프로젝트 결과 보고서

주제 : 국내 과일 시장과 소비 분석 및 수요 예측

제출일	2023. 12. 03	전 공	컴퓨터정보과
과 목	빅데이터 처리	학 번	202044044
담당교수	민정혜 교수님	이 름	정다운

[목 차]

1. 과일 1일 섭취량 분석

1-1. 연령층 별 과일섭취량 추이와 섭취량 순위

※ 데이터 수집 : 사용한 데이터 정보

자료명 : 2021 국민건강통계
파일명 : 식품섭취.xlsx
시트명 : 11.과일류
과일 섭취량 : 과일류 식품의 섭취 중량에 대한 합/분석대상자 수
소득수준 : 월가구균등화소득(월가구소득/√가구원수)을 성별·연령별(5세단위) 오분위로 분류

```
# 파일 경로와 시트명 설정
file_path = '식품섭취.xlsx'# 파일경로
sheet_name = '11.과일류'# 시트명
# Excel 파일 읽기 : 연령별 데이터
age_df = pd .read_excel (file_path , sheet_name =sheet_name , header =None ,
                         index_col = 2 , skiprows =15 , nrows =8 )
# 데이터프레임 확인
age_df
```

2	0	1	3	4	5	6	7	8	9	10	...	47	48	49	50	51	52	53	54	55	56
1-9	NaN	NaN	1459	176.9	(9.0)	1506	187.1	(8.0)	1199	91.8	...	(8.1)	736	139.2	(9.6)	535	122.3	(9.3)	483	129.4	(8.8)
10-18	NaN	NaN	1440	196.9	(10.3)	1370	187.5	(9.9)	1205	83.9	...	(8.8)	639	96.1	(7.5)	469	94.8	(16.1)	484	102.1	(10.5)
19-29	NaN	NaN	1550	220.0	(11.3)	1370	203.7	(11.6)	1045	90.3	...	(7.0)	669	86.8	(8.2)	624	66.7	(7.1)	584	64.3	(5.8)
30-39	NaN	NaN	1866	228.8	(9.8)	1805	227.0	(10.4)	1466	110.8	...	(11.5)	855	110.1	(8.3)	633	104.4	(8.5)	549	83.1	(8.0)
40-49	NaN	NaN	1461	202.9	(8.5)	1601	222.4	(12.4)	1559	88.7	...	(7.4)	1036	165.2	(18.1)	807	143.3	(9.4)	810	128.1	(9.1)
50-59	NaN	NaN	1104	189.7	(11.2)	979	235.4	(12.6)	1015	83.7	...	(9.9)	1061	201.6	(10.3)	861	169.6	(11.4)	909	180.2	(10.1)
60-69	NaN	NaN	942	158.6	(11.6)	788	210.2	(14.1)	847	63.3	...	(10.6)	1043	188.6	(9.2)	892	177.5	(10.3)	981	177.3	(10.4)
70+	NaN	NaN	578	115.1	(11.7)	549	150.1	(11.5)	594	41.4	...	(10.6)	1108	147.6	(7.9)	987	156.7	(8.4)	1140	142.1	(8.9)

8 rows × 56 columns

※ 데이터 가공 및 정제

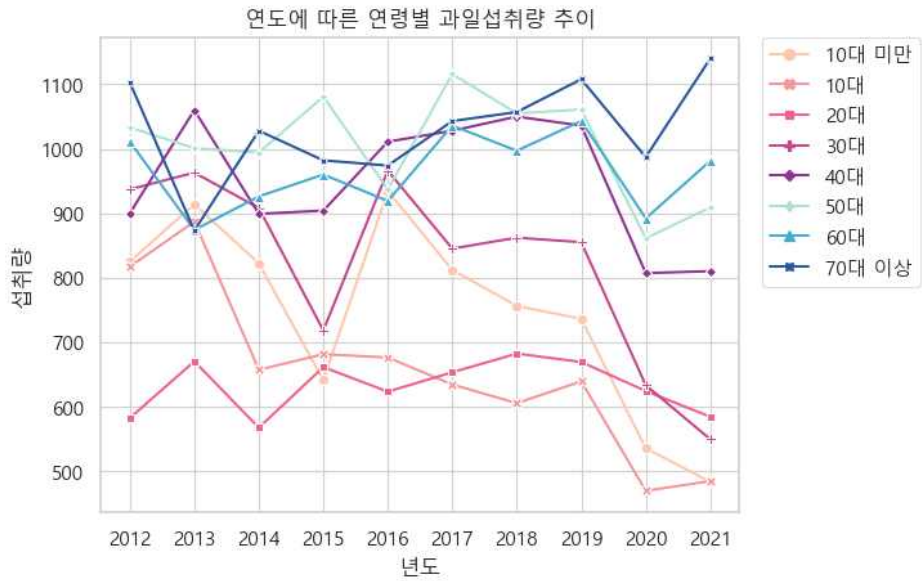
```
# 2012~2021(10년간) 연령별 섭취 데이터만 수집
df =age_df .drop ([0 ,1 ], axis =1 )
df =df .iloc [:,24 ::3 ]

df .columns =['2012','2013','2014','2015','2016','2017'
 ,'2018','2019','2020','2021'] # 컬럼명 변경
df .index =['10대 미만','10대','20대','30대',
'40대','50대','60대','70대 이상'] # 인덱스명 변경
df
```

	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
10대 미만	825	914	822	641	934	812	756	736	535	483
10대	817	887	657	681	676	634	605	639	469	484
20대	582	670	567	661	623	653	682	669	624	584
30대	937	963	908	718	966	845	862	855	633	549
40대	900	1060	899	904	1011	1028	1050	1036	807	810
50대	1033	1001	994	1081	937	1116	1055	1061	861	909
60대	1011	874	926	960	919	1036	997	1043	892	981
70대 이상	1103	873	1028	982	974	1043	1057	1108	987	1140

※ 데이터 분석 및 시각화

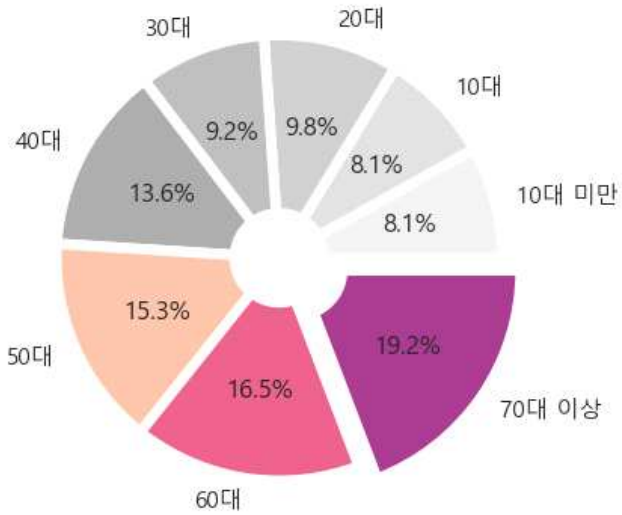
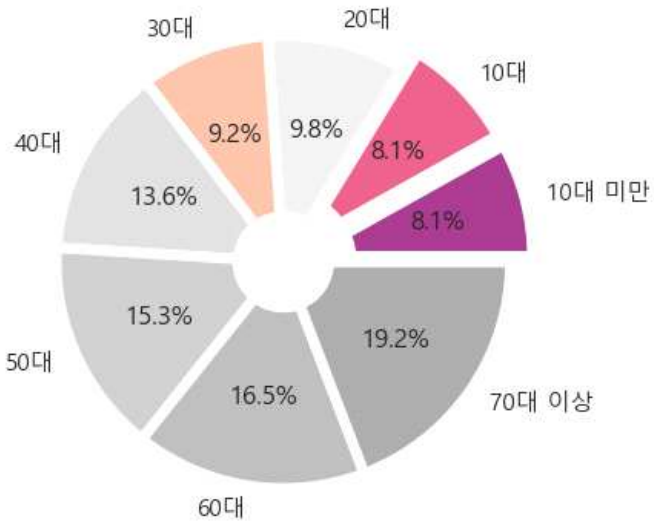
[10년 간 연령별 과일 섭취량 추이 살피기]

```
df_copy = df .T # x축 y축 치환
colors = ['#f8c1a8', '#ef9198', '#e8608a', '#c0458a', '#8f3192', '#aadacc', '#44a7cb', '#2a5599' ] # 색상 리스트
# 그래프 그리기
sns .set(style ='whitegrid', font ='Malgun Gothic', font_scale =1 , palette =colors )
ax = sns .lineplot(data =df_copy , markers =True , dashes =False )
ax .legend(loc ='upper left', bbox_to_anchor =(1.02 , 1.02 ))  # 범례 위치 조정
ax .set(xlabel ='년도', ylabel ='섭취량',title ='연도에 따른 연령별 과일섭취량 추이')
```



결 과

전반적으로 연령대가 높아질수록 과일 섭취량이 높은 경향을 보인다. 70대 이상 연령 그룹이 가장 높은 과일 섭취량을 보이고 있다. 30대 이하의 연령대에서는 대체로 섭취량이 감소하는 추세이다.

[과일 섭취량이 많은 연령대 순위]	[과일 섭취량이 적은 연령대 순위]
explode = [0 , 0 , 0 , 0 , 0 , 0 , 0 , 0.10] colors = ['#eee', '#ddd', '#ccc', '#bbb', '#aaa', '#f8c1a8', '#e8608a', '#a73b8f'] wedgeprops ={'width': 0.8 , 'edgecolor': 'w', 'linewidth': 5 } plt .pie (df_copy .loc ['2021'], labels =df_copy .columns , explode =explode ,autopct ='%.1f%%', colors =colors , wedgeprops =wedgeprops) plt .show ()	explode = [0.10 , 0.10 , 0 , 0 , 0 , 0 , 0 , 0 ,] colors = ['#a73b8f', '#e8608a', '#eee','#f8c1a8', '#ddd', '#ccc', '#bbb', '#aaa'] wedgeprops ={'width': 0.8 , 'edgecolor': 'w', 'linewidth': 5 } plt .pie (df_copy .loc ['2021'], labels =df_copy .columns , explode =explode ,autopct ='%.1f%%', colors =colors , wedgeprops =wedgeprops) plt .show ()
	
결 과 70대 이상이 19.2%로 1위, 60대가 16.5%로 2위, 50대가 15.3%로 3위 → 대체로 연령대가 높을수록 과일 섭취량이 많은 것으로 나타났다.	**결 과** 10대 미만이 8.1%로 1위, 10대가 8.1%로 2위, 30대가 9.2%로 3위 → 낮은 연령대의 과일 섭취량이 많은 것으로 나타났다.

1. 과일 1일 섭취량 분석

1-2. 소득 수준별 과일섭취량 추이와 섭취량 순위

※ 데이터 수집 : 사용한 데이터 정보

자료명 : 2021 국민건강통계
파일명 : 식품섭취.xlsx
시트명 : 11.과일류
과일 섭취량 : 과일류 식품의 섭취 중량에 대한 합/분석대상자 수
소득수준 : 월가구균등화소득(월가구소득/√가구원수)을 성별·연령별(5세단위) 오분위로 분류

```
# 파일 경로와 시트명 설정
file_path = '식품섭취.xlsx'# 파일경로
sheet_name = '11.과일류'# 시트명

# Excel 파일 읽기 : 소득수준별 데이터
income_df = pd .read_excel (file_path , sheet_name =sheet_name , header =None ,
                            index_col = 2 , skiprows =49 , nrows =5 )
# 데이터프레임 확인
income_df
```

2	0	1	3	4	5	6	7	8	9	10	...	47	48	49	50	51	52	53	54	55	56
하	NaN	NaN	2037	145.9	(8.6)	1689	174.6	(9.7)	1870	59.4	...	(6.3)	1422	112.6	(6.3)	1165	98.6	(9.1)	1182	104.5	(8.4)
중하	NaN	NaN	2009	179.1	(7.7)	1896	196.5	(10.4)	1790	77.8	...	(5.8)	1403	127.0	(6.9)	1161	116.2	(7.9)	1200	97.0	(5.9)
중	NaN	NaN	2172	197.5	(8.3)	1898	220.3	(11.5)	1732	91.2	...	(7.7)	1454	123.2	(6.7)	1166	127.6	(9.5)	1206	120.3	(6.8)
중상	NaN	NaN	2114	212.6	(10.1)	1778	213.6	(11.3)	1736	98.2	...	(8.1)	1436	152.7	(14.8)	1116	124.9	(8.7)	1161	119.8	(8.8)
상	NaN	NaN	2068	236.5	(10.4)	2130	234.1	(10.7)	1715	116.7	...	(7.4)	1405	160.1	(8.8)	1182	135.8	(10.0)	1163	138.7	(7.8)

5 rows × 56 columns

※ 데이터 가공 및 정제

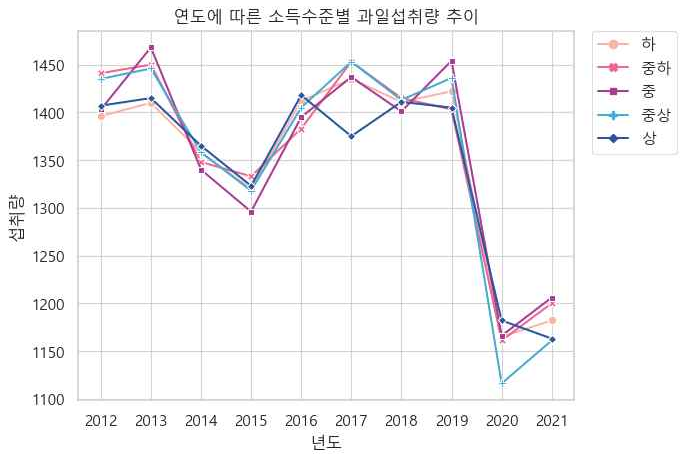
```
# 2012~2021(10년간) 소득수준별 섭취 데이터만 수집
df2 =income_df .drop ([0 ,1 ], axis =1 )
df2 = df2 .iloc [:,24 ::3 ]

df2 .index = ['하','중하','중','중상','상'] # 인덱스명 변경(공백제거)
df2 .columns
=['2012','2013','2014','2015','2016','2017','2018','2019','2020','2021'] # 컬럼명 변경
df2
```

	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
하	1396	1410	1358	1320	1412	1435	1411	1422	1165	1182
중하	1441	1450	1348	1333	1383	1453	1415	1403	1161	1200
중	1403	1468	1340	1296	1395	1437	1401	1454	1166	1206
중상	1435	1446	1358	1318	1405	1453	1413	1436	1116	1161
상	1407	1415	1365	1323	1418	1375	1411	1405	1182	1163

※ 데이터 분석 및 시각화

[10년 간 소득수준별 과일 섭취량 추이 살피기]

```
df2_copy = df2 .T # x축 y축 치환
colors = [ '#f4aea3', '#e8638b',
'#a73b8f','#44a7cb','#2a5599'] # 색상 리스트
# 그래프 그리기
sns .set(style ='whitegrid', font ='Malgun Gothic',
font_scale =1 )
ax = sns .lineplot(data =df2_copy , markers =True ,
dashes =False , palette =colors ) # 컬러 지정
ax .legend(loc ='upper left', bbox_to_anchor =(1.02 ,
1.02 )) # 범례 위치 조정
ax .set(xlabel ='년도', ylabel ='섭취량', title ='연도에
따른 소득수준별 과일섭취량 추이')
plt .show ()
```

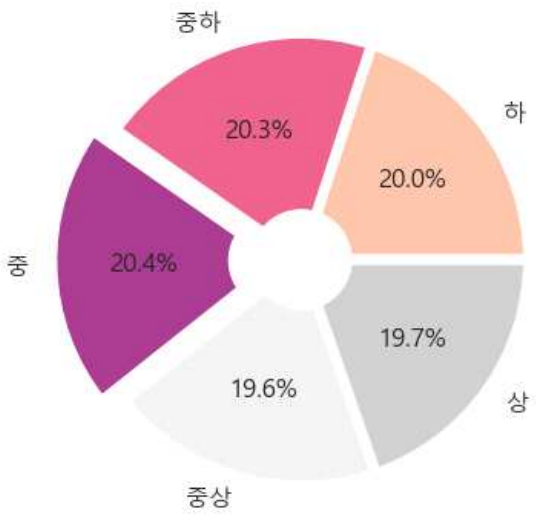


결 과

모든 소득 계층의 섭취량 증감이 비슷한 추이를 보이는 경향이 있다. 2013년부터 2015년까지 전반적 하락세를 보이다 2016년부터 증가 추세를 보였지만, 2020년도에 전 소득계층에서 과일 섭취량이 대폭 하락세를 보였다.

[과일 섭취량이 많은 소득계층 순위]

```
explode = [0 , 0 , 0.10 , 0 , 0 ]
colors = ['#f8c1a8', '#e8608a', '#a73b8f', '#eee',
'#ccc']
wedgeprops ={'width': 0.8 , 'edgecolor': 'w',
'linewidth': 5 }
plt .pie (df2_copy .loc ['2021'], labels =df2_copy
.columns , explode =explode ,autopct ='%.1f%%', colors
=colors , wedgeprops =wedgeprops )
plt .show ()
```



결 과

중위계층이 20.4%로 1위, 중하위계층이 20.3%로 2위, 하위계층이 20.0%로 3위를 나타냈지만, 계층별 큰 격차를 보이지 않았다.

