ 0 52  0  0  0  1]
 [ 0  0 13  0  0  0]
 [ 0  0  0 18  1  0]
 [ 2  0  0  0 56  0]
 [ 0  0  0  0  0 31]]
              precision    recall  f1-score   support

           0       0.96      0.96      0.96        47
           1       1.00      0.98      0.99        53
           2       1.00      1.00      1.00        13
           3       0.95      0.95      0.95        19
           4       0.97      0.97      0.97        58
           5       0.97      1.00      0.98        31

    accuracy                           0.97       221
   macro avg       0.97      0.98      0.97       221
weighted avg       0.97      0.97      0.97       221
```

3-2) 결과분석

분류 정확도가 0.972정도로 나타났고, 정밀도와 재현율의 평균인 f1-score가 군집2에서 1의 값을 가지므로 선형 판별모형이 군집2를 완벽히 분류해내고 있다고 판단된다. 나머지 군집에서도 분류 성능이 매우 좋은 것으로 보인다.

4) SVM 결과 - 다항 커널(poly)

4-1) 다항 커널을 이용한 SVM 결과

```
In [38]:
# SVM - 커널함수= 다항식(poly)
svclassifier = SVC(kernel='poly', degree=8)
svclassifier.fit(x, y)
y_pred = svclassifier.predict(x)
print('score : ',svclassifier.score(x,y))
print(confusion_matrix(y, y_pred))
print(classification_report(y, y_pred))
```

```
score :  1.0
[[47  0  0  0  0  0]
 [ 0 53  0  0  0  0]
 [ 0  0 13  0  0  0]
 [ 0  0  0 19  0  0]
 [ 0  0  0  0 58  0]
 [ 0  0  0  0  0 31]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        47
           1       1.00      1.00      1.00        53
           2       1.00      1.00      1.00        13
           3       1.00      1.00      1.00        19
           4       1.00      1.00      1.00        58
           5       1.00      1.00      1.00        31

    accuracy                           1.00       221
   macro avg       1.00      1.00      1.00       221
weighted avg       1.00      1.00      1.00       221
```

4-2) 결과분석

분류정확도가 1의 값을 가져 모든 군집을 잘 분류하고 있는 것으로 판단된다.

5) SVM 결과 - 가우시안 커널(정규분포함수)

5-1) 가우시안 커널을 이용한 SVM 결과

```
In [50]:
# SVM - 커널함수= 가우시안(정규분포함수)
svclassifier = SVC(kernel='rbf')
svclassifier.fit(x, y)
y_pred = svclassifier.predict(x)
print('score : ',svclassifier.score(x,y))
print(confusion_matrix(y, y_pred))
print(classification_report(y, y_pred))
```

```
score :  0.9592760180995475
[[46  0  0  1  0  0]
 [ 0 52  0  0  0  1]
 [ 0  0 12  0  1  0]
 [ 0  0  0 19  0  0]
 [ 2  0  0  0 56  0]
 [ 1  0  0  1  2 27]]
              precision    recall  f1-score   support

           0       0.94      0.98      0.96        47
           1       1.00      0.98      0.99        53
           2       1.00      0.92      0.96        13
           3       0.90      1.00      0.95        19
           4       0.95      0.97      0.96        58
           5       0.96      0.87      0.92        31

    accuracy                           0.96       221
   macro avg       0.96      0.95      0.96       221
weighted avg       0.96      0.96      0.96       221
```

5-2) 결과분석

분류 정확도가 0.96정도로 나타났고, 정밀도와 재현율의 평균인 f1-score가 군집1에서 0.99의 값을 가지므로 선형 판별모형이 군집1를 분류하는 성능이 가장 좋다고 볼 수 있다.

6) SVM 결과 - 시그모이드 커널(sigmoid)

6-1) 시그모이드 커널을 이용한 SVM 결과

```
In [52]:
# SVM - 커널함수= 시그모이드(로지스틱함수)
svclassifier = SVC(kernel='sigmoid')
svclassifier.fit(x, y)
y_pred = svclassifier.predict(x)
print('score : ',svclassifier.score(x,y))
print(confusion_matrix(y, y_pred))
print(classification_report(y, y_pred))
```

```
score :  0.26244343891402716
[[ 0  0  0  0 47  0]
 [ 0  0  0  0 53  0]
 [ 0  0  0  0 13  0]
 [ 0  0  0  0 19  0]
 [ 0  0  0  0 58  0]
 [ 0  0  0  0 31  0]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        47
           1       0.00      0.00      0.00        53
           2       0.00      0.00      0.00        13
           3       0.00      0.00      0.00        19
           4       0.26      1.00      0.42        58
           5       0.00      0.00      0.00        31

    accuracy                           0.26       221
   macro avg       0.04      0.17      0.07       221
weighted avg       0.07      0.26      0.11       221
```

6-2) 결과분석

분류정확도가 0.26으로 낮게 나타났다. 또한 군집0, 1, 2, 3, 5에 대하여 정밀도와 재현율이 모두 0으로 따라서 f1-score도 0인 것을 확인할 수 있다. 이를 통해 군집0, 1, 2, 3, 5에 대한 분류를 전혀 수행하지 못하고 있음을 알 수 있다. 반면 군집4의 재현율은 1의 값을 가지므로, 실제 군집4로 분류되는 것들을 모형을 통해 잘 분류하였다고 판단할 수 있다.

7) 판별분석과 SVM의 분류정확도 비교

7-1) 분류정확도 결과

```
In [45]:
## 전체 자료에 대한 판별분석과 분류정확도
lda = LinearDiscriminantAnalysis()
lda=lda.fit(x, y)
print( "R² =", lda.score(x, y))
```

```
R² = 0.9230769230769231
```

```
In [51]:
# SVM - 분류정확도 분석(선형)
svclassifier = SVC(kernel='linear')
svclassifier.fit(x, y)
print('score : ',svclassifier.score(x,y))
```

```
score :  0.9728506787330317
```

```
In [38]:
# SVM - 커널함수= 다항식(poly)
svclassifier = SVC(kernel='poly', degree=8)
svclassifier.fit(x, y)
print('score : ',svclassifier.score(x,y))
```

```
score :  1.0
```

```
In [50]:
# SVM - 커널함수= 가우시안(정규분포함수)
svclassifier = SVC(kernel='rbf')
svclassifier.fit(x, y)
print('score : ',svclassifier.score(x,y))
```

```
score :  0.9592760180995475
```

```
In [52]:
# SVM - 커널함수= 시그모이드(로지스틱함수)
svclassifier = SVC(kernel='sigmoid')
svclassifier.fit(x, y)
print('score : ',svclassifier.score(x,y))
```

```
score :  0.26244343891402716
```

7-2) 결과해석

다항식(poly) 커널 함수를 이용한 SVM의 분류정확도가 1으로 군집을 가장 잘 분류하는 것으로 보인다.
다음으로 선형 커널 함수를 이용한 SVM, 가우시안(정규분포함수) 커널 함수를 이용한 SVM, 판별분석, 시그모이드 커널 함수를 이용한 SVM 순으로 분류 성능이 좋은 것으로 나타났다.