1. 과일 1일 섭취량 분석

1-3. 성별 과일섭취량 비교

※ 데이터 수집 : 사용한 데이터 정보

자료명 : 2021 국민건강통계
파일명 : 식품섭취.xlsx
시트명 : 11.과일류
과일 섭취량 : 과일류 식품의 섭취 중량에 대한 합/분석대상자 수
소득수준 : 월가구균등화소득(월가구소득/√가구원수)을 성별·연령별(5세단위) 오분위로 분류

[남성 데이터]

```
# 파일 경로와 시트명 설정
file_path = '식품섭취.xlsx'# 파일경로
sheet_name = '11.과일류'# 시트명


# Excel 파일 읽기 : 남성 데이터
male_df = pd .read_excel (file_path , sheet_name =sheet_name , header =None ,
                          index_col = 2 , skiprows =62 , nrows =8 )
# 데이터프레임 확인
male_df .head ()
```

2	0	1	3	4	5	6	7	8	9	10	...	47	48	49	50	51	52	53	54	55	56
1-9	NaN	NaN	766	166.2	(10.6)	811	199.3	(9.5)	615	90.3	...	(10.2)	373	145.0	(11.6)	273	133.6	(11.2)	240	132.5	(13.0)
10-18	NaN	NaN	738	199.0	(13.7)	696	180.1	(11.4)	634	76.8	...	(11.6)	334	84.4	(9.9)	255	100.1	(26.7)*	259	98.2	(13.4)
19-29	NaN	NaN	707	188.3	(13.5)	615	155.2	(12.2)	451	73.2	...	(11.0)	323	92.1	(13.6)	292	63.1	(9.0)	279	60.7	(8.9)
30-39	NaN	NaN	922	197.0	(12.1)	857	182.4	(11.8)	659	99.0	...	(11.9)	384	91.5	(11.6)	270	93.7	(12.5)	232	71.8	(10.7)
40-49	NaN	NaN	720	161.6	(10.6)	784	167.9	(15.1)	731	73.3	...	(10.6)	441	139.6	(10.0)	348	123.6	(10.7)	351	124.2	(15.0)

[여성 데이터]

```
# Excel 파일 읽기 : 여성 데이터
female_df = pd .read_excel (file_path , sheet_name =sheet_name , header =None ,
                            index_col = 2 , skiprows =109 , nrows =8 )
# 데이터프레임 확인
female_df .head ()
```

2	0	1	3	4	5	6	7	8	9	10	...	47	48	49	50	51	52	53	54	55	56
1-9	NaN	NaN	693	188.9	(12.7)	695	172.9	(10.0)	584	93.5	...	(10.0)	363	133.2	(11.1)	262	110.5	(12.7)	243	126.1	(11.2)
10-18	NaN	NaN	702	194.7	(12.5)	674	195.2	(13.3)	571	92.0	...	(12.5)	305	109.0	(10.3)	214	88.5	(10.2)	225	106.4	(14.6)
19-29	NaN	NaN	843	246.6	(13.8)	755	243.7	(15.4)	594	108.3	...	(7.6)	346	80.8	(7.7)	332	70.6	(9.7)	305	68.2	(7.9)
30-39	NaN	NaN	944	260.2	(12.5)	948	266.7	(13.0)	807	123.3	...	(16.8)	471	130.4	(9.4)	363	116.2	(10.8)	317	95.5	(10.1)
40-49	NaN	NaN	741	245.8	(11.7)	817	273.0	(14.8)	828	104.7	...	(9.1)	595	191.9	(34.6)	459	163.8	(13.4)	459	132.2	(9.8)

※ 데이터 가공 및 정제

[2012~2021(10년간) 섭취 데이터만 수집]

```
# 2012~2021 여성 섭취 데이터만 수집
female_origin =female_df .drop ([0 ,1 ], axis =1 )
female_origin =female_origin .iloc [:,24 ::3 ]
female_origin .index # 인덱스명 변경(공백제거)
female_origin .columns
=['2012','2013','2014','2015','2016','2017','2018','2019
','2020','2021'] # 컬럼명 변경
female_origin .head ()
```

2	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
1-9	353	444	394	322	460	383	382	363	262	243
10-18	391	426	308	306	327	313	277	305	214	225
19-29	366	380	334	346	359	344	359	346	332	305
30-39	581	583	528	438	597	483	480	471	363	317
40-49	550	616	532	545	594	611	625	595	459	459

```
# 2012~2021 남성 섭취 데이터만 수집
male_origin =male_df .drop ([0 ,1 ], axis =1 )
male_origin =male_origin .iloc [:,24 ::3 ]
male_origin .index # 인덱스명 변경(공백제거)
male_origin                                    .columns
=['2012','2013','2014','2015','2016','2017','2018','2019
','2020','2021'] # 컬럼명 변경
male_origin .head ()
```

2	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
1-9	472	470	428	319	474	429	374	373	273	240
10-18	426	461	349	375	349	321	328	334	255	259
19-29	216	290	233	315	264	309	323	323	292	279
30-39	356	380	380	280	369	362	382	384	270	232
40-49	350	444	367	359	417	417	425	441	348	351

[연도별 연령별 섭취량 데이터 확보]

```
male_origin_copy = male_origin .T
female_origin_copy = female_origin .T
print (male_origin_copy .head ())
print (female_origin_copy .head ())
```

```
2      1-9  10-18  19-29  30-39  40-49  50-59  60-69  70+
2012   472    426    216    356    350    411    435  461
2013   470    461    290    380    444    405    392  354
2014   428    349    233    380    367    387    409  423
2015   319    375    315    280    359    453    440  401
2016   474    349    264    369    417    368    404  418
2      1-9  10-18  19-29  30-39  40-49  50-59  60-69  70+
2012   353    391    366    581    550    622    576  642
2013   444    426    380    583    616    596    482  519
2014   394    308    334    528    532    607    517  605
2015   322    306    346    438    545    628    520  581
2016   460    327    359    597    594    569    515  556
```

[행과 열의 합계를 구해서 여성, 남성의 total 데이터 프레임 생성]

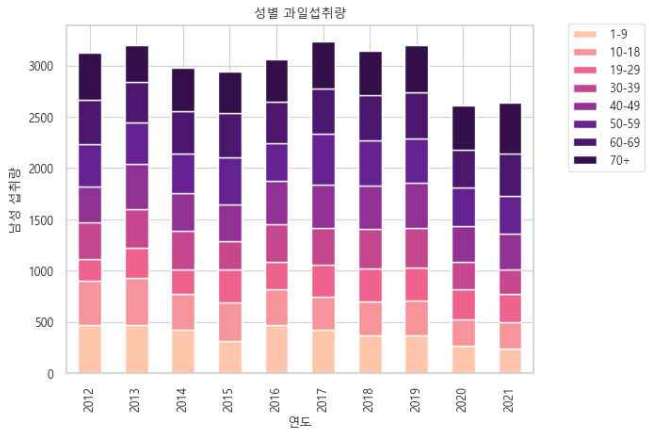
```
# 행과 열의 합계를 구해서 total 데이터 프레임 생성
male_origin_total = male_origin_copy .sum (axis =1 )
female_origin_total = female_origin_copy .sum (axis =1 )
print (male_origin_total .head ())
print (female_origin_total .head ())
```

```
2012    3127
2013    3196
2014    2976
2015    2942
2016    3063
dtype: int64
2012    4081
2013    4046
2014    3825
2015    3686
2016    3977
dtype: int64
```

※ 데이터 분석 및 시각화

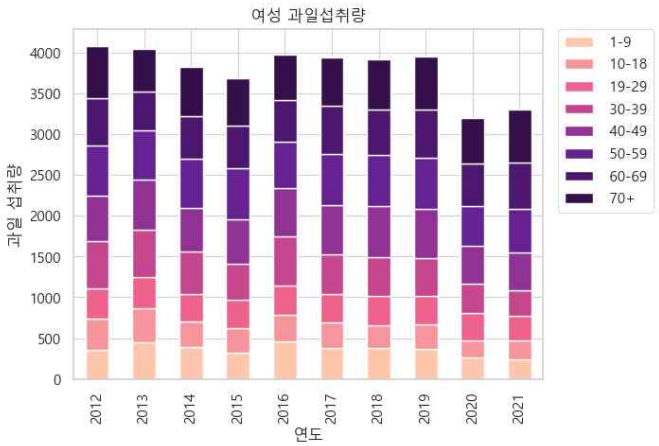
[남성 과일 섭취량 살펴보기]

```
colors = ['#f8c1a8', '#ef9198', '#e8608a', '#c0458a', '#8f3192',
'#63218f', '#4b186c', '#33104a' ]
male_origin_copy .plot (kind ='bar', stacked =True , color
=colors )
plt .title ('성별 과일섭취량')
plt .xlabel ('연도')
plt .ylabel ('남성 섭취량')
# 범례 위치와 레이아웃 조정
plt .legend (loc ='upper right', bbox_to_anchor =(1.25 , 1.02 ))
plt .show ()
```



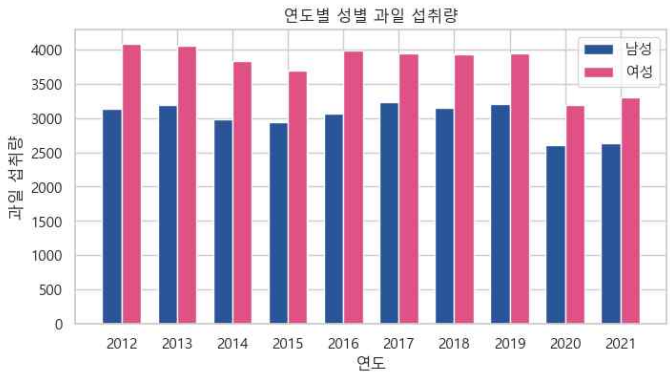
[여성 과일 섭취량 살펴보기]

```
colors = ['#f8c1a8', '#ef9198', '#e8608a', '#c0458a', '#8f3192',
'#63218f', '#4b186c', '#33104a' ]
female_origin_copy .plot (kind ='bar', stacked =True , color
=colors )
plt .title ('여성 과일섭취량')
plt .xlabel ('연도')
plt .ylabel ('과일 섭취량')
# 범례 위치와 레이아웃 조정
plt .legend (loc ='upper right', bbox_to_anchor =(1.25 , 1.02 ))
plt .show ()
```



[연도별 성별 과일 섭취량 살펴보기]

```
# 그래프의 크기 설정
plt .figure (figsize =(8 , 4 ))
# 연도별 남성과 여성의 토탈 섭취량 멀티 바 차트
bar_width = 0.35
index = np .arange(len (male_origin_total ))
plt .bar (index , male_origin_total , bar_width , color
='#2a5599', label ='남성')
plt .bar (index + bar_width , female_origin_total , bar_width ,
color ='#e05286', label ='여성')
# 축과 레이블 설정
plt .xlabel ('연도')
plt .ylabel ('과일 섭취량')
plt .title ('연도별 성별 과일 섭취량')
plt .xticks (index + bar_width / 2 , male_origin_total .index )
plt .legend ()
# 그림 보이기
plt .show ()
```



결 과

모든 연도에서 여성의 과일 섭취량이 남성의 과일 섭취량보다 높은 수치를 였으며, 남녀 모두 2020년에 과일 섭취량이 큰폭으로 감소하는 추세를 보였다.

※ 데이터 가공 및 정제

[2012~2021(10년간) 섭취 데이터만 수집]

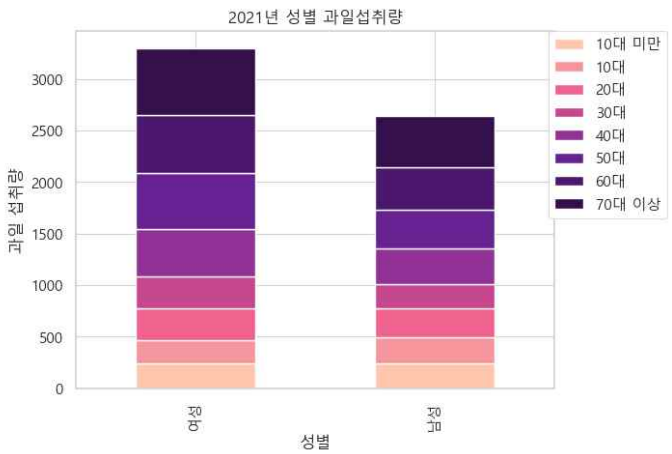
```
# 2021 여성 섭취 데이터만 수집
female_data = female_origin .iloc [:,9 ]

# 2021 남성 섭취 데이터만 수집
male_data = male_origin .iloc [:,9 ]

total = {'여성':female_data , '남성':male_data }
total_data = pd .DataFrame (total )
otal_data .index =['10대 미만','10대','20대','30대','40대','50대','60대','70대 이상'] # 인덱스명 변경
total_data_copy = total_data .T # x축 y축 치환
total_data_copy
```

	10대 미만	10대	20대	30대	40대	50대	60대	70대 이상
여성	243	225	305	317	459	539	567	646
남성	240	259	279	232	351	370	414	494

※ 데이터 분석 및 시각화

[성별 과일 섭취량 비교하기]

```
colors = ['#f8c1a8', '#ef9198', '#e8608a',
'#c0458a', '#8f3192', '#63218f', '#4b186c',
'#33104a' ]
total_data_copy .plot (kind ='bar',
stacked =True , color =colors )
plt .title ('2021년 성별 과일섭취량')
plt .xlabel ('성별')
plt .ylabel ('과일 섭취량')
# 범례 위치와 레이아웃 조정
plt .legend (loc ='upper right', bbox_to_anchor
=(1.25 , 1.02 ))
plt .show ()
```



결 과

여성의 과일 섭취량이 남성의 과일 섭취량 보다 많은것으로 나타났으며, 연령대가 높아질 수록 그 폭이 상승했다.

2. 과일 가격변화와 식품 물가의 상관 관계 분석

2-1. 과일별 가격 변동 추이

※ 데이터 수집 : 사용한 데이터 정보

```
[ 과일 가격 정보 ]
자료명 : 연도별 도.소매가격정보 OPEN-API
```

```
# 부류코드
code_categories = [400 ]  # 200:채소류, 400: 과일류
# 품종별 등급코드
code_product = {
   221 : ['00'],
   226 : ['00'],
   411 : ['01', '05', '06', '07'],
   412 : ['01', '02', '03', '04'],
   413 : ['01', '04', '05'],
   414 : ['01', '02', '03', '06', '07', '08', '09', '10', '11', '12'],
   415 : ['00', '01', '02']
}
# 221: 수박, 226: 딸기, 411: 사과, 412: 배, 413: 복숭아, 414: 포도, 415: 감귤
# API 요청 보내기
base_url = 'http://www.kamis.or.kr/service/price/xml.do?action=yearlySalesList'
p_cert_key = 'a667632f-857f-40c5-805c-6ee126d7a1f4'
p_cert_id = 3772
p_yyyy = 2021
p_countycode = 1101
p_convert_kg_yn = 'N'
dfset = pd .DataFrame ()  # 빈 데이터프레임
for code_category in code_categories :
   for item_code , kind_codes in code_product .items ():
     for kind_code in kind_codes : # 품종별 등급코드 순회
       for rank_code in range (1 , 6 ): # 품종 순회
         try :
           # 요청
           response = requests .get (
             f '{base_url}&p_yyyy={p_yyyy}&p_itemcategorycode={code_category}&p_itemcode={item_code
}&p_kindcode={kind_code}&p_graderank={rank_code }&p_countycode={p_countycode }&p_convert_kg_yn={p_convert_kg_yn
}&p_cert_key={p_cert_key }&p_cert_id={p_cert_id }&p_returntype=xml'
           )
           # 응답 처리
           if response .status_code == 200 :
             # XML 파싱
             root = ET .fromstring (response .content )
             # 데이터 추출
             data_list = []
             for price_elem in root .findall ('price'):
               try :
                 productclscode = price_elem .find ('productclscode').text
```

```
            except AttributeError :
              continue  # productclscode가 없는 경우 스킵
            caption = price_elem .find ('caption').text
            for item_elem in price_elem .findall ('item'):
              temp_dict = {'caption': caption }
              for child in item_elem :
                temp_dict [child .tag ] = child .text
              data_list .append (temp_dict )
          # 판다스 데이터프레임으로 변환
          df = pd .DataFrame (data_list )
          if dfset .empty :
            # 첫 데이터
            dfset = df
          else : # 첫 데이터 이후부터는 이어붙임
            if df .empty :
              continue  # 빈 데이터일때 스킵
            #print(df)
            dfset = pd .concat ([dfset , df ], axis =0 , join ='outer', ignore_index =True )
        else :
          print ('Error occurred:', response .status_code )
      except requests .exceptions .RequestException as e :
        print ('Request failed:', e )
      except Exception as e :
        print ('An error occurred:', e )
print (dfset.shape)
dfset .head () # 처음 5개 데이터
```

```
(455, 8)
```

	caption	div	avg_data	max_data	min_data	stddev_data	cv_data	af_data
0	중도매인 판매가격 > 과일류 > 수박 > 수박 > 상품 > 1개	2016	16,230	23,000	11,000	2,447	15.08	1.09
1	중도매인 판매가격 > 과일류 > 수박 > 수박 > 상품 > 1개	2017	15,235	25,000	9,000	3,666	24.06	1.78
2	중도매인 판매가격 > 과일류 > 수박 > 수박 > 상품 > 1개	2018	18,715	33,000	9,000	4,834	25.83	2.67
3	중도매인 판매가격 > 과일류 > 수박 > 수박 > 상품 > 1개	2019	18,626	28,000	12,000	2,638	14.16	1.33
4	중도매인 판매가격 > 과일류 > 수박 > 수박 > 상품 > 1개	2020	16,511	28,300	6,300	5,620	34.04	3.49

※ 데이터 가공 및 정제

[불필요한 데이터 제거]

```
# 도매가 제거
mask1 = dfset ['caption'].str.contains('소매가격')
filter_df = dfset .loc [mask1 , :]

# 불필요한 데이터 제거
filter_df = filter_df .drop (['stddev_data','cv_data','af_data'], axis =1 )

# 컬럼확인
print (filter_df .columns )
```

```
Index(['caption', 'div', 'avg_data', 'max_data', 'min_data'], dtype='object')
```

```
# 컬렴명 변경
```

```
filter_df = filter_df .rename (columns ={'caption':'과일이름', 'div':'연도','avg_data':'평균가격',
'max_data':'최대가격', 'min_data':'최소가격'})

# 년도 데이터의 고유값 확인
filter_df ['연도'].unique()
```

```
array(['2016', '2017', '2018', '2019', '2020', '2021', '평년'], dtype=object)
```

```
# 평년 데이터 삭제
filter_df = filter_df [filter_df ['연도'] != '평년']
# 년도 데이터의 고유값 확인
filter_df ['연도'].unique()
```

```
array(['2016', '2017', '2018', '2019', '2020', '2021'], dtype=object)
```

[데이터 타입 변경]

```
# 데이터 타입 확인
print (filter_df .dtypes )
```

```
과일이름    object
연도      object
평균가격    object
최대가격    object
최소가격    object
dtype: object
```

```
# 데이터 타입 변경을 위해 숫자가 아닌 데이터 삭제
filter_df .loc [:, '평균가격':'최소가격'] = filter_df .loc [:, '평균가격':'최소가격'].apply (lambda x : x
.str.replace(',', ''))
filter_df .head ()

# 데이터 타입 변경
filter_df_copy = filter_df .astype ({'과일이름':'string', '연도':'int', '평균가격':'int', '최대가격':'int',
'최소가격':'int'})

#데이터 타입 확인
print (filter_df_copy .dtypes )
```

```
과일이름    string
연도       int32
평균가격     int32
최대가격     int32
최소가격     int32
dtype: object
```

[카테고리 데이터에서 과일명 추출]

```
# 과일 이름 추출
filter_df_copy ['과일이름'] = filter_df_copy ['과일이름'].str.split('>').str[2]


filter_df_copy
```

	과일이름	연도	평균가격	최대가격	최소가격
7	수박	2016	19720.0	28250.0	13717.0
8	수박	2017	18941.0	30375.0	12775.0
9	수박	2018	21322.0	30833.0	12925.0
10	수박	2019	21704.0	31050.0	15850.0
11	수박	2020	21077.0	29450.0	14933.0
...	...	...	...	...	...
442	감귤	2019	5677.0	6960.0	4765.0
443	감귤	2020	6773.0	7873.0	5900.0
444	감귤	2021	8077.0	9700.0	4950.0
452	감귤	2020	5211.0	6840.0	4710.0
453	감귤	2021	6563.0	8160.0	4260.0

[카테고리 데이터에서 과일명 추출]

```
# 연도 고유값 확인
filter_df_copy ['연도'].unique()
```

```
array([2016, 2017, 2018, 2019, 2020, 2021])
```

```
# 그룹별 연도의 평균가격
result = filter_df_copy .groupby (['과일이름', '연도']).mean ().round (2 )
result.head(10)
```

		평균가격	최대가격	최소가격
과일이름	연도			
감귤	2016	2680.00	3945.00	1970.50
	2017	3259.50	6205.00	2453.00
	2018	3663.00	4396.75	2794.62
	2019	3611.00	4458.00	3034.33
	2020	4248.50	5405.75	3705.00
	2021	4957.75	6468.75	3367.50
딸기	2016	1071.50	1531.50	764.50
	2017	1087.50	1750.00	733.00
	2018	1072.50	1636.50	764.00
	2019	1020.00	1629.50	668.50

```
# 과일명과 컬럼 분리
fruits = result .index .get_level_values ('과일이름').unique ()
years = result .index .get_level_values ('연도').unique ()
colum_names = result .columns
print (fruits )
print (years )
print (colum_names )
```

```
Index([' 감귤 ', ' 딸기 ', ' 배 ', ' 복숭아 ', ' 사과 ', ' 수박 ', ' 포도 '], dtype='object',
name='과일이름')Int64Index([2016, 2017, 2018, 2019, 2020, 2021], dtype='int64', name='연도')Index(['평균가격',
'최대가격', '최소가격'], dtype='object')
```

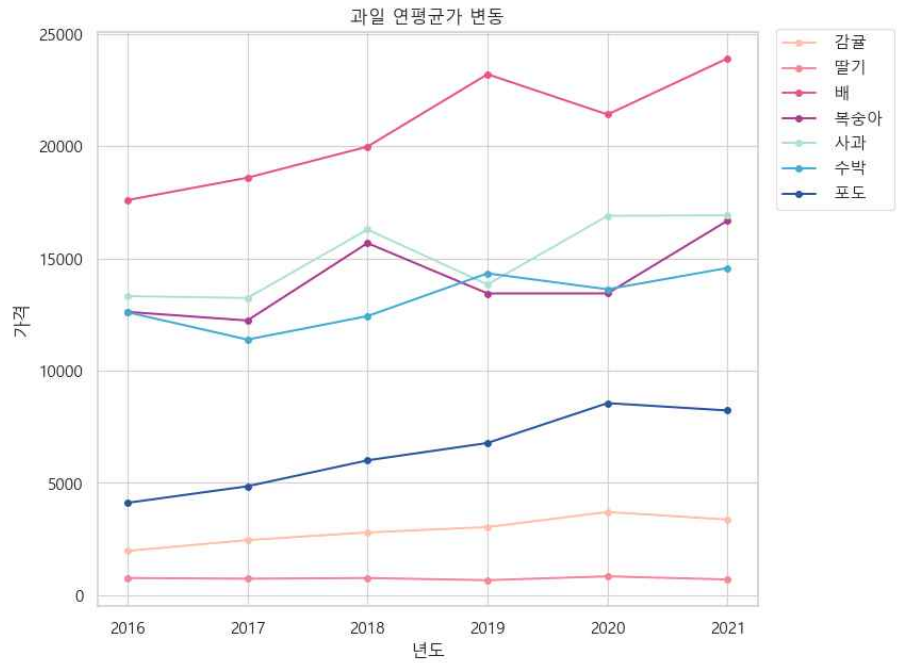
※ 데이터 분석 및 시각화

[과일의 연도별 가격 살펴보기]

```
import pandas as pd
import matplotlib .pyplot as plt
df = result .copy ()
# MultiIndex에서 각 레벨의 값들을 가져오기
years = df .index .get_level_values ('연도').unique ()
fruits = df .index .get_level_values ('과일이름').unique ()
# 다중 막대 그래프 생성
fig , ax = plt .subplots (figsize =(12 , 6 ))
bar_width = 0.1  # 막대 너비
opacity = 1
colors = ['#f7bba6', '#ed8495', '#e05286', '#a73b8f', '#aadacc', '#44a7cb', '#2a5599']  # 색상 리스트
# 각 연도별로 각 과일에 대한 '판매량'과 '평균가격'을 다중 막대 그래프로 표현
for i , fruit in enumerate (fruits ):
   x_positions = range (len (years ))
   prices = df .loc [fruit , '평균가격']
   # '평균가격' 막대 그래프
   ax .bar ([pos + bar_width /2 + i *bar_width for pos in x_positions ], prices , bar_width ,
       alpha =opacity , label =f '{fruit }', color =colors [i ])
# X 축에 과일 이름 표시
ax .set_xticks ([pos + bar_width *(len (years )-1 )/2 for pos in x_positions ])
ax .set_xticklabels (years )
ax .set_xlabel ('연도')
ax .set_ylabel ('평균가격')
ax .set_title ('연도별 과일 평균가격')
ax .legend ()
plt .show ()
```

[과일의 연도별 가격 변동 추이]

```
# 과일별 연평균가 변동추이 그래프 그리기
colors = ['#f7bba6', '#ed8495', '#e05286', '#a73b8f', '#aadacc', '#44a7cb', '#2a5599']  # 색상 리스트
for i , fruit in enumerate (fruits ):
   fruit_df = result [colum_names [2 ]].xs (fruit , level ='과일이름')
   fruit_df .plot (kind ='line', marker ='o', markersize =4 , figsize =(8 , 7 ), label =fruit , color =colors [i ])
plt .title ('과일 연평균가 변동')
plt .xlabel ('년도')
plt .ylabel ('가격')
plt .legend (title ='년도')
plt .legend (loc ='upper right', bbox_to_anchor =(1.22 , 1.02 ))
plt .show ()
```



결 과

전반적으로 과일의 평균 가격이 상승하는 경향을 보인다. 모든 과일의 가격이 2016년 대비 2021년 상승한 결과를 보였다. 그 중 배의 가격이 가장 많이 상승했다.

[과일별 최대값 최소값]

```
fig = plt .figure (figsize =(18 , 24 ))
# 서브플롯
ax1 = fig .add_subplot (4 , 2 , 1 )
ax2 = fig .add_subplot (4 , 2 , 2 )
ax3 = fig .add_subplot (4 , 2 , 3 )
ax4 = fig .add_subplot (4 , 2 , 4 )
ax5 = fig .add_subplot (4 , 2 , 5 )
ax6 = fig .add_subplot (4 , 2 , 6 )
ax7 = fig .add_subplot (4 , 2 , 7 )
# 각 과일에 대해 서브플롯에 최대가격과 최소가격을 다중 바 그래프로 그림
for ax , fruit in zip ([ax1 , ax2 , ax3 , ax4 , ax5 , ax6 , ax7 ], fruits ):
    min_prices = df .loc [fruit , '최소가격']
    max_prices = df .loc [fruit , '최대가격']
    # 바 차트 그리기
    ax .bar (years - 0.2 , min_prices , width =0.4 , label ='최소가격', alpha =.85 , color ='#786be1')
    ax .bar (years + 0.2 , max_prices , width =0.4 , label ='최대가격', alpha =.85 , color ='#f05c94')
    ax .set_title (f '{fruit }의 최대/최소 가격')
    ax .set_ylabel ('가격')
    ax .legend ()
# X 축 연도 표시
ax6 .set_xticks (years )
ax6 .set_xticklabels (years )
ax6 .set_xlabel ('연도')
plt .tight_layout ()
plt .show ()
```

* 좌측 상단의 그래프부터 감귤, 딸기, 배, 복숭아, 사과, 수박, 포도 순으로 연도별(2016~2012) 최대 가격(핑크색)과 최소가격(보라색)을 보여주는 그래프

결 과

각 과일의 최소값은 최대값에 비해 안정적으로 유지되거나 감소하는 경향을 보이지만, 최대값은 증가하는 경향을 보이고 있다. 특히 2021년에 최대값이 크게 증가하는 경향을 보였다.
딸기와 수박의 최소값, 최대값이 매년 두배 가량의 차이를 보이며 가장 큰 격차를 보이고 있다.

2. 과일 가격변화와 식품 물가의 상관 관계 분석

2.2 과일 가격과 식품 물가 상관 관계 분석

※ 데이터 수집 : 사용한 데이터 정보

파일명 : 물가상승률.xlsx

```
# 파일 경로와 시트명 설정
file_path2 = '물가상승률.xlsx'# 파일경로

# Excel 파일 읽기 : 연령별 데이터
price =pd .read_excel (file_path2 , header =2 , index_col = 0 , skipfooter =3 )

# 데이터프레임 확인
price
```

	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
소비자물가 총지수(2020=100)	91.8	93.0	94.2	94.9	95.8	97.6	99.1	99.5	100.0	102.5
소비자물가상승률(%)	2.2	1.3	1.3	0.7	1.0	1.9	1.5	0.4	0.5	2.5
식료품/비주류음료	4.0	0.9	0.3	1.6	2.3	3.4	2.8	0.0	4.4	5.9
주류 및 담배	1.5	1.7	-0.1	50.1	0.7	1.5	0.3	0.6	0.3	0.4
의류 및 신발	4.8	2.9	4.0	1.3	1.8	1.1	1.1	0.1	0.7	0.6
주택·수도·전기 및 연료	4.6	3.5	2.9	-0.6	-0.8	1.7	0.7	1.2	0.5	1.6
가정용품 및 가사 서비스	2.9	0.3	2.1	2.6	1.6	1.1	2.3	2.1	0.0	1.9
보건	0.9	0.4	0.7	1.3	1.0	0.9	-0.1	0.5	1.5	-0.1
교통	3.2	-0.5	-1.6	-7.8	-2.2	3.6	2.4	-1.8	-1.8	6.3
통신	-2.6	-0.1	-0.1	-0.2	0.1	0.3	-0.9	-2.3	-2.1	-0.9
오락 및 문화	0.3	1.0	0.4	-0.5	1.8	0.1	0.5	-0.2	-1.0	0.4
교육	1.4	1.2	1.5	1.7	1.6	1.1	1.4	0.5	-2.1	0.9
음식 및 숙박	1.2	1.6	1.5	2.3	2.5	2.4	3.0	1.8	0.9	2.7
기타 상품 및 서비스	-3.3	0.5	3.1	2.7	3.4	2.8	0.6	1.6	2.0	2.0

※ 데이터 가공 및 정제

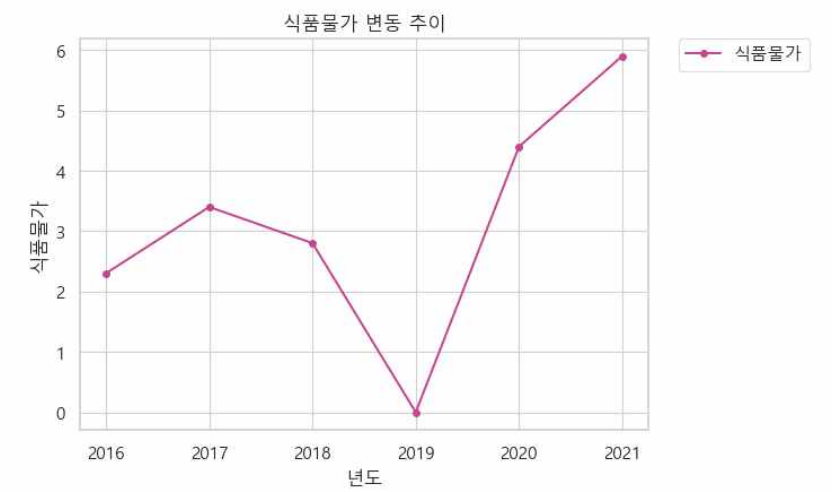
```
# 필요한 데이터만 추출
price_df = price .loc [['식료품/비주류음료'],'2016':]
# x축 y축 치환
price_df = price_df .T
# 컬럼명 변경
price_df = price_df .rename (columns
={'식료품/비주류음료':'식품물가'})
price_df
```

	식품물가
2016	2.3
2017	3.4
2018	2.8
2019	0.0
2020	4.4
2021	5.9

※ 데이터 분석 및 시각화

[식품 물가 변동 추이]

```
# 식품물가 변동 추이 그래프 그리기
price_df .plot (kind ='line', marker ='o', markersize =4 , figsize =(6.5 , 4.5 ), label =fruit , color ='#c0458a')
plt .title ('식품물가 변동 추이')
plt .xlabel ('년도')
plt .ylabel ('식품물가')
plt .legend (title ='년도')
plt .legend (loc ='upper right', bbox_to_anchor =(1.3 , 1.02 ))
plt .show ()
```



※ 데이터 가공 및 정제

```
# 인덱스 데이터 타입 확인
price_comp_df = price_df .copy ()
price_comp_df
print (price_comp_df .index .dtype )
```

```
object
```

```
# 인덱스 데이터 타입 변경
price_comp_df .index = price_comp_df .index .astype (int )
print (price_comp_df .index .dtype )
```

```
int64
```

```
# 데이터 프레임 합치기
new = pd .DataFrame ()
# 각 과일에 대한 평균가격 열 추가
for fruit in fruits :
    new [fruit ] = result .xs (fruit , level ='과일이름')['평균가격']
new .index .name = ''
result_df = pd .merge (new , price_comp_df , left_index =True , right_index =True , how ='outer')

# 결과 출력
```

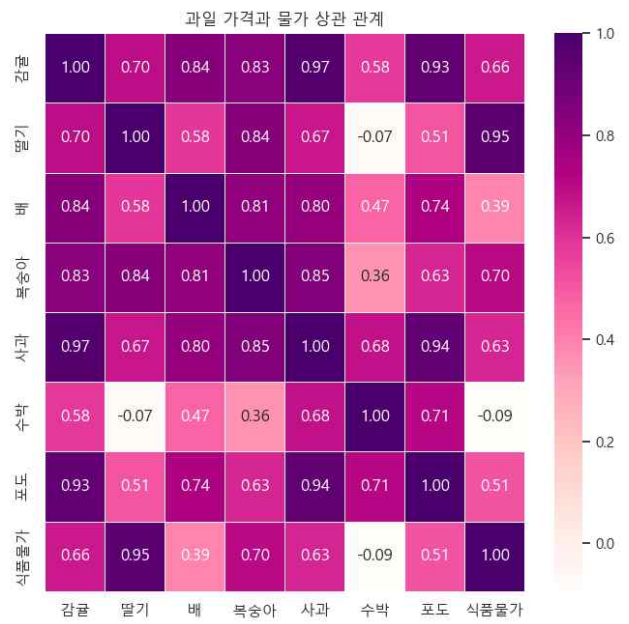
```
result_df
```

	감귤	딸기	배	복숭아	사과	수박	포도	식품물가
2016	2680.00	1071.5	22771.00	15193.0	16295.33	18417.5	5090.07	2.3
2017	3259.50	1087.5	23908.75	15496.5	16361.33	17703.5	5552.71	3.4
2018	3663.00	1072.5	24062.25	17834.5	18703.50	19965.0	6915.67	2.8
2019	3611.00	1020.0	28262.50	15614.0	18023.17	20238.0	7784.62	0.0
2020	4248.50	1096.0	25507.25	16366.0	19856.00	19957.0	9985.70	4.4
2021	4957.75	1179.0	33359.00	22065.0	21546.17	19684.5	9997.05	5.9

※ 데이터 분석 및 시각화

[과일 가격과 물가 상관 관계 분석]

```
# 그래프 그리기
correlation_matrix = result_df .corr ()
# 히트맵
plt .figure (figsize =(8 , 7.5 ))
sns .heatmap(correlation_matrix , annot =True , cmap
='RdPu', fmt ='.2f', linewidths =0.5 )
plt .title ('과일 가격과 물가 상관 관계')
plt .show ()
```



결 과

식품물가는 대부분의 과일과 양의 상관 관계를 가지고 있다. 특히, 딸기와의 상관 관계가 0.95로 매우 높다. 이는 식품물가와 딸기의 가격이 밀접하게 연관되어 있다는 것을 나타낸다. 반대로 수박은 식품 물가와 음의 상관 관계를 보이고 있어 식품물가와의 관련성이 없다는 것을 보여주며, 수박의 경우 딸기와의 음의 상관관계를 보여, 가격이 반대로 움직인 다는 것을 보여준다.

3. 과일별 판매량 분석

3.1 홈플러스 크롤링

※ 데이터 수집

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time
import re

service = Service(executable_path ="C:\chromedriver\chromedriver.exe")
options = webdriver.ChromeOptions()
options .add_argument('--headless')
options .add_argument('--no-sandbox')
options .add_argument('--disable-dev-shm-usage')
driver = webdriver.Chrome(service =service , options =options )
url = 'https://front.homeplus.co.kr/list?categoryId=100001&categoryDepth=1'#과일
driver .get(url )
time .sleep (3 )
dataList = [] # 상품정보를 담을 리스트
def readData ():
   # 데이터 읽어오기
   elements = driver .find_elements(By.CSS_SELECTOR, '.unitItemBox')
   for element in elements :
     try :
       fruitName = element .find_element(By.CSS_SELECTOR, '.css-12cdo53-defaultStyle-Typography-ellips').text
     except :
       fruitName = np .NaN
     try :
       monthly_purchase = element .find_element(By.XPATH, './/div[contains(@class,
"prodScoreWrap")]/span[last()]').text
     except :
       monthly_purchase = np .NaN
     data = {'상품명': fruitName , '한달판매량': monthly_purchase }
     dataList .append (data )
cnt = 0
while True :
   readData () # 데이터 읽기
   next_page_buttons = driver .find_elements(By.CSS_SELECTOR, '.css-1ij9dss-number') # 현재 페이지를 제외한 페이징
버튼 수
   try :
     # cnt = cnt + 1
     # print(cnt,'페이지 출력완료')
     next_page_button = driver .find_element(By.CSS_SELECTOR, '.btnNext') # Next 버튼
     driver .execute_script("arguments[0].click();", next_page_button )
     time .sleep (1 )  # 페이지가 로드되기를 기다림
```

```
    except Exception as e :
      print ('No Next Button')
      for btn in next_page_buttons :
        # Next 버튼이 삭제 된 후의 처리(남은 pagination 클릭)
        # cnt = cnt + 1
        # print(cnt,'페이지 출력완료')
        driver .execute_script("arguments[0].click();", btn )
        time .sleep (1 )  # 페이지가 로드되기를 기다림
        readData () # 데이터 읽기
      break

print ('크롤링 끝', len (dataList ), '개 데이터 수집 완료')

df = pd .DataFrame (dataList )
df .tail ()
df .to_csv ('./fruits_info.csv') # csv 저장
print ('csv 저장 완료')
```

3. 과일별 판매량 분석

3.2 품목별 판매량 시각화

※ 데이터 정제

[불필요한 데이터 제거]

```
# 텍스트 정제 함수 : 한글 이외의 문자는 전부 제거
def text_cleaning (text ): # 한글의 정규표현식으로 한글만 추출
    hangul = re .compile ('[^ ㄱ-ㅣ가-힣]+')
    result = hangul .sub ('', text )
    return result
def num_cleaning (text ): # 숫자 정제 함수 : 숫자 이외의 문자는 전부 제거
    if pd .isna (text ):  # NaN 값이면 0으로 대체
      return 0
    else :
      # 숫자 정규표현식으로 숫자만 추출
      numbers = re .compile ('[^0-9]+')
      result = numbers .sub ('', text )
      return result
# 함수를 적용하여 상품명에서 한글만 추출
df ['상품명'] = df ['상품명'].apply (lambda x : text_cleaning (x ))
df ['한달판매량'] = df ['한달판매량'].apply (lambda x : num_cleaning (x ))
df .head (5 )
```

	상품명	한달판매량
0	스테비아 대추방울토마토 팩	36038
1	신선농장 샤인머스캣 박스	9364
2	제주밀감 박스	65027
3	타이벡밀감 박스	36699
4	고당도 스위트마운틴 바나나필리핀 송이	55392

```
# 과일이름
fruits_name = ['수박', '딸기', '사과', '배', '복숭아', '포도', '샤인머스켓', '감귤']

# 특정 문자열이 포함되지 않은 행을 제거
filtered_dataList = df [df ['상품명'].str.contains('|'.join (fruits_name ))]

# 상품명 변경
for fruit in fruits_name :
   filtered_dataList .loc[filtered_dataList ['상품명'].str.contains(fruit ), '상품명'] = fruit
filtered_dataList .head()
```

	상품명	한달판매량
5	사과	6529
8	딸기	15496
10	배	5583
17	사과	61883
27	포도	2232

```
print (filtered_dataList .dtypes)
```

```
상품명      object
한달판매량    object
dtype: object
```

```
# 중복된 과일 이름의 한달 판매량 합산
new_df = filtered_dataList .groupby('상품명', as_index =False ).sum()
new_df

# 상품명을 인덱스로 지정
new_df = new_df .set_index('상품명')
new_df

# 샤인머스켓은 포도로 통합
new_df .loc['포도'] = new_df .loc['포도'] + new_df .loc['샤인머스켓']
new_df = new_df .drop('샤인머스켓', axis =0 )

new_df = new_df .sort_values(by ='한달판매량', ascending =False )
new_df

new_df_copy = new_df .T

print (new_df_copy .shape)
new_df_copy
```

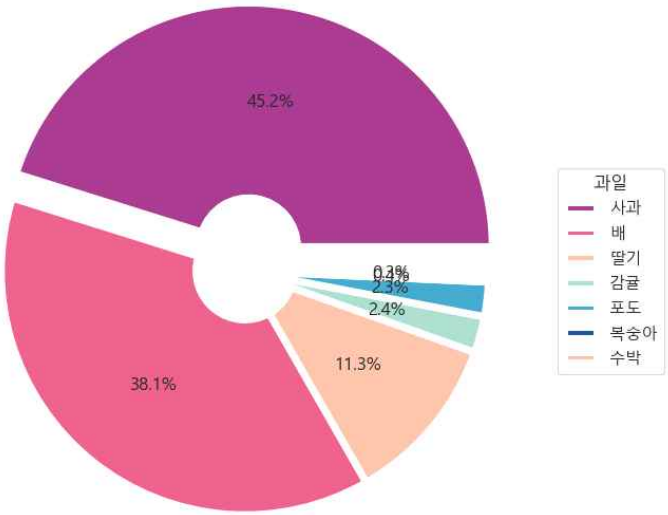
```
(1, 7)
```

상품명	사과	배	딸기	감귤	포도	복숭아	수박
한달판매량	80352	67840	20087	4291	4056	671	621

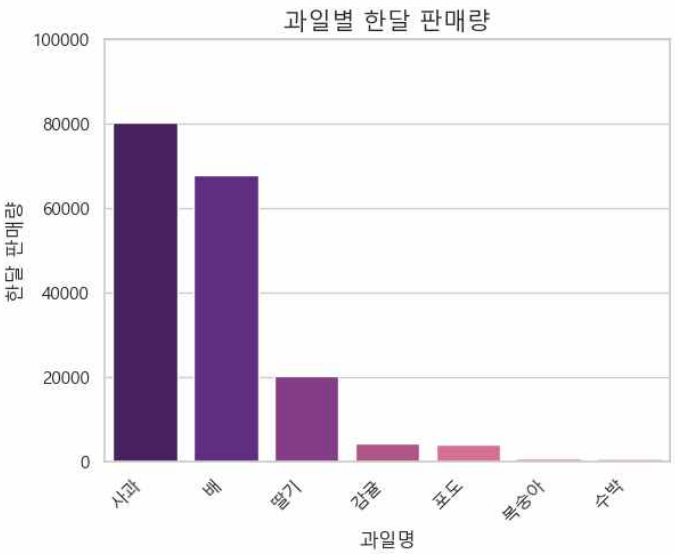
※ 데이터 분석 및 시각화

[한달간 가장 많이 팔린 품목 순위]

```
# 파이 차트 그리기
explode = [0.1 , 0 , 0 , 0 , 0 , 0 , 0 ]
colors = ['#a73b8f',  '#e8608a', '#f8c1a8', '#aadacc',
'#44a7cb', '#2a5599', '#f7bba6',]  # 색상 리스트
wedgeprops ={'width': 0.8 , 'edgecolor': 'w',
'linewidth': 5 }
plt .figure (figsize =(8 , 7.5 ))
wedges , texts , autotexts = plt .pie (new_df_copy
.loc['한달판매량'], labels =None , explode =explode ,
autopct ='%.1f%%', colors =colors , wedgeprops
=wedgeprops )
plt .legend (wedges , new_df_copy .columns, title
="과일", loc ="center left", bbox_to_anchor =(1 , 0 ,
0.5 , 1 ))
plt .setp (autotexts , size =12 )
plt .show ()
```




```
# 판매량 차트 그리기
colors = ['#f8c1a8', '#ef9198', '#e8608a', '#c0458a',
'#8f3192', '#63218f', '#4b186c' ]
ax = sns .barplot(data =new_df , x =new_df .index, y
='한달판매량', palette =reversed (colors ))
ax .set_title('과일별 한달 판매량', fontsize =15 )
ax .set_xlabel('과일명', fontsize =12 )
ax .set_ylabel('한달 판매량', fontsize =12 )
ax .set_xticklabels(ax .get_xticklabels(), rotation =45
, ha ="right")  # x축 눈금 지정
ax .set_yticks([0 , 20000 , 40000 , 60000 , 80000 ,
100000 ])  # y축 눈금 지정
plt .show ()
```



결 과

사과의 한 달 판매량이 가장 높으며, 두 번째로 배, 딸기, 감귤, 포도 등의 순서를 보이고 있다.

4. 과일별 소비량 분석

4.1 소비량 시각화

※ 데이터 수집 : 사용한 데이터 정보

출처 : atfis 식품산업통계정보
자료명 : 식품원료별 사용량(총/국산/수입) API

```
# 품종별 등급코드
kind_codes = ['AF0501', 'AF0503', 'AF0505', 'AF0506', 'AF0509', 'AF0507', 'AF0508']
# 221: 수박, 226: 딸기, 411: 사과, 412: 배, 413: 복숭아, 414: 포도, 415: 감귤
# API 요청 보내기
base_url = 'https://www.atfis.or.kr/home/api/consumption/basic.do'
apiKey = 'bfTU1VTsA1t5Uj00HMof83LQCjL/F5tvD4SdyrBUYj8='
beginYear = 2016
endYear = 2021
dfset2 = pd .DataFrame ()  # 빈 데이터프레임
for kind_code in kind_codes : # 품종별 등급코드 순회
   try :
     # 요청
     response = requests .get (
       f '{base_url }?apiKey={apiKey }&cnsmpMtralCd={kind_code }&beginYear={beginYear }&endYear={endYear }'
     )
     # 응답 처리
     if response .status_code == 200 :
       # JSON 데이터를 DataFrame으로 변환하여 dfset2에 추가
       data = response .json ()
       df_temp = pd .DataFrame (data )
       dfset2 = pd .concat ([dfset2 , df_temp ], ignore_index =True )
     else :
       print ('Error occurred:', response .status_code )
   except requests .exceptions .RequestException as e :
     print ('Request failed:', e )
   except Exception as e :
     print ('An error occurred:', e )

dfset2 .head ()
```

	cnsmpYear	cnsmpCd	cnsmpMtralUpperNm	cnsmpMtralNm	cnsmpTotUsgqty	cnsmpLocalusgqty	cnsmpImportUsgqty	cnsmpLocalRelimp
0	2021	AF0501	과일.채소류 및 과일 채소류 유래 식품 소재	수박	265.0	265.0	0.0	100.0
1	2020	AF0501	과일.채소류 및 과일 채소류 유래 식품 소재	수박	351.0	351.0	0.0	0.1
2	2019	AF0501	과일.채소류 및 과일 채소류 유래 식품 소재	수박	297.0	297.0	0.0	100.0
3	2018	AF0501	과일.채소류 및 과일 채소류 유래 식품 소재	수박	9.0	9.0	0.0	100.0
4	2017	AF0501	과일.채소류 및 과일 채소류 유래 식품 소재	수박	30.0	30.0	0.0	100.0

※ 데이터 정제

```
#데이터 확인
print (dfset2 .shape )
print (dfset2 .index )
print (dfset2 .columns )
print (dfset2 ['cnsmpMtralNm'].unique())
```

```
(42, 8)
RangeIndex(start=0, stop=42, step=1)
Index(['cnsmpYear', 'cnsmpCd', 'cnsmpMtralUpperNm', 'cnsmpMtralNm', 'cnsmpTotUsgqty', 'cnsmpLocalusgqty',
'cnsmpImportUsgqty', 'cnsmpLocalRelimp'], dtype='object')
['수박' '딸기' '사과' '배' '복숭아' '포도' '감귤']
```

```
#불필요한 데이터 제거
dfset2_copy = dfset2 .drop (columns =['cnsmpCd', 'cnsmpMtralUpperNm', 'cnsmpLocalRelimp'])
dfset2_copy .columns =['연도', '과일', '총 소비량', '국산', '수입산']\
```

```
# 데이터 타입 변경
dfset2_copy = dfset2_copy .astype ({'연도':'int', '과일':'string', '총 소비량':'int', '국산':'int',
'수입산':'int'})

# 그룹화
dfset2_gr = dfset2_copy .groupby (['과일','연도']).sum ()
dfset2_gr .head (15 )
```

과일	연도	총 소비량	국산	수입산
감귤	2016	141419	141419	0
	2017	129091	129091	0
	2018	147250	147239	11
	2019	141042	141042	0
	2020	143496	143496	0
	2021	137601	137601	0
딸기	2016	24595	21869	2726
	2017	24425	22528	1897
	2018	28778	26997	1780
	2019	24669	24205	464
	2020	24472	24041	431
	2021	23095	21454	1640
배	2016	11270	11270	0
	2017	10639	10639	0
	2018	13386	13386	0

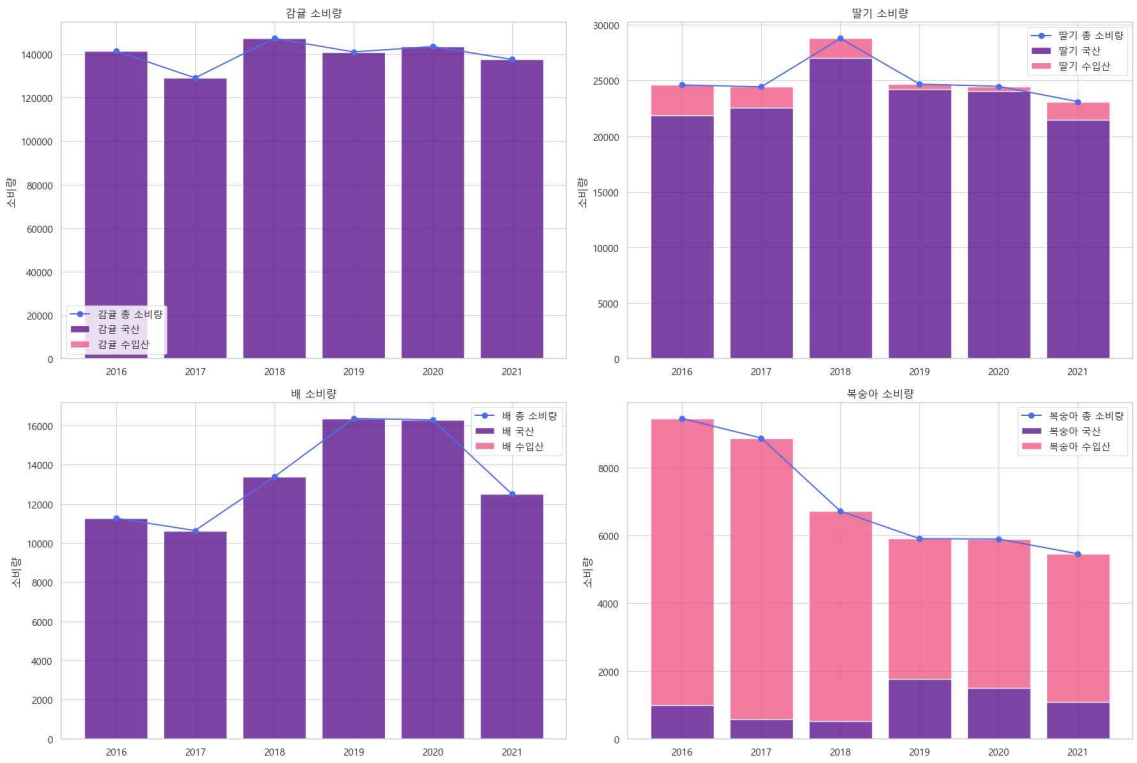
```
# 과일명과 컬럼 분리
fruits = dfset2_gr .index .get_level_values ('과일').unique ()
years = dfset2_gr .index .get_level_values ('연도').unique ()
colum_names = dfset2_gr .columns
print (fruits )
print (years )
print (colum_names )
```

```
Index(['감귤', '딸기', '배', '복숭아', '사과', '수박', '포도'], dtype='string', name='과일')
Int64Index([2016, 2017, 2018, 2019, 2020, 2021], dtype='int64', name='연도')
Index(['총 소비량', '국산', '수입산'], dtype='object')
```

※ 데이터 분석 및 시각화

[2016~2021년의 과일별 소비량]

```
fig = plt .figure (figsize =(18 , 24 ))
# 서브플롯
ax1 = fig .add_subplot (4 , 2 , 1 )
ax2 = fig .add_subplot (4 , 2 , 2 )
ax3 = fig .add_subplot (4 , 2 , 3 )
ax4 = fig .add_subplot (4 , 2 , 4 )
ax5 = fig .add_subplot (4 , 2 , 5 )
ax6 = fig .add_subplot (4 , 2 , 6 )
ax7 = fig .add_subplot (4 , 2 , 7 )
# 각 과일에 대해 국산 소비량과 수입 소비량을 다중 바 그래프로 그림
for ax , fruit in zip ([ax1 , ax2 , ax3 , ax4 , ax5 , ax6 , ax7 ], fruits ):
    domestic_consumption = dfset2_gr .loc [fruit , '국산']
    import_consumption = dfset2_gr .loc [fruit , '수입산']
    # 바 차트 그리기
    ax .bar (years , domestic_consumption , label =f '{fruit } 국산', alpha =.85 , color ='#63218f')
    ax .bar (years , import_consumption , label =f '{fruit } 수입산', bottom =domestic_consumption , alpha =.85 ,
color ='#e8608a')

    # 선 그래프 그리기 (총 소비량)
    total_consumption = dfset2_gr .loc [fruit , '총 소비량']
    ax .plot (years , total_consumption , label =f '{fruit } 총 소비량', color ='#4f6ae0', marker ='o')
    ax .set_title (f '{fruit } 소비량')
    ax .set_ylabel ('소비량')
    ax .legend ()
plt .tight_layout ()
plt .show ()
```



결 과

감귤, 딸기, 포도의 소비량은 전반적으로 안정적이며, 사과의 소비량으 연도에 따라 증가하는 추세를 보이고 있다. 복숭아의 소비량은 감소 추세이며, 국산보다 수입산이 많은 비중을 차지하고 있다.

5. 과일 재배지 분포 시각화

5.1 과일 재배지 시각화

※ 데이터 수집 : 사용한 데이터 정보

자료명 : 농가리스트
파일명1 : 농가리스트_과실류.xls
파일명2 : 농가리스트_과일과채류.xls

```
# 파일 경로와 시트명 설정
file_path1 = '농가리스트_과실류.xls'# 파일경로1
# Excel 파일 읽기 : 농가리스트_과실류
nong_list1 = pd .read_excel (file_path1 , header =1 )
# 데이터프레임 확인
print (nong_list1 .shape )
nong_list1 .head ()
```

```
(1865, 5)
```

	조직유형	농가명	생산품목	소재지	전화번호
0	농협	제주감귤농업협동조합	감귤	제주특별자치도 서귀포시 강정동155 제주감귤농협	064-739-5401
1	농협	제주지역조합공동사업법인	감귤	제주특별자치도 제주시 삼도일동794-3	064-720-1335
2	농협	김해시조합공동사업법인	단감	경상남도 김해시 서상동48-2	055-327-1500
3	농협	농협경제지주㈜경남지역본부	단감	경상남도 창원시 성산구 신월동95	055-268-1622
4	농협	청도군조합공동사업법인	떫은감	경상북도 청도군 화양읍 유등리	054-373-4983

```
file_path2 = '농가리스트_과일과채류.xls'# 파일경로2
# Excel 파일 읽기 : 농가리스트_과일과채류
nong_list2 = pd .read_excel (file_path2 ,
           header =1 )
# 데이터프레임 확인
print (nong_list2 .shape )
nong_list2 .head ()
```

```
(663, 5)
```

	조직유형	농가명	생산품목	소재지	전화번호
0	농협	논산시농협조합공동사업법인	딸기	충청남도 논산시 연산면 청동리467	041-735-8602
1	농협	농업회사법인주식회사조이팜	딸기	경상남도 산청군 단성면 사월리25-2	055-973-1033
2	농협	농협경제지주㈜하동군연합사업단	딸기	경상남도 하동군 하동읍 읍내리34	055-883-1142
3	농협	곡성군연합사업단	메론	전라남도 곡성군 곡성읍 읍내리264	061-360-4708
4	일반법인	광일영농조합	수박	경상남도 함안군 법수면 강주리1339-1	055-582-5440

```
# 두 테이블 연결
nong_list = pd .concat ([nong_list1 , nong_list2 ])
nong_list .shape
```

```
(2528, 5)
```

※ 데이터 정제

[결측값 제거 및 중복행 제거]

```
# 결측치 제거
nong_list = nong_list .replace ('-',np .nan)
nong_list = nong_list .dropna (subset =['생산품목', '소재지'])
nong_list .shape
```

```
(2455, 5)
```

```
# 중복행 체크
nong_list .duplicated (subset =['생산품목', '소재지']).sum ()
```

```
872
```

```
# 중복행 제거
nong_list = nong_list .drop_duplicates (subset =['생산품목', '소재지'])
print (nong_list .shape )
print (nong_list .duplicated (subset =['생산품목', '소재지']).sum ())
```

```
(1583, 5)
0
```

[필요한 데이터 추출]

```
# 소재지 시 데이터 추출
nong_list ['소재지'] = nong_list ['소재지'].str.split(' ').str[0 ]
nong_list.head ()
```

	조직유형	농가명	생산품목	소재지	전화번호
0	농협	제주감귤농업협동조합	감귤	제주특별자치도	064-739-5401
1	농협	제주지역조합공동사업법인	감귤	제주특별자치도	064-720-1335
2	농협	김해시조합공동사업법인	단감	경상남도	055-327-1500
3	농협	농협경제지주㈜경남지역본부	단감	경상남도	055-268-1622
4	농협	청도군조합공동사업법인	떫은감	경상북도	054-373-4983

```
nong_list_cp = nong_list .copy ()
```

```
# 소재지 고유값 확인
print (nong_list_cp ['소재지'].unique())
# 숫자는 np.nan으로 치환
def process_text (text ):
   if re .search (r '\d', str (text )):  # 정규표현식으로 문자열에 숫자가 포함되어 있는지 확인
     return np .nan
   else :
     return text
# 함수 호출
nong_list_cp ['소재지'] = nong_list ['소재지'].apply (process_text )
nong_list_cp .dropna (inplace =True )
# 생산품목 고유값 확인
print (nong_list_cp ['소재지'].unique())
```

```
['제주특별자치도' '경상남도' '경상북도' '경기도' '충청남도' '전라북도' '충청북도' '688-31' '대구' '48'
 '충북' '전라남도' '인천광역시' '강원도' '세종특별자치시' '대전광역시' '광주광역시' '대구광역시' '울산광역시'
 '서울특별시' '부산광역시' '제주' '경남' '경북' '강원' '전남' '충남' '경기' '전북' '인천' '전남보성군']
['제주특별자치도' '경상남도' '경상북도' '경기도' '충청남도' '전라북도' '충청북도' '대구' '충북' '세종특별자치시'
 '강원도' '대전광역시' '전라남도' '서울특별시' '울산광역시' '부산광역시' '경남' '경북' '강원' '전남' '충남'
 '경기' '전북' '인천' '제주' '광주광역시' '대구광역시']
```

```
# 명칭 변경
nong_list_cp .loc [nong_list_cp ['소재지'] == '대구', '소재지'] = '대구광역시'
nong_list_cp .loc [nong_list_cp ['소재지'] == '충북', '소재지'] = '충청북도'
nong_list_cp .loc [nong_list_cp ['소재지'] == '강원', '소재지'] = '강원도'
nong_list_cp .loc [nong_list_cp ['소재지'] == '경남', '소재지'] = '경상남도'
nong_list_cp .loc [nong_list_cp ['소재지'] == '경북', '소재지'] = '경상북도'
nong_list_cp .loc [nong_list_cp ['소재지'] == '전북', '소재지'] = '전라북도'
nong_list_cp .loc [nong_list_cp ['소재지'] == '전남', '소재지'] = '전라남도'
nong_list_cp .loc [nong_list_cp ['소재지'] == '경기', '소재지'] = '경기도'
nong_list_cp .loc [nong_list_cp ['소재지'] == '충남', '소재지'] = '충청남도'
nong_list_cp .loc [nong_list_cp ['소재지'] == '인천', '소재지'] = '인천광역시'
nong_list_cp .loc [nong_list_cp ['소재지'] == '제주', '소재지'] = '제주특별자치도'

# 생산품목 고유값 확인
print (nong_list_cp ['소재지'].unique())
```

```
['제주특별자치도' '경상남도' '경상북도' '경기도' '충청남도' '전라북도' '충청북도' '대구광역시' '세종특별자치시' '강원도' '대전광역시' '전라남도' '서울특별시' '울산광역시' '부산광역시' '인천광역시' '광주광역시']
```

```
map_data = nong_list_cp ['소재지'].value_counts ().reset_index ()
map_data .columns = ['소재지', '농가수']
map_data.head ()
```

	소재지	농가수
0	경상북도	38
1	경남	34
2	경상남도	27
3	전라남도	23
4	제주특별자치도	22

```
map_data = map_data .astype ({'소재지':'string', '농가수':'int64'})
print (map_data .dtypes )
```

```
소재지    string
농가수     int64
dtype: object
```

※ 데이터 수집 : 사용한 데이터 정보

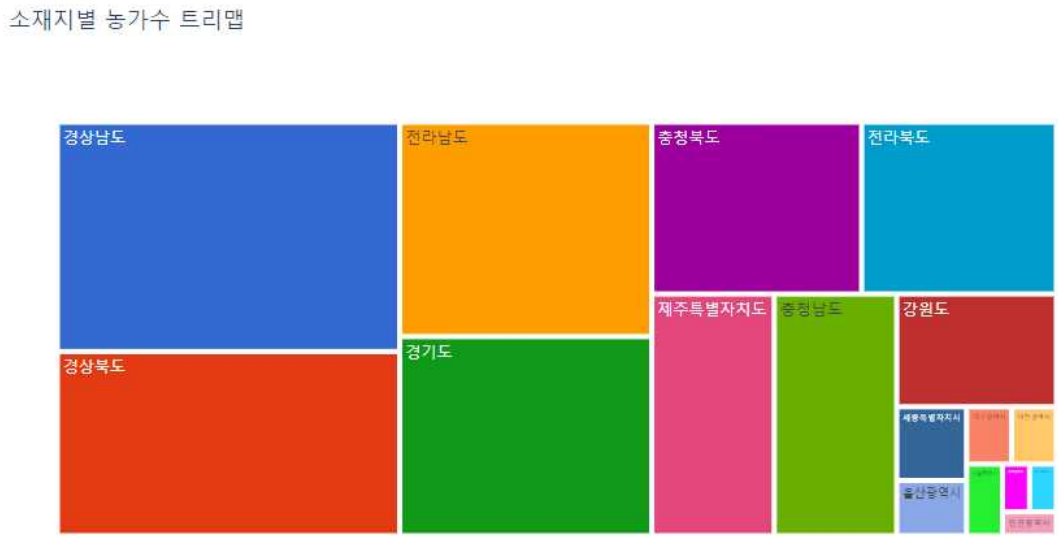
```
자료명 : 시군구 경계 정보
파일명 : TL_SCCO_CTPRVN.json
```

```
import json
# 시군구 경계 정보를 가진 json 파일
state_geo = './TL_SCCO_CTPRVN.json'

# # JSON 파일 읽기
sido_map = json .load (open (state_geo , encoding ='utf-8'))
sido_map ['features'][0 ]['properties']
```

※ 데이터 시각화

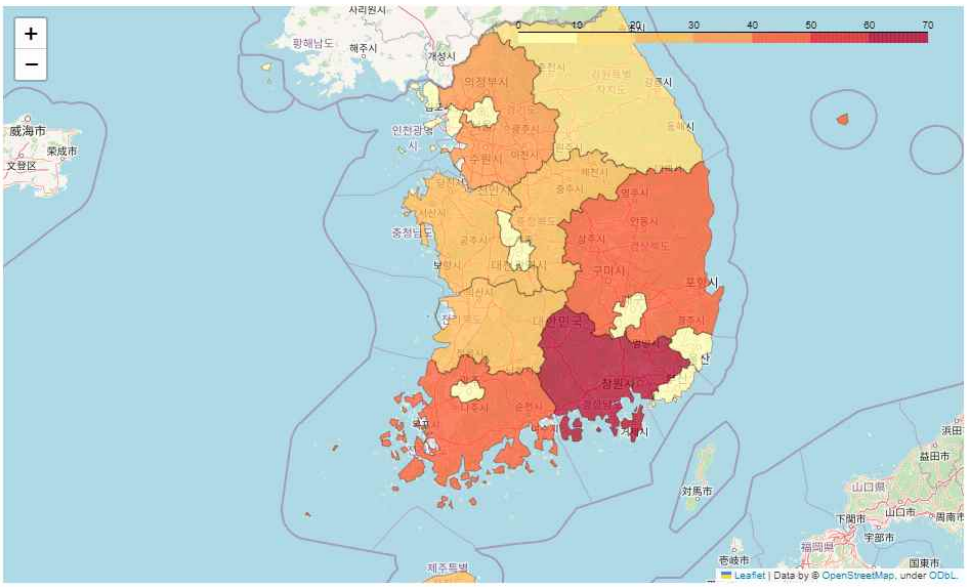
[과일 재배지 분포]

```
# plotly treemap 그래프 작성
import plotly.express as px
# Plotly Treemap 생성
fig = px .treemap(map_data , path =['소재지'], values ='농가수', title ='소재지별 농가수 트리맵',
color_discrete_sequence =px .colors.qualitative.G10)
fig .show()
```




```
import folium
# 지도 만들기
g_map = folium .Map(location =[36 ,127 ],
zoom_start =7 )
# Choropleth 클래스로 단계구분도 표시하기
folium .Choropleth(
            geo_data =sido_map ,  # 지도 경계
            data =map_data ,  # 표시하려는 데이터
            columns =['소재지', '농가수'],  # 열 지정
            name ='농가수',
            key_on ='feature.properties.CTP_KOR_NM',  # geo-json 파일의 특정 속성을 기준으로 데이터 매핑
            fill_color ='YlOrRd',
```

```
            fill_opacity =0.7 ,
            line_opacity =0.3 ,
            threshold_scale =[0 , 10 , 20 , 30 , 40 , 50 , 60 , 70 ],
           ).add_to(g_map )
g_map
```



결 과

경상남도의 재배지의 분포가 가장 높은 분포를 보였으며, 그 다음 경상북도, 전라남도, 경기도 순으로 높은 분포를 보였다. 서울, 인천을 포함한 수도권과 부산, 광주 등의 지역에서는 낮은 분포를 보였다.

6. 과일 특성에 따른 선호도 분류

6.1 산도, 당도, 횡경, 종경으로 분류하는 모델 만들기

※ 데이터 수집 : 사용한 데이터 정보

출처 : 공공데이터 포털
자료명1 : 농촌진흥청 국립원예특작과학원_복숭아 생육품질정보
자료명2 : 농촌진흥청 국립원예특작과학원_배 생육품질정보
자료명3 : 농촌진흥청 국립원예특작과학원_감귤생육품질정보
자료명4 : 농촌진흥청 국립원예특작과학원_포도 생육품질정보

[배 데이터 수집]

```
import requests
import xmltodict
import json
import pandas as pd
# 품종별 코드
years = ['2016', '2017', '2018', '2019', '2020', '2021']
kind_codes = ['pear01', 'pear02']
# API 요청 보내기
url = 'http://apis.data.go.kr/1390804/Nihhs_Fruit_Pear_GrwhInfo/pearAnalsDataList'
fruit_pear = pd .DataFrame ()  # 빈 데이터프레임
for year in years :
   for kind_code in kind_codes :
     params = {'serviceKey':
'rhKXU5xPur/LMP2M8j2kA/SY0OAo97wV4Br00ZJI92pJuhi5AsOBr1Xfc/HPASy3SJ+xlnzbgctADLwRg2LqNQ==', 'selyear': year ,
'spciesCode': kind_code }
     try : # 요청
       response = requests .get (url , params =params )
       if response .status_code == 200 :
         xml_data =response .content
         xml_dict = xmltodict .parse (xml_data ) # XML을 Python dictionary로 변환
         json_data = json .dumps (xml_dict , indent =2 ) # Python dictionary를 JSON 문자열로 변환
         df = pd .json_normalize (json .loads (json_data )['Response']['Body']['Model']) # JSON > DataFrame으로
         df ['과일'] = '배'
         # fruit_pear에 추가
         fruit_pear = pd .concat ([fruit_pear , df ], ignore_index =True )
       else :
         print (f 'Error occurred for year {year } and kind_code {kind_code }: {response .status_code }')
     except requests .exceptions .RequestException as e :
       print ('Request failed:', e )
     except Exception as e :
       print ('An error occurred:', e )
fruit_pear.to_csv ('./fruit_pear.csv') # csv 저장
fruit_pear.head ()
```

	farm_name	examin_datetm	avg_frut_wt	avg_lgdiamtr	avg_prbrd	avg_hrdnss	avg_brx	avg_prcn	과일
0	김제	2016-09-28	650.0	98.9	109.4	36.7	14.3	0.177	배
1	김제	2016-08-30	530.4	86.7	104.1	38.6	13.0	0.099	배
2	나주	2017-09-26	896.0	113.0	122.5	40.9	13.8	0.187	배
3	나주	2017-09-28	958.5	110.1	121.9	27.9	14.1	0.180	배
4	영천	2017-10-20	715.8	97.5	114.1	34.3	14.3	0.179	배

[복숭아 데이터 수집]

```
# 복숭아 자료
# 품종별 코드
years = ['2016', '2017', '2018', '2019', '2020', '2021']
kind_codes = ['peach01', 'peach02', 'peach03']
# API 요청 보내기
url = 'http://apis.data.go.kr/1390804/Nihhs_Fruit_Peach_GrwhInfo/peachAnalsDataList'
fruit_peach = pd .DataFrame ()  # 빈 데이터프레임
for year in years :
   for kind_code in kind_codes :
                                    params                                    =                                    {'serviceKey':
'rhKXU5xPur/LMP2M8j2kA/SY0OAo97wV4Br00ZJI92pJuhi5AsOBr1Xfc/HPASy3SJ+xlnzbgctADLwRg2LqNQ==',    'selyear':    year    ,
'spciesCode': kind_code }
     try :
       # 요청
       response = requests .get (url , params =params )
       # 응답 처리
       if response .status_code == 200 :
         xml_data =response .content
         # XML을 Python dictionary로 변환
         xml_dict = xmltodict .parse (xml_data )
         # Python dictionary를 JSON 문자열로 변환
         json_data = json .dumps (xml_dict , indent =2 )
         # JSON 데이터를 DataFrame으로 변환
         df = pd .json_normalize (xml_dict ['response']['body_Anals']['result_Anals'])
         df ['과일'] = '복숭아'
         # dfset3에 추가
         fruit_peach = pd .concat ([fruit_peach , df ], ignore_index =True )
       else :
         print (f 'Error occurred for year {year } and kind_code {kind_code }: {response .status_code }')
     except requests .exceptions .RequestException as e :
       print ('Request failed:', e )
     except Exception as e :
       print ('An error occurred:', e )
fruit_peach.to_csv ('./fruit_peach.csv') # csv 저장
fruit_peach.head ()
```

	farmName	examinDatetm	avgFrutWt	avgLgdiamtr	avgPrbrd	avgHrdnss	avgHrdnss2	avgHrdnss3	avgBrx	avgPrcn	avgCrdValue	avgCrd2Value	avgCrd3Value	과일
0	김제	2016-09-12	354.8	79.1	88.5	36.1	35.0	35.5	12.5	0.26	65.0	11.0	42.3	복숭아
1	김제	2016-09-08	352.8	78.7	88.8	43.1	47.5	45.3	12.9	0.25	64.3	10.7	42.5	복숭아
2	김제	2017-08-16	421.3	85.3	95.5	37.8	39.8	38.8	11.9	0.32	62.5	13.8	26.3	복숭아
3	이천	2017-08-17	386.5	76.1	93.5	16.7	17.9	17.3	12.7	0.36	69.9	4.3	28.6	복숭아
4	청도	2017-08-16	256.6	71.8	80.1	26.2	23.7	24.9	12.7	0.21	57.8	21.8	26.2	복숭아

[포도 데이터 수집]

```
# 포도 자료
# 품종별 코드
years = ['2016', '2017', '2018', '2019', '2020', '2021']
kind_codes = ['grape01', 'grape02', 'grape03', 'grape04']
# API 요청 보내기
url = 'http://apis.data.go.kr/1390804/Nihhs_Fruit_Grape_GrwhInfo/grapeAnalsDataList'
fruit_grape = pd .DataFrame ()  # 빈 데이터프레임
for year in years :
   for kind_code in kind_codes :
     params = {'serviceKey':
'rhKXU5xPur/LMP2M8j2kA/SY0OAo97wV4Br00ZJI92pJuhi5AsOBr1Xfc/HPASy3SJ+xlnzbgctADLwRg2LqNQ==', 'selyear': year ,
'spciesCode': kind_code }
     try :
       # 요청
       response = requests .get (url , params =params )
       # 응답 처리
       if response .status_code == 200 :
         xml_data =response .content
         # XML을 Python dictionary로 변환
         xml_dict = xmltodict .parse (xml_data )
         # Python dictionary를 JSON 문자열로 변환
         json_data = json .dumps (xml_dict , indent =2 )
         # JSON 데이터를 DataFrame으로 변환
         df = pd .json_normalize (xml_dict ['response']['body_Anals']['result_Anals'])
         df ['과일'] = '포도'
         # dfset3에 추가
         fruit_grape = pd .concat ([fruit_grape , df ], ignore_index =True )
       else :
         print (f 'Error occurred for year {year } and kind_code {kind_code }: {response .status_code }')
     except requests .exceptions .RequestException as e :
       print ('Request failed:', e )
     except Exception as e :
       print (f 'An error occurred for year {year } and kind_code {kind_code }: {e }')
fruit_grape.to_csv ('./fruit_grape.csv') # csv 저장
fruit_grape.head ()
```

	farmName	examinDatetm	avgFrutWt	avgFlshWgh	avgLgdiamtr	avgPrbrd	avgColorValue	avgBrx	avgPrcn	avgCrdValue	avgCrd2Value	avgCrd3Value	avgAnthoValue	과일
0	천안	2016-09-28	521.57	13.05	30.42	25.91	8.67	20.73	0.381	25.56	6.17	-0.41	0.0700793	포도
1	김제	2016-08-22	447.00	5.62	22.44	20.83	9.00	15.56	0.558	27.13	3.01	-2.13	0.2385955	포도
2	천안	2017-10-10	563.65	11.04	28.44	24.58	8.35	21.19	0.395	23.81	5.98	-0.54	0.0752715	포도
3	김제	2017-08-29	379.36	6.07	21.88	20.72	9.83	14.50	0.534	27.20	2.38	-2.20	0.1848284	포도
4	화성	2017-09-07	371.31	6.54	23.70	21.55	9.67	17.06	0.468	23.76	2.81	-1.26	0.3211184	포도

[감귤 데이터 수집]

```
# 감귤 자료
# 품종별 코드
years = ['2016', '2017', '2018', '2019', '2020', '2021']
farm_codes = ['01', '02', '04', '06', '07', '09', '12', '14', '16', '26', '29', '30', '31', '32']
#01:용흥, 02:성산, 04:토산, 06:아라, 07:무릉, 09:하원, 12:덕천, 14:신촌, 16:금악, 26:하례, 29:창천, 30:덕수, 31:신효, 32:신흥
# API 요청 보내기
url = 'http://apis.data.go.kr/1390804/Nihhs_Fruit_Citrus_GrwhInfo/citrusFrutfrutAnals'
fruit_gul = pd .DataFrame ()  # 빈 데이터프레임
for year in years :
   for farm_code in farm_codes :
     params = {
       'serviceKey': 'rhKXU5xPur/LMP2M8j2kA/SY0OAo97wV4Br00ZJI92pJuhi5AsOBr1Xfc/HPASy3SJ+xlnzbgctADLwRg2LqNQ==',
       'selyear': year , 'farmCode' : farm_code , 'numOfRows' : '10', 'pageNo' : '1'
     }
     try :
       # 요청
       response = requests .get (url , params =params )
       # 응답 처리
       if response .status_code == 200 :
         xml_data = response .content
         # XML을 Python dictionary로 변환
         xml_dict = xmltodict .parse (xml_data )
         # 'Response.Body.Model' 안에 'Model' 키가 있는 경우
         if 'Model'in xml_dict ['Response']['Body']:
           data_list = xml_dict ['Response']['Body']['Model']
           # 'Model' 키 안에 여러 개의 데이터가 리스트 형태로 들어가 있으므로 각각 처리
           for data in data_list :
             # 각 데이터를 JSON 문자열로 변환
             json_data = json .dumps ({'Model': data }, indent =2 )
             # JSON 데이터를 DataFrame으로 변환
             df = pd .json_normalize (json .loads (json_data ))
             # 과일 정보 추가
             df ['과일'] = '감귤'
             # df를 fruit_gul에 추가
             fruit_gul = pd .concat ([fruit_gul , df ], ignore_index =True )
         else :
           print (f "No 'Model' key in the response for year {year } and farm_code {farm_code }")
       else :
         print (f 'Error occurred for year {year } and farm_code {farm_code }: {response .status_code }')
     except requests .exceptions .RequestException as e :
       print ('Request failed:', e )
     except Exception as e :
       print ('An error occurred:', e )
fruit_gul .to_csv ('./fruit_gul.csv') # csv 저장
fruit_gul .head ()
```

	Model.farm_code	Model.farm_name	Model.examin_datetm	Model.avg_prbrd	Model.avg_lgdiamtr	Model.avg_frut_wt	Model.avg_fish_wgh	Model.avg_fish_rate	Model.avg_frsk_one	Model.avg_brx	Model.avg_prcn	Model.avg_brx_prcn_rate	과일
0	01	용흥	2016-08-01	37.5	33.8	24.8	16.6	66.7	2.6	8.5	3.6	2.4	감귤
1	01	용흥	2016-08-16	41.3	36.7	33.2	24.2	72.9	2.4	9.2	3.2	2.9	감귤
2	01	용흥	2016-09-01	46.5	40.2	46.2	35.9	77.6	2.3	8.9	2.4	3.9	감귤
3	01	용흥	2016-09-20	53.1	44.4	68.2	56.2	82.2	1.8	8.7	1.5	5.7	감귤
4	01	용흥	2016-10-04	55.9	45.4	77.8	65.3	83.9	1.7	8.9	0.9	9.5	감귤

※ 데이터 정제

[배 데이터 정제]

```
# 배 데이터 살피기
df_pear = pd .read_csv ('./fruit_pear.csv', encoding ='utf-8', header =0 , index_col =0 )
print (df_pear .shape )
print (df_pear .columns ) # 농장명, 조사일, 과중 평균, 종경 평균, 횡경 평균, 경도 평균, 당도 평균, 산도 평균
```

```
(56, 9)
Index(['farm_name', 'examin_datetm', 'avg_frut_wt', 'avg_lgdiamtr', 'avg_prbrd', 'avg_hrdnss', 'avg_brx',
'avg_prcn', '과일'], dtype='object')
```

```
# 배 불필요 데이터 제거
df_pear = df_pear [['examin_datetm', 'avg_lgdiamtr',  'avg_prbrd', 'avg_brx', 'avg_prcn', '과일']]
# 컬럼명 변경
print (df_pear .isnull ().sum ())
df_pear = df_pear .rename (columns ={'examin_datetm':'조사일', 'avg_lgdiamtr':'종경',
'avg_prbrd':'횡경','avg_brx':'당도', 'avg_prcn':'산도'})
df_pear .head ()
```

```
examin_datetm    0
avg_lgdiamtr     0
avg_prbrd        0
avg_brx          0
avg_prcn         0
과일               0
dtype: int64
```

	조사일	종경	횡경	당도	산도	과일
0	2016-09-28	98.9	109.4	14.3	0.177	배
1	2016-08-30	86.7	104.1	13.0	0.099	배
2	2017-09-26	113.0	122.5	13.8	0.187	배
3	2017-09-28	110.1	121.9	14.1	0.180	배
4	2017-10-20	97.5	114.1	14.3	0.179	배

[복숭아 데이터 정제]

```
# 복숭아 데이터 살피기
df_peach = pd .read_csv ('./fruit_peach.csv', encoding ='utf-8', header =0 , index_col =0 )
print (df_peach .shape )
print (df_peach .columns ) # 농장명, 조사일, 과중 평균, 종경 평균, 횡경 평균, 경도1, 경도2, 경도 평균, 당도 평균,
산도 평균, 착색L, 착색a, 착색 b
```

```
(46, 14)
Index(['farmName', 'examinDatetm', 'avgFrutWt', 'avgLgdiamtr', 'avgPrbrd', 'avgHrdnss', 'avgHrdnss2', 'avgHrdnss3',
'avgBrx', 'avgPrcn', 'avgCrdValue', 'avgCrd2Value', 'avgCrd3Value', '과일'], dtype='object')
```

```
# 복숭아 불필요 데이터 제거
df_peach = df_peach [['examinDatetm', 'avgLgdiamtr', 'avgPrbrd', 'avgBrx', 'avgPrcn', '과일']]
# 컬럼명 변경
print (df_peach .isnull ().sum ())
df_peach = df_peach .rename (columns ={'examinDatetm':'조사일','avgLgdiamtr':'종경',
'avgPrbrd':'횡경','avgBrx':'당도', 'avgPrcn':'산도'})
df_peach .head ()
```

```
examinDatetm    0
avgLgdiamtr     0
avgPrbrd        0
avgBrx          0
avgPrcn         0
과일              0
dtype: int64
```

	조사일	종경	횡경	당도	산도	과일
0	2016-09-12	79.1	88.5	12.5	0.26	복숭아
1	2016-09-08	78.7	88.8	12.9	0.25	복숭아
2	2017-08-16	85.3	95.5	11.9	0.32	복숭아
3	2017-08-17	76.1	93.5	12.7	0.36	복숭아
4	2017-08-16	71.8	80.1	12.7	0.21	복숭아

[포도 데이터 정제]

```
# 포도 데이터 살피기
df_grape = pd .read_csv ('./fruit_grape.csv', encoding ='utf-8', header =0 , index_col =0 )
print (df_grape .shape )
print (df_grape .columns ) #농장명, 조사일자, 과방중, 과립중, 종경, 횡경, 칼라차트, 당도, 산도, 색차계(L),
색차계(a), 색차계(b), 안토시아닌
```

```
(38, 14)
Index(['farmName', 'examinDatetm', 'avgFrutWt', 'avgFlshWgh', 'avgLgdiamtr', 'avgPrbrd', 'avgColorValue', 'avgBrx',
'avgPrcn', 'avgCrdValue', 'avgCrd2Value', 'avgCrd3Value', 'avgAnthoValue', '과일'], dtype='object')
```

```
# 포도 불필요 데이터 제거
df_grape = df_grape [['examinDatetm', 'avgLgdiamtr', 'avgPrbrd', 'avgBrx', 'avgPrcn', '과일']]
# 컬렴명 변경
print (df_grape .isnull ().sum ())
df_grape = df_grape .rename (columns ={'examinDatetm':'조사일', 'avgLgdiamtr':'종경',
'avgPrbrd':'횡경','avgBrx':'당도', 'avgPrcn':'산도'})
df_grape .head ()
```

```
examinDatetm    0
avgLgdiamtr     0
avgPrbrd        0
avgBrx          0
avgPrcn         0
과일              0
dtype: int64
```

	조사일	종경	횡경	당도	산도	과일
0	2016-09-28	30.42	25.91	20.73	0.381	포도
1	2016-08-22	22.44	20.83	15.56	0.558	포도
2	2017-10-10	28.44	24.58	21.19	0.395	포도
3	2017-08-29	21.88	20.72	14.50	0.534	포도
4	2017-09-07	23.70	21.55	17.06	0.468	포도

[감귤 데이터 정제]

```
# 감귤 데이터 살피기
df_gul = pd .read_csv ('./fruit_gul.csv', encoding ='utf-8', header =0 , index_col =0 )
print (df_gul .shape )
print (df_gul .columns ) #농장코드, 농장명, 조사일자, 횡경, 종경, 과중, 과육종, 과육율, 과피, 당도, 산도, 당산비
```

```
(706, 13)
Index(['Model.farm_code', 'Model.farm_name', 'Model.examin_datetm',
       'Model.avg_prbrd', 'Model.avg_lgdiamtr', 'Model.avg_frut_wt',
       'Model.avg_flsh_wgh', 'Model.avg_flsh_rate', 'Model.avg_frsk_one',
       'Model.avg_brx', 'Model.avg_prcn', 'Model.avg_brx_prcn_rate', '과일'],
      dtype='object')
```

```
# 감귤 불필요 데이터 제거
df_gul = df_gul [['Model.examin_datetm', 'Model.avg_lgdiamtr', 'Model.avg_prbrd', 'Model.avg_brx',
'Model.avg_prcn', '과일']]
# 컬렴명 변경
print (df_gul .isnull ().sum ())
df_gul = df_gul .rename (columns ={'Model.examin_datetm':'조사일', 'Model.avg_lgdiamtr':'종경',
'Model.avg_prbrd':'횡경','Model.avg_brx':'당도', 'Model.avg_prcn':'산도'})
df_gul .head ()
```

```
Model.examin_datetm    0
Model.avg_lgdiamtr     3
Model.avg_prbrd        3
Model.avg_brx          3
Model.avg_prcn         3
과일                     0
dtype: int64
```

	조사일	종경	횡경	당도	산도	과일
0	2016-08-01	33.8	37.5	8.5	3.6	감귤
1	2016-08-16	36.7	41.3	9.2	3.2	감귤
2	2016-09-01	40.2	46.5	8.9	2.4	감귤
3	2016-09-20	44.4	53.1	8.7	1.5	감귤
4	2016-10-04	45.4	55.9	8.9	0.9	감귤

[데이터 합치기]

```
# 데이터프레임을 리스트로 저장
dfs = [df_pear , df_peach , df_grape , df_gul ]
# 데이터프레임 합치기
fruit_data_df = pd .concat (dfs , ignore_index =True )
fruit_data_df .head ()
```

	조사일	종경	횡경	당도	산도	과일
0	2016-09-28	98.9	109.4	14.3	0.177	배
1	2016-08-30	86.7	104.1	13.0	0.099	배
2	2017-09-26	113.0	122.5	13.8	0.187	배
3	2017-09-28	110.1	121.9	14.1	0.180	배
4	2017-10-20	97.5	114.1	14.3	0.179	배

[조사일 데이터 정제]

```
# 조사일 년도만 추출
fruit_data_df ['조사일'] = fruit_data_df ['조사일'].str.split('-').str[0 ]
fruit_data_df
```

	조사일	종경	횡경	당도	산도	과일
0	2016	98.9	109.4	14.3	0.177	배
1	2016	86.7	104.1	13.0	0.099	배
2	2017	113.0	122.5	13.8	0.187	배
3	2017	110.1	121.9	14.1	0.180	배
4	2017	97.5	114.1	14.3	0.179	배
...	...	...	...	...	...	...
838	2021	49.5	58.8	7.2	1.000	감귤
839	2021	51.9	59.7	8.0	0.800	감귤
840	2021	52.0	63.5	8.7	0.800	감귤
841	2021	53.2	62.0	8.7	0.800	감귤
842	2021	51.3	63.5	9.8	0.800	감귤

[컬럼명 변경]

```
# 데이터 타입 확인
print (fruit_data_df .dtypes )
# 데이터 타입 변경
fruit_data_df = fruit_data_df .astype ({'조사일':'int'})
# 컬럼명 변경
fruit_data_df = fruit_data_df .rename (columns ={'조사일':'연도'})
# 데이터 타입 확인
print (fruit_data_df .dtypes )
```

```
조사일     object
종경      float64
횡경      float64
당도      float64
산도      float64
과일       object
dtype: object
연도       int32
종경     float64
횡경     float64
당도     float64
산도     float64
과일      object
dtype: object
```

[당도, 산도, 횡경, 종경 관계 비교를 위한 데이터 프레임 생성]

```
fruit_data_df_copy = fruit_data_df .copy ()
fruit_data_df_copy = fruit_data_df_copy .drop ('연도', axis =1 )
fruit_data_df_copy .head ()
```

	종경	횡경	당도	산도	과일
0	98.9	109.4	14.3	0.177	배
1	86.7	104.1	13.0	0.099	배
2	113.0	122.5	13.8	0.187	배
3	110.1	121.9	14.1	0.180	배
4	97.5	114.1	14.3	0.179	배

```
from sklearn .preprocessing import MinMaxScaler

# MinMaxScaler 초기화
scaler = MinMaxScaler ()

# '종경', '횡경', '당도', '산도'를 0에서 1 사이로 정규화
fruit_data_df_copy [['종경', '횡경', '당도', '산도']] = scaler .fit_transform (fruit_data_df_copy [['종경',
'횡경', '당도', '산도']])

print (fruit_data_df_copy .shape )
fruit_data_df_copy
```

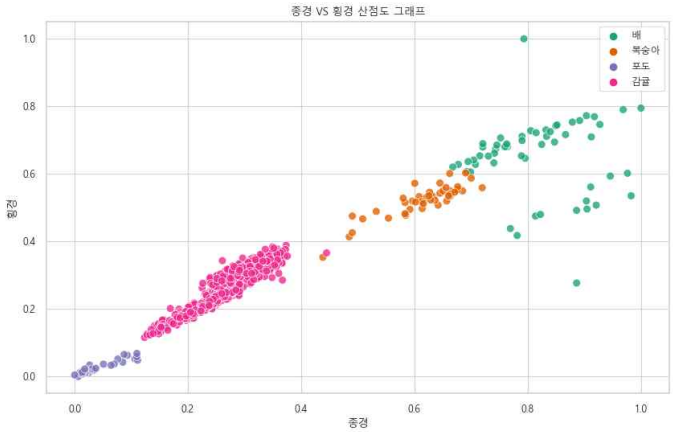
(843, 5)

	종경	횡경	당도	산도	과일
0	0.847419	0.694176	0.566394	0.038478	배
1	0.715399	0.653450	0.484581	0.021522	배
2	1.000000	0.794836	0.534928	0.040652	배
3	0.968618	0.790226	0.553807	0.039130	배
4	0.832269	0.730290	0.566394	0.038913	배
...	...	...	...	...	...
838	0.312845	0.305363	0.119572	0.217391	감귤
839	0.338816	0.312279	0.169918	0.173913	감귤
840	0.339898	0.341478	0.213971	0.173913	감귤
841	0.352884	0.329952	0.213971	0.173913	감귤
842	0.332323	0.341478	0.283197	0.173913	감귤

※ 데이터 분석

[종경VS횡경 산점도 표현]

```
# 산점도 초기화
plt .figure (figsize =(10 , 6 ))
# 각 과일에 대한 산점도 그리기
sns .set(style ='whitegrid', font ='Malgun Gothic',
font_scale =.8 )
sns .scatterplot(x ='종경', y ='횡경', hue ='과일', data
=fruit_data_df_copy , palette ='Dark2', s =50 , alpha
=0.8 )
# 축 라벨링과 제목
plt .xlabel ('종경')
plt .ylabel ('횡경')
plt .title ('종경 VS 횡경 산점도 그래프')
# 범례 표시
plt .legend ()
# 그림 표시
plt .show ()
```

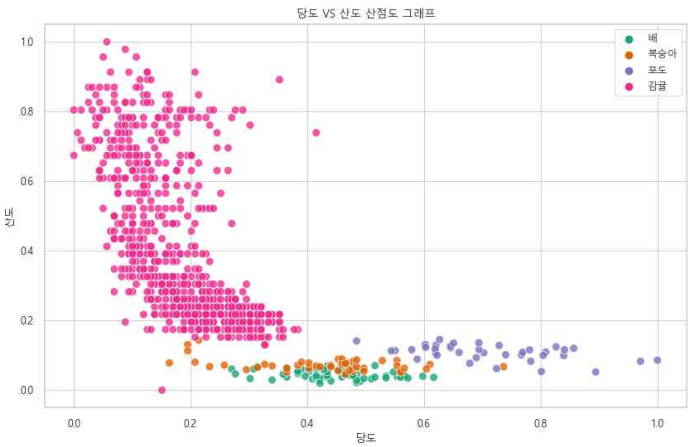


결 과

종경과 횡경이 비례하는 분포를 보였으며, 배 > 복숭아 > 감귤 > 포도 순으로 종횡 비율이 큰 것으로 보여졌다.

[당도VS산도 산점도 표현]

```
# 산점도 초기화
plt .figure (figsize =(10 , 6 ))
# 각 과일에 대한 산점도 그리기
sns .set(style ='whitegrid', font ='Malgun Gothic',
font_scale =.8 )
sns .scatterplot(x ='당도', y ='산도', hue ='과일', data
=fruit_data_df_copy , palette ='Dark2', s =50 , alpha
=0.8 )
# 축 라벨링과 제목
plt .xlabel ('당도')
plt .ylabel ('산도')
plt .title ('당도 VS 산도 산점도 그래프')
# 범례 표시
plt .legend ()
# 그림 표시
plt .show ()
```

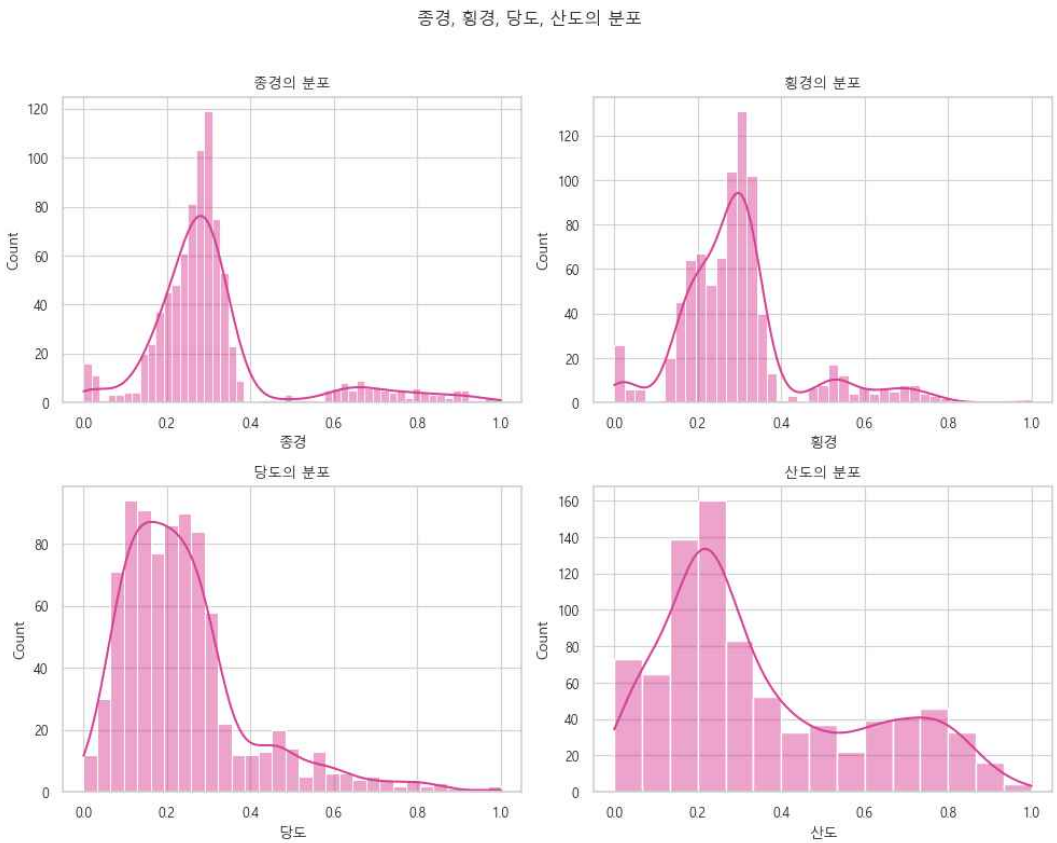


결 과

감귤의 산도가 대체로 높은 분포를 보이며 다른 과일들에 비해 당도가 낮았다. 배, 복숭아, 포도는 산도는 낮고 당도가 높은편이다. 가장 높은 당도를 보인 과일은 포도이다.

[각 열데이터의 분포 확인]

```
# 서브플롯 초기화
fig , axes = plt .subplots (2 , 2 , figsize =(10 , 8 ))
sns .set(style ='whitegrid', font ='Malgun Gothic', font_scale =.8 , palette ='PiYG')
# '종경'의 분포
sns .histplot(fruit_data_df_copy ['종경'], kde =True , ax =axes [0 , 0 ])
axes [0 , 0 ].set_title('종경의 분포')
# '횡경'의 분포
sns .histplot(fruit_data_df_copy ['횡경'], kde =True , ax =axes [0 , 1 ])
axes [0 , 1 ].set_title('횡경의 분포')
# '당도'의 분포
sns .histplot(fruit_data_df_copy ['당도'], kde =True , ax =axes [1 , 0 ])
axes [1 , 0 ].set_title('당도의 분포')
# '산도'의 분포
sns .histplot(fruit_data_df_copy ['산도'], kde =True , ax =axes [1 , 1 ])
axes [1 , 1 ].set_title('산도의 분포')
# 전체 그림의 제목
fig .suptitle ('종경, 횡경, 당도, 산도의 분포')
# 레이아웃 조정
plt .tight_layout (rect =[0 , 0 , 1 , 0.96 ])
# 그림 표시
plt .show ()
```



결 과

종경, 횡경은 0.0~0.4 사이에 많이 분포하고 있으며, 두 비율이 거의 유사한 것을 볼 수 있다.
당도, 산도 또한 0.0~0.4 사이에 가장 많이 분포하고 있다.

[각 열데이터의 상관계수 확인]

```
#상관 계수 히트맵
sns .set(font ='Malgun Gothic', font_scale =1 )
sns .heatmap(data =fruit_data_df_copy .corr (), annot =True , cmap ='RdPu',square =True , cbar =True )
plt .show ()
```

결 과

횡경과 종경은 0.95의 높은 수치를 보이며 밀접한 관계를 보이고 있으며, 산도는 당도, 횡경, 종경 모두와 음수의 결과로 관련이 없음을 보여주고 있다.

※ 분류 알고리즘

[학습-테스트 데이터셋 분할]

```
import pandas as pd
from sklearn .metrics import accuracy_score , classification_report
from sklearn .preprocessing import LabelEncoder
from sklearn .model_selection import train_test_split , cross_val_score
# LabelEncoder 초기화
label_encoder = LabelEncoder ()
# 데이터와 정답 분류
X = fruit_data_df_copy [['종경', '횡경', '당도', '산도']]
y = label_encoder .fit_transform (fruit_data_df_copy ['과일']) # '과일' 열을 숫자로 변환
# 데이터를 학습 및 테스트셋으로 나누기
X_train , X_test , y_train , y_test = train_test_split (X , y , test_size =0.2 , random_state =42 )
```

[knn 분류 알고리즘]

```
from sklearn .ensemble import RandomForestClassifier
# 랜덤 포레스트 분류 모델 초기화
model = RandomForestClassifier ()
# k-겹 교차 검증 수행
cv_scores = cross_val_score (model , X , y , cv =5 )  # 5겹 교차 검증 예시
# 교차 검증 정확도 출력
print ("Cross-Validation Scores:", cv_scores )
```

```
print ("Mean Accuracy: {:.2f}".format (cv_scores .mean()))
# 모델 학습
model .fit (X_train , y_train )
# 테스트셋에 대한 예측
y_pred = model .predict (X_test )
print ('예측값 : ', y_pred )
print ('정답 : ', y_test )
# 정확도
accuracy = accuracy_score (y_test , y_pred )
print (f '테스트 정확도: {accuracy :.2f}')
# 분류 보고서
print (classification_report (y_test , y_pred ))
```

```
Cross-Validation Scores: [0.98224852 0.99408284 0.99408284 1.         0.98809524]
Mean Accuracy: 0.99
예측값 :  [0 0 0 0 2 0 0 2 0 2 0 0 0 0 0 0 0 0 0 0 3 2 0 0 0 0 0 0 0 0 0 3 0 0 3 0
 1 0 0 3 2 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 2 0 0 0 0 0 0 0 0
 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 3 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
 1 0 0 0 0 0 2 0 2 0 0 0 0 0 0 0 2 1 0 0 0 3 0 0 1 0 0 0 0 2 0 0 0 0 0 0 0
 0 0 0 0 0 0 1 0 0 0 2 0 0 0 0 1 0 0 0 0 0]
정답 :  [0 0 0 0 2 0 0 2 0 1 0 0 0 0 0 0 0 0 0 0 3 2 0 0 0 0 0 0 0 0 0 3 0 0 3 0
 1 0 0 3 2 0 0 0 0 0 0 1 0 0 0 2 0 0 0 0 0 0 0 2 0 0 0 0 2 0 0 0 0 0 0 0 0
 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 3 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
 1 0 0 0 0 0 2 0 2 0 0 0 0 0 0 0 2 1 0 0 0 3 0 0 1 0 0 0 0 2 0 0 0 0 0 0 0
 0 0 0 0 0 0 1 0 0 0 2 0 0 0 0 1 0 0 0 0 0]
테스트 정확도: 0.99
              precision    recall  f1-score   support

           0       0.99      1.00      1.00       137
           1       1.00      0.92      0.96        12
           2       0.93      0.93      0.93        14
           3       1.00      1.00      1.00         6

    accuracy                           0.99       169
   macro avg       0.98      0.96      0.97       169
weighted avg       0.99      0.99      0.99       169
```

[knn 분류 알고리즘]

```
from sklearn .linear_model import LogisticRegression
# Logistic Regression 모델 초기화
model = LogisticRegression ()
# 모델 학습
model .fit (X , y )
# 테스트셋에 대한 예측
y_pred = model .predict (X_test )
print ('예측값 : ', y_pred )
print ('정답 : ', y_test )
# 정확도
accuracy = accuracy_score (y_test , y_pred )
print (f '테스트 정확도: {accuracy :.2f}')
# 분류 보고서
print (classification_report (y_test , y_pred ))
```

```
예측값 :  [0 0 0 0 2 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 3 2 0 0 0 0 0 0 0 0 0 3 0 0 3 0
 1 0 0 3 2 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 2 0 0 0 0 0 0 0 0
 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 3 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 3 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0]
정답 :  [0 0 0 0 2 0 0 2 0 1 0 0 0 0 0 0 0 0 0 0 3 2 0 0 0 0 0 0 0 0 0 3 0 0 3 0
```

```
 1 0 0 3 2 0 0 0 0 0 0 1 0 0 0 2 0 0 0 0 0 0 0 2 0 0 0 0 2 0 0 0 0 0 0 0 0
 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 3 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
 1 0 0 0 0 0 2 0 2 0 0 0 0 0 0 0 2 1 0 0 0 3 0 0 1 0 0 0 0 2 0 0 0 0 0 0 0
 0 0 0 0 0 0 1 0 0 0 2 0 0 0 0 1 0 0 0 0 0]
테스트 정확도: 0.95
              precision    recall  f1-score   support

           0       0.97      1.00      0.99       137
           1       0.71      1.00      0.83        12
           2       1.00      0.36      0.53        14
           3       1.00      1.00      1.00         6

    accuracy                           0.95       169
   macro avg       0.92      0.84      0.83       169
weighted avg       0.96      0.95      0.94       169
```

결 과

로지스틱 회귀 알고리즘 보다 knn 분류 알고리즘을 사용했을 때 0.04정도 더 높은 정확도를 보였다.
두 분류 모두 높은 확률로 데이터를 분류하고 있다.

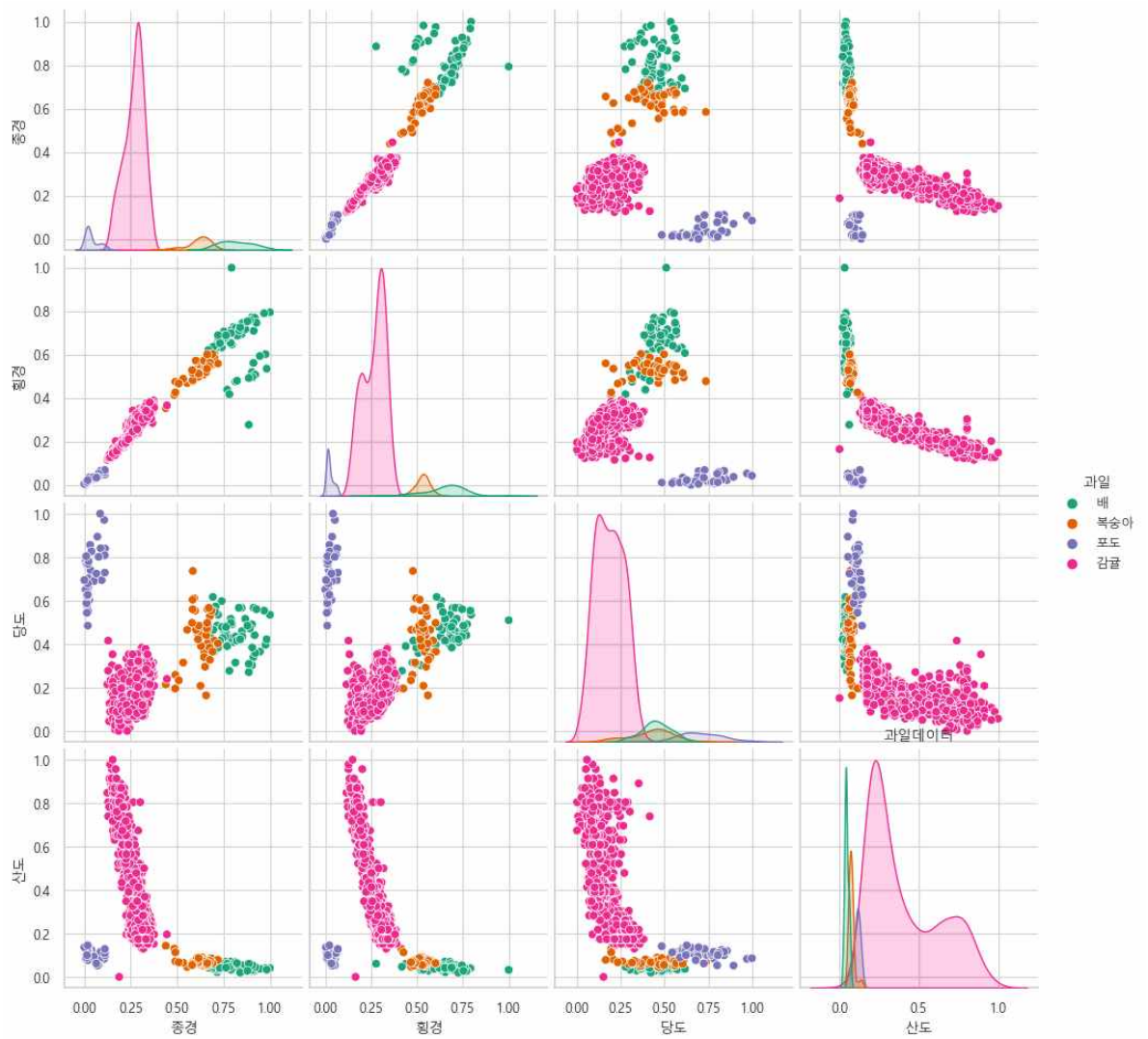
6. 과일 특성에 따른 선호도 분류

6.2 판매량으로 어떤 특징의 과일 선호도가 높은지 분석

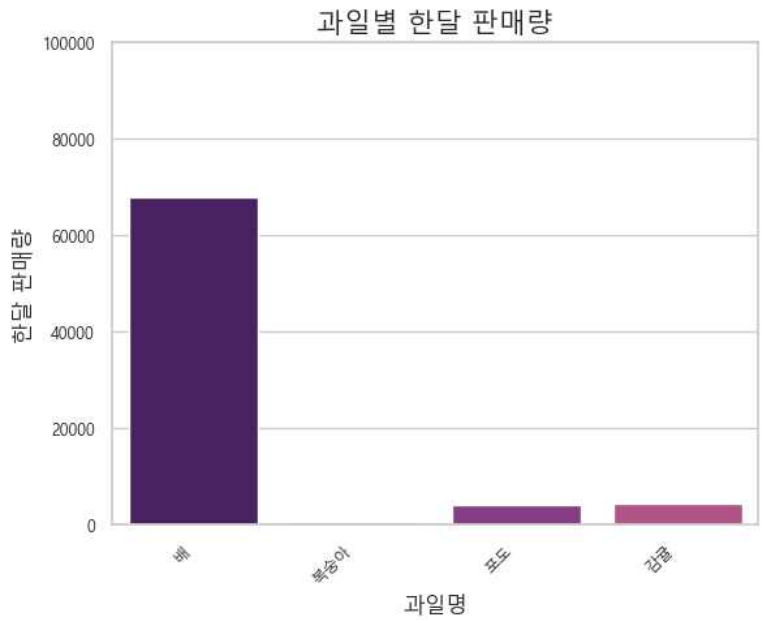
※ 데이터 시각화

[각 열데이터의 상관계수 확인]

```
# 과일 특징 시각화
sns .set(style ='whitegrid', font ='Malgun Gothic', font_scale =.8 , palette ='Dark2')
sns .pairplot(fruit_data_df_copy , hue ='과일', height =2.5 , diag_kind ='kde')
plt .title ('과일데이터')
plt .show ()
```




```
# 판매량 차트 그리기
colors = ['#f8c1a8', '#ef9198', '#e8608a', '#c0458a', '#8f3192', '#63218f', '#4b186c' ]
ax = sns .barplot(data =new_df , x =new_df .index, y ='한달판매량', palette =reversed (colors ))
ax .set_title('과일별 한달 판매량', fontsize =15 )
ax .set_xlabel('과일명', fontsize =12 )
ax .set_ylabel('한달 판매량', fontsize =12 )
ax .set_xticklabels(ax .get_xticklabels(), rotation =45 , ha ="right")  # x축 눈금 지정
ax .set_yticks([0 , 20000 , 40000 , 60000 , 80000 , 100000 ])  # y축 눈금 지정
plt .show ()
```



결 과

종횡비가 크고 당도가 높은 배의 판매량이 가장 많은 것으로 보였고, 두 번째로 크기가 큰 복숭아는 당도와 산도가 비슷한 배의 비에 판매량이 현격히 저조했다. 당도, 산도, 종경, 횡경은 판매량에 큰 영향을 주는 것으로 보이지 않았다.

6. 과일 특성에 따른 선호도 분류

6.3 소비량을 통한 수요 예측

※ 데이터 정제

[과일 정보 데이터 준비]

```
fruit_data_info = fruit_data_df .copy ()
fruit_data_info .head ()
```

	연도	종경	횡경	당도	산도	과일
0	2016	98.9	109.4	14.3	0.177	배
1	2016	86.7	104.1	13.0	0.099	배
2	2017	113.0	122.5	13.8	0.187	배
3	2017	110.1	121.9	14.1	0.180	배
4	2017	97.5	114.1	14.3	0.179	배

[소비량 데이터 준비]

```
totalData = dfset2 .drop (columns =['cnsmpCd', 'cnsmpMtralUpperNm', 'cnsmpLocalusgqty','cnsmpImportUsgqty',
'cnsmpLocalRelimp'])
totalData .columns =['연도', '과일', '총 소비량']
totalData .head ()
```

	연도	과일	총 소비량
0	2021	수박	265.0
1	2020	수박	351.0
2	2019	수박	297.0
3	2018	수박	9.0
4	2017	수박	30.0

[데이터 합치기]

```
# 연도 열의 데이터 타입을 int32로 변환
totalData ['연도'] = totalData ['연도'].astype ('int32')
fruit_data_info ['연도'] = fruit_data_info ['연도'].astype ('int32')

# 두 데이터 프레임을 병합
merged_df = pd .merge (fruit_data_info , totalData , on =['연도', '과일'], how ='inner')
merged_df
```

	연도	종경	횡경	당도	산도	과일	총 소비량
0	2016	98.9	109.4	14.3	0.177	배	11270.0
1	2016	86.7	104.1	13.0	0.099	배	11270.0
2	2017	113.0	122.5	13.8	0.187	배	10639.0
3	2017	110.1	121.9	14.1	0.180	배	10639.0
4	2017	97.5	114.1	14.3	0.179	배	10639.0
...	...	...	...	...	...	...	...
838	2021	49.5	58.8	7.2	1.000	감귤	137601.0
839	2021	51.9	59.7	8.0	0.800	감귤	137601.0
840	2021	52.0	63.5	8.7	0.800	감귤	137601.0
841	2021	53.2	62.0	8.7	0.800	감귤	137601.0
842	2021	51.3	63.5	9.8	0.800	감귤	137601.0

※ 회귀 알고리즘

[종경, 횡경에 따른 연도별 소비량 예측]

```
import pandas as pd
from sklearn .model_selection import train_test_split
from sklearn .linear_model import LinearRegression
from sklearn .metrics import mean_squared_error

# 특성과 타겟 선택
X = merged_df [['연도', '종경', '횡경']]
y = merged_df ['총 소비량']
# 데이터를 훈련 세트와 테스트 세트로 나눔
X_train , X_test , y_train , y_test = train_test_split (X , y , test_size =0.2 , random_state =42 )
# 선형 회귀 모델 생성 및 훈련
model = LinearRegression ()
model .fit (X_train , y_train )
# 테스트 데이터로 예측
y_pred = model .predict (X_test )
# 성능 평가 (평균 제곱근 오차)
rmse = mean_squared_error (y_test , y_pred , squared =False )
print (f '모델 성능 (RMSE): {rmse }')

# 테스트 데이터 첫 번째 샘플에 대한 예측과 실제 값 출력
print (f '실제: {y_test .iloc[0 ]}')
print (f '예측: {y_pred [0 ]}')
```

```
모델 성능 (RMSE): 36868.580293323335
실제: 141042.0
예측: 121454.93635364948
```

[당도, 산도에 따른 연도별 소비량 예측]

```
# 특성과 타겟 선택
X = merged_df [['연도', '당도', '산도']]
y = merged_df ['총 소비량']
# 데이터를 훈련 세트와 테스트 세트로 나눔
X_train , X_test , y_train , y_test = train_test_split (X , y , test_size =0.2 , random_state =42 )
# 선형 회귀 모델 생성 및 훈련
model = LinearRegression ()
model .fit (X_train , y_train )
# 테스트 데이터로 예측
y_pred = model .predict (X_test )
# 성능 평가 (평균 제곱근 오차)
rmse = mean_squared_error (y_test , y_pred , squared =False )
print (f '모델 성능 (RMSE): {rmse }')
# 테스트 데이터 첫 번째 샘플에 대한 예측과 실제 값 출력
print (f '실제: {y_test .iloc[0 ]}')
print (f '예측: {y_pred [0 ]}')
```

```
모델 성능 (RMSE): 33538.37955277624
실제: 141042.0
예측: 155409.34036672674
```

[종경, 횡경, 당도, 산도에 따른 연도별 소비량 예측]

```
import pandas as pd
from sklearn .model_selection import train_test_split
from sklearn .linear_model import LinearRegression
from sklearn .metrics import mean_squared_error
# 특성과 타겟 선택
X = merged_df [['연도', '종경', '횡경', '당도', '산도']]
y = merged_df ['총 소비량']
# 데이터를 훈련 세트와 테스트 세트로 나눔
X_train , X_test , y_train , y_test = train_test_split (X , y , test_size =0.2 , random_state =42 )
# 선형 회귀 모델 생성 및 훈련
model = LinearRegression ()
model .fit (X_train , y_train )
# 테스트 데이터로 예측
y_pred = model .predict (X_test )
# 성능 평가 (평균 제곱근 오차)
rmse = mean_squared_error (y_test , y_pred , squared =False )
print (f '모델 성능 (RMSE): {rmse }')
# 테스트 데이터 첫 번째 샘플에 대한 예측과 실제 값 출력
print (f '실제: {y_test .iloc[0 ]}')
print (f '예측: {y_pred [0 ]}')
```

```
모델 성능 (RMSE): 23922.472609792672
실제: 141042.0
예측: 148928.33698741673
```

결 과

당도, 산도 또는 종경, 횡경 일부 보다 모든 데이터를 특성으로 했을 때 모델의 성능이 가장 좋았다.

7. 블로그 데이터 클라우드로 보는 과일 트렌드 분석

7.1 네이버 블로그 API로 자료 수집

※ 데이터 수집 : 사용한 데이터 정보

출처 : NEVER Developers
자료명 : 블로그 검색 서비스 API

```
import os
import sys
import urllib .request
from datetime import datetime
import time
import json
client_id = 'rMIg8ujisG5V2gqM8LpI'
client_secret = 'XlkFhrMewH'
#[CODE 1]
def getRequestUrl (url ):
   req = urllib .request .Request (url )
   req .add_header ("X-Naver-Client-Id", client_id )
   req .add_header ("X-Naver-Client-Secret", client_secret )
   try :
     response = urllib .request .urlopen (req )
     if response .getcode() == 200 :
       print ("[%s ] Url Request Success" % datetime .now ())
       return response .read().decode('utf-8')
   except Exception as e :
     print (e )
     print ("[%s ] Error for URL : %s " % (datetime .now (), url ))
     return None
#[CODE 2]
def getNaverSearch (node , srcText , start , display ):
   base = "https://openapi.naver.com/v1/search"
   node = "/%s .json" % node
   parameters = "?query=%s &start=%s &display=%s " % (urllib .parse.quote(srcText ), start , display )
   url = base + node + parameters
   responseDecode = getRequestUrl (url )  #[CODE 1]
   if (responseDecode == None ):
     return None
   else :
     return json .loads (responseDecode )
#[CODE 3]
def getPostData (post , jsonResult , cnt ):
   title = post ['title']
   description = post ['description']
   link = post ['link']

   pDate = datetime .strptime (post ['postdate'], '%Y%m %d ')
   pDate = pDate .strftime ('%d %b %Y')
   jsonResult .append({'cnt':cnt , 'title':title , 'description': description , 'link': link ,  'pDate':pDate })
   return
#[CODE 0]
```

```
def main ():
    node = 'blog'  # 크롤링 할 대상
    srcText = input ('검색어를 입력하세요: ')
    cnt = 0
    jsonResult = []

    # 현재 날짜를 '년월일' 형식의 문자열로 가져오기
    current_date = datetime .now ().strftime ('%Y%m %d ')
    jsonResponse = getNaverSearch (node , srcText , 1 , 100 )  #[CODE 2]
    total = jsonResponse ['total']
    while ((jsonResponse != None ) and (jsonResponse ['display'] != 0 )):
      for post in jsonResponse ['items']:
        cnt += 1
        getPostData (post , jsonResult , cnt )  #[CODE 3]
      start = jsonResponse ['start'] + jsonResponse ['display']
      jsonResponse = getNaverSearch (node , srcText , start , 100 )  #[CODE 2]
    print ('전체 검색 : %d 건' %total )
    with open (current_date +'%s _naver_%s .json' % (srcText , node ), 'w', encoding ='utf8') as outfile :
      jsonFile = json .dumps (jsonResult ,  indent =4 , sort_keys =True ,  ensure_ascii =False )
      outfile .write (jsonFile )

    print ("가져온 데이터 : %d 건" %(cnt))
    print (current_date +'%s _naver_%s .json SAVED' % (srcText , node ))
if __name__ == '__main__':
    main ()
```

```
검색어를 입력하세요: 과일
[2023-11-27 18:42:48.587788] Url Request Success
[2023-11-27 18:42:48.829700] Url Request Success
[2023-11-27 18:42:49.092798] Url Request Success
[2023-11-27 18:42:49.378439] Url Request Success
[2023-11-27 18:42:49.651815] Url Request Success
[2023-11-27 18:42:49.973823] Url Request Success
[2023-11-27 18:42:50.265934] Url Request Success
[2023-11-27 18:42:50.536108] Url Request Success
[2023-11-27 18:42:50.866303] Url Request Success
[2023-11-27 18:42:51.163836] Url Request Success
HTTP Error 400: Bad Request
[2023-11-27 18:42:51.228748] Error for URL :
https://openapi.naver.com/v1/search/blog.json?query=%EA%B3%BC%EC%9D%BC&start=1001&display=100
전체 검색 : 14522660 건
가져온 데이터 : 1000 건
20231127과일_naver_blog.json SAVED
```

7. 블로그 데이터 클라우드로 보는 과일 트렌드 분석

7.2 데이터 클라우드 생성

※ 데이터 로드

```
import json
import re
from konlpy.tag import Okt
from collections import Counter
import matplotlib
import matplotlib .pyplot as plt
from matplotlib import font_manager , rc
from wordcloud import WordCloud

inputFileName = './20231127과일_naver_blog'
data = json .loads (open (inputFileName +'.json', 'r', encoding ='utf-8').read ())
print (data [:3 ])  # 처음 3개 항목만 출력
```

```
[{'cnt': 1, 'description': '인스타에서 봤던 성수 한정선인데 <b>과일</b> 찹쌀떡을 종류별로 판매하고 있는 거였다. 핫한 디저트는 못... 사실 요즘 여기저기서 <b>과일</b>모찌 이래가지고 좀 보기 싫었는데 떡이라고 한글로 써 놓으니까 흡족했다. 두... ', 'link': 'https://blog.naver.com/nowwegom/223268456335', 'pDate': '18 Nov 2023', 'title': '성수 한정선 쫀득쫀득한 <b>과일</b> 찹쌀떡 포장'}, {'cnt': 2, 'description': '포스팅은 <b>과일</b>멘솔 액상 4종 추천 리뷰를 해드리려고 해요 :-) 저는 5년 동안 전담을 사용 하는 베이퍼인데요 확실히 연초를 끊길 잘한 것 같아요 다양한 액상의 종류를 접했지만 제 취향엔 <b>과일</b> 멘솔 액상이 잘... ', 'link': 'https://blog.naver.com/cooldj79/223272689786', 'pDate': '23 Nov 2023', 'title': '<b>과일</b> 멘솔 액상 4종 추천 리뷰'}, {'cnt': 3, 'description': '건강을 위해 <b>과일</b>과 채소를 섭취하는 분들이 많습니다. 그런데, 혈당 때문에 문제가 생기는 당뇨의 경우에는 어떨까요? <b>과일</b>은 대표적으로 당이 많은 식품인데 당뇨에 좋은 <b>과일</b>이라는 게 다소 어불성설일 수... ', 'link': 'https://blog.naver.com/pamaba/223271581989', 'pDate': '22 Nov 2023', 'title': '당뇨에 좋은 <b>과일</b> 먹는 방법 간식까지 알아보기'}]
```

※ 데이터 정제

[불필요한 정보 제거(한글이 아닌 데이터 제거)]

```
message = ''
for item in data :
   if 'description'in item .keys():
     message = message + re .sub (r '[^a-zA-Z\s\w ]', ' ', item ['description']) +''
print (message [:150 ],'...') # 데이터 일부 확인
```

```
인스타에서 봤던 성수 한정선인데 b 과일 b 찹쌀떡을 종류별로 판매하고 있는 거였다 핫한 디저트는 못 사실 요즘 여기저기서 b 과일 b 모찌 이래가지고 좀 보기 싫었는데 떡이라고 한글로 써 놓으니까 흡족했다 두 포스팅은 b 과일 b 멘솔 액상 ...
```

[명사 추출]

```
tokens_ko = komo.nouns(message )
import nltk
ko = nltk .Text(tokens =tokens_ko )
print (len (ko ))
count_df_data = pd .DataFrame (tokens_ko )
count_df = pd .DataFrame (count_df_data .value_counts ())
count_df .columns =['빈도수']
count_df .head ()
```

```
20848
```

	빈도수
0	
과일	2547
을	246
선물	214
수	206
제철	160

[워드클라우드, 빈도수 그래프 함수 선언]

```
from PIL import Image

class Visualization :
   def __init__(self , wordList ):
     self .wordList = wordList
     self .wordDict = dict (wordList )  # list를 딕셔너리로 변경
     print (self .wordDict )
   def makeWordCloud (self ): # 워드 클라우드
     a_color_file = 'fruit.jpg'
     a_coloring = np .array(Image .open (a_color_file ))
     font_path = "C:/Windows/Fonts/malgun.ttf"
     wordcloud = WordCloud(font_path =font_path ,  mask =a_coloring , \
                 relative_scaling =0.3 , background_color ='lightyellow')
     wordcloud = wordcloud .generate_from_frequencies(self .wordDict )
     plt .imshow (wordcloud )
     plt .axis ('off')
     filename = 'myWordCloud.png'
     plt .figure (figsize =(9 , 9 ))
     plt .savefig (filename , dpi =700 , bbox_inches ='tight')
     print (filename + ' 파일이 저장되었습니다.')

   def makeBarChart (self ): # 막대 그래프 그리기
     barcount = 15  # 막대 개수 : 10개만 그리겠다.
     xlow , xhigh = - 0.5 , barcount - 0.5
     result = self .wordList [:barcount ]
     chartdata = []  # 차트 수치
     xdata = []  # 글씨
     mycolor = ['r', 'g', 'b', 'y', 'm', 'c', '#FFF0F0', '#CCFFBB', '#05CCFF', '#11CCFF']
     for idx in range (len (result )):
       chartdata .append (result [idx ][1 ])
       xdata .append (result [idx ][0 ])
       value = str (chartdata [idx ]) + '건'  # 예시 : 60건
       plt .text (x =idx , y =chartdata [idx ] - 5 , s =value , fontsize =8 , horizontalalignment ='center')
     plt .xticks (range (barcount ), xdata , rotation =45 )
     plt .bar (range (barcount ), chartdata , align ='center', color =mycolor )
     plt .title ('상위 ' + str (barcount ) + '빈도수')
     plt .xlim ([xlow , xhigh ])
     plt .xlabel ('주요 키워드')
     plt .ylabel ('빈도수')
     filename = 'myBarChart.png'
     plt .savefig (filename , dpi =400 , bbox_inches ='tight')
     print (filename + ' 파일이 저장되었습니다.')
```

[빈도수가 가장 높은 단어 500개를 추출하여 함수 호출]

```
data = ko .vocab().most_common(500 )
wordlist = list ()  # 튜플(단어, 빈도수)을 저장할 리스트
for word , count in data :
   # ccount는 빈도수를 의미하고, len(word)는 단어의 길이를 의미
   if (count >= 1 and len (word ) >= 2 ) :
     wordlist .append ((word , count ))
visual = Visualization (wordlist )
```

```
{'과일': 2547, '선물': 214, '제철': 160, '사과': 151, '아기': 123, '오늘': 114, '요즘': 96, '간식': 95, '추석': 88, '건강': 86, '채소': 79, '주문': 79, '샐러드': 79, '안녕하세요': 76, '카페': 73, '음식': 70, '추천': 68, '단감': 65, '주스': 65, '종류': 64, '세트': 61, '야채': 60, '딸기': 59, '바구니': 58, '강아지': 58, '개월': 58, '액상': 57, '소개': 52, '포장': 51, '세척': 50, '박스': 50, '이번': 50, '준비': 49, '제주': 49, '케이크': 48, '구매': 48, '생각': 47, '가게': 47, '아침': 47, '복숭아': 47, '디저트': 46, '망고': 46, '사용': 45, '가격': 45, '빙수': 45, '겨울': 44, '명절': 43, '대표': 41, '시기': 41, '시작': 41, '가지': 41, '포도': 41, '요리': 40, '여행': 40, '키위': 39, '맛집': 39, '젤리': 36, '가을': 36, '열대': 36, '아이': 35, '사라': 35, '크림': 35, '제품': 34, '시장': 34, '사진': 34, '이유식': 33, '바나나': 33, '감귤': 33, '계절': 33, '마트': 33, '때문': 32, '제가': 32, '후기': 32, '판매': 31, '샤인': 31, '비타민': 30, '아이들': 30, '여름': 30, '섭취': 29, '황금향': 29, '가족': 29, '찹쌀떡': 28, '모찌': 28, '마음': 28, '보관': 28, '머스캣': 27, '방법': 26, '재료': 25, '방문': 25, '10월': 25, '껍질': 25, '세제': 25, '사실': 24, '평소': 24, '가정': 24, '메뉴': 24, '위치': 24, '주방': 24, '프리미엄': 24, '사람': 24, '하루': 24, '처음': 23, '효능': 23, '시간': 23, '정도': 23, '과즙': 23, '커피': 23, '쥬스': 23, '고민': 23, '어린이': 22, '꾸러미': 22, '칼로리': 22, '도시락': 21, '매장': 21, '건조': 21, '오렌지': 21, '배달': 21, '참외': 21, '예전': 20, '농장': 20, '과자': 20, '수박': 20, '제주도': 20, '토마토': 20, '이나': 20, '변비': 20, '산지': 19, '보니': 19, '보육': 19, '신청': 19, '경기도': 19, '마요네즈': 19, '얼마': 19, '저택': 19, '푸드': 19, '친구': 19, '타르트': 18, '파인애플': 18, '이상': 18, '택배': 18, '구성': 18, '사랑': 18, '이용': 18, '이름': 18, '청주': 18, '음료': 17, '소스': 17, '당뇨': 16, '오후': 16, '엄마': 16, '성분': 16, '11월': 16, '쿠키': 16, '인기': 16, '김치': 16, '경우': 15, '식품': 15, '고급': 15, '마켓': 15, '레드': 15, '수확': 15, '라고': 15, '다이어트': 15, '블루베리': 15, '그중': 15, '오픈': 15, '치즈': 15, '수분': 15, '제일': 15, '주의': 14, '대부분': 14, '아이스크림': 14, '레시': 14, '지금': 14, '냉장고': 14, '구입': 14, '영양': 14, '샌드위치': 14, '지역': 14, '일반': 14, '골드': 14, '이유': 14, '자몽': 14, '창업': 14, '대만': 14, '고구마': 14, '상자': 14, '멜론': 14, '최고': 13, '상태': 13, '이다': 13, '옛날': 13, '남편': 13, '근처': 13, '당도': 13, '도착': 13, '장염': 13, '도움': 13, '부산': 13, '파파야': 13, '베트남': 13, '스팅': 12, '리뷰': 12, '활용': 12, '냉동': 12, '도마': 12, '날씨': 12, '지원': 12, '치킨': 12, '애월': 12, '전문점': 12, '선택': 12, '함유': 12, '무화과': 12, '모양': 12, '하이': 12, '정선': 11, '전문': 11, '최근': 11, '대전': 11, '랜덤': 11, '운영': 11, '식초': 11, '시즌': 11, '드레싱': 11, '자리': 11, '아삭': 11, '강정': 11, '입구': 11, '각종': 11, '피부': 11, '생일': 11, '시루': 11, '설탕': 11, '자두': 11, '중국': 11, '혈관': 11, '느낌': 10, '단맛': 10, '한라봉': 10, '재배': 10, '유기농': 10, '기본': 10, '온라인': 10, '다음': 10, '조금': 10, '리얼': 10, '보관법': 10, '필수': 10, '오랜만': 10, '자연': 10, '대신': 10, '현대': 10, '디자인': 10, '예약': 10, '이야기': 10, '진행': 10, '주소': 10, '확인': 10, '무인': 10, '견과': 10, '반찬': 10, '대구': 10, '식사': 10, '이랑': 10, '거리': 10, '경남': 10, '서울': 10, '다낭': 10, '맥주': 10, '감사': 10, '화채': 10, '동안': 9, '검색': 9, '임산부': 9, '네이버': 9, '한번': 9, '감기': 9, '만큼': 9, '레몬': 9, '육아': 9, '주말': 9, '재미': 9, '브랜드': 9, '깍두기': 9, '동결': 9, '인천': 9, '나트': 9, '우유': 9, '진열': 9, '정기': 9, '입맛': 9, '사장': 9, '주변': 9, '백화점': 9, '동물': 9, '걱정': 9, '8월': 9, '친환경': 9, '수능': 9, '하면': 9, '국산': 9, '분이': 9, '당일': 9, '오이': 9, '며칠': 9, '여지': 9, '주차': 9, '회사': 9, '사서': 9, '9월': 9, '손질': 9, '중간': 9, '감자': 9, '귀신': 8, '영양소': 8, '직송': 8, '대한민국': 8, '대회': 8, '농산물': 8, '아이스': 8, '결혼': 8, '사업': 8, '품질': 8, '국내': 8, '우리집': 8, '이예': 8, '동네': 8, '천혜': 8, '부담': 8, '대봉': 8, '생활': 8, '사항': 8, '초음파': 8, '성수동': 8, '출시': 8, '지도': 8, '키트': 8, '기분': 8, '초기': 8, '외출': 8, '상견례': 8, '지인': 8, '표현': 8, '홍시': 8, '동남아': 8, '믹스': 8, '만족': 8, '체리': 8, '도안': 8, '놀이': 8, '스타': 7, '취향': 7, '임신': 7, '태국': 7, '이것저것': 7, '등등': 7, '함량': 7, '타이': 7, '선발': 7, '봉지': 7, '먹거리': 7, '세계': 7, '생산': 7, '그런지': 7}
```

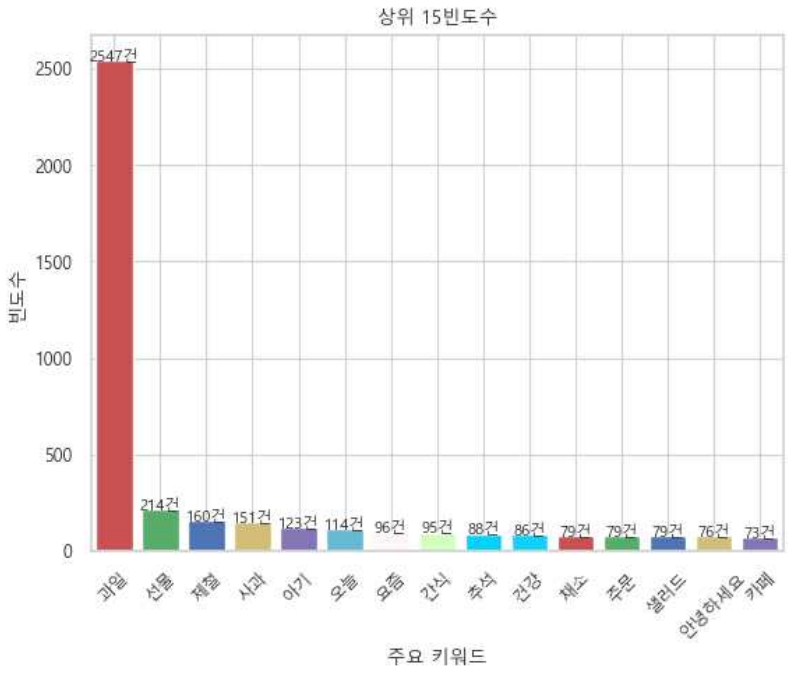
※ 데이터 시각화

[데이터 클라우드]

```
visual .makeWordCloud ()
```

[빈도수 그래프]

```
visual .makeBarChart ()
```



※ 데이터 정제

[불용어 제거]

```
# 불용어 제거
from konlpy.tag import Komoran
komo = Komoran()
tokens_ko2 = komo .nouns(message )
stop_word_file = 'stopword.txt'
stop_file = open (stop_word_file , 'rt', encoding ='utf-8')
stop_words = [ word .strip () for word in stop_file .readlines ()]
print ("불용어", stop_words )
tokens_ko2 = [each_word for each_word in tokens_ko2 if each_word not in stop_words ]

ko2 = nltk .Text(tokens =tokens_ko2 )
print (len (ko2 ))
```

불용어 ['안녕하세요', '종류', '소개', '강아지', '사용', '사라', '제가', '때문', '후기', '정도', '방문', '처음', '위치', '이나', '사실', '보니', '평소', '시간', '라고', '그중', '리뷰', '이다', '레시', '대부분', '이랑', '하이', '자리', '이상', '경우', '최근', '근처', '입구', '하면', '그런지', '분이', '동안', '원래', '각종', '이제', '사다', '해서', '지난번', '며칠', '오늘', '요즘', '이번', '조금', '다음', '오랜만', '진행', '나트', '주변', '사서', '중간', '이예', '등등', '오새', '올해']
19522

[명사 추출]

```
data = ko2 .vocab().most_common(500 )
wordlist2 = list ()  # 튜플(단어, 빈도수)을 저장할 리스트
for word , count in data :
   # count는 빈도수를 의미하고, len(word)는 단어의 길이를 의미합니다.
   if (count >= 1 and len (word ) >= 2 ) :
     wordlist2 .append ((word , count ))
visual2 = Visualization (wordlist2 )
```

{'과일': 2516, '선물': 265, '제철': 178, '사과': 158, '추석': 115, '아기': 112, '세트': 91, '간식': 86, '샐러드': 76, '음식': 75, '카페': 72, '주문': 71, '건강': 69, '추천': 66, '야채': 66, '액상': 66, '개월': 65, '주스': 64, '준비': 63, '채소': 62, '명절': 59, '가게': 57, '바구니': 57, '세척': 56, '빙수': 52, '복숭아': 52, '케이크': 52, '키위': 52, '단감': 51, '포도': 49, '디저트': 47, '딸기': 46, '제주': 46, '가격': 45, '시작': 45, '망고': 44, '포장': 44, '생각': 43, '아침': 43, '구매': 42, '가을': 42, '가지': 40, '여행': 39, '맛집': 37, '박스': 36, '요리': 36, '열대': 36, '바나나': 35, '여름': 35, '이유식': 34, '아이': 34, '사진': 34, '마트': 34, '시기': 31, '보관': 31, '계절': 31, '대표': 31, '세제': 31, '샤인': 31, '머스캣': 31, '10월': 31, '가족': 30, '아이들': 30, '크림': 30, '판매': 29, '시장': 29, '비타민': 29, '감귤': 29, '주방': 29, '젤리': 29, '마음': 29, '고민': 29, '오렌지': 28, '쥬스': 26, '제품': 26, '방법': 26, '프리미엄': 26, '황금향': 26, '모찌': 25, '겨울': 25, '재료': 25, '수박': 24, '농장': 24, '참외': 24, '변비': 24, '섭취': 23, '껍질': 23, '하루': 23, '찹쌀떡': 22, '메뉴': 22, '도시락': 22, '효능': 21, '제주도': 21, '커피': 21, '사람': 21, '도마': 21, '배달': 20, '이용': 20, '얼마': 20, '베트남': 19, '선택': 19, '친구': 19, '가정': 19, '수분': 19, '파인애플': 19, '블루베리': 19, '샌드위치': 19, '음료': 19, '택배': 19, '제일': 19, '매장': 18, '과즙': 18, '예전': 18, '골드': 18, '청주': 18, '상자': 18, '이름': 18, '냉동': 17, '성분': 17, '사랑': 17, '건조': 17, '마요네즈': 17, '칼로리': 17, '당도': 17, '구성': 17, '무화과': 17, '멜론': 17, '구입': 17, '일반': 16, '오픈': 16, '김치': 16, '토마토': 16, '전문점': 16, '꾸러미': 16, '날씨': 16, '푸드': 16, '수확': 16, '11월': 16, '지금': 15, '과자': 15, '도움': 15, '고급': 15, '소스': 15, '냉장고': 15, '함유': 15, '치즈': 15, '이유': 15, '대구': 15, '어린이': 15, '주의': 14, '남편': 14, '아삭': 14, '오후': 14, '영양': 14, '인기': 14, '타르트': 14, '보육': 14, '다이어트': 14, '유기농': 14, '활용': 13, '감기': 13, '초음파': 13, '부산': 13, '대만': 13, '마켓': 13, '최고': 13, '저택': 13, '스팅': 13, '파파야': 12, '엄마': 12, '생일': 12, '주소': 12, '확인': 12, '자몽': 12, '식사': 12, '9월': 12, '레드': 12, '모양': 12, '지인': 12, '도착': 12, '예약': 12, '도안': 12, '정선': 11, '영양소': 11, '창업': 11, '필수': 11, '옛날': 11, '상태': 11, '드레싱': 11, '산지': 11, '단맛': 11, '쿠키': 11, '기분': 11, '맥주': 11, '지역': 11, '시즌': 11, '설탕': 11, '수업': 11, '아이스크림': 11, '혈관': 11, '동남아': 11, '회사': 11, '체리': 11, '화채': 11, '감자': 11, '여의도': 11, '다낭': 10, '거리': 10, '키트': 10, '반찬': 10, '육아': 10, '유지': 10, '이야기': 10, '주차': 10, '중국': 10, '대신': 10, '출시': 10, '믹스': 10, '시루': 10, '기준': 10, '재배': 10, '자두': 10, '강정': 10, '손질': 10, '서울': 10, '함량': 10, '보관법': 10, '임신': 9, '백화점': 9, '기본': 9, '베리': 9, '식품': 9, '정기': 9, '관리': 9, '주말': 9, '이웃': 9, '이후': 9, '광주': 9, '8월': 9, '신청': 9, '경남': 9, '메로': 9, '수입': 9, '수능': 9, '표현': 9, '견과': 9, '감사': 9, '경기도': 9, '고구마': 9, '만족': 9, '아보카도': 9, '보충': 9, '놀이': 9, '리얼': 8, '초기': 8, '깍두기': 8, '무인': 8, '치킨': 8, '저녁': 8, '신제품': 8, '검색': 8, '유치원': 8, '친환경': 8, '성수동': 8, '네이버': 8, '냉장': 8, '진열': 8, '개인': 8, '추가': 8, '식기': 8, '숙성': 8, '다래': 8, '식초': 8, '농축': 8, '과육': 8, '생산': 8, '부담': 8, '장염': 8, '공구': 8, '시럽': 8, '미니': 8, '걱정': 7, '임산부':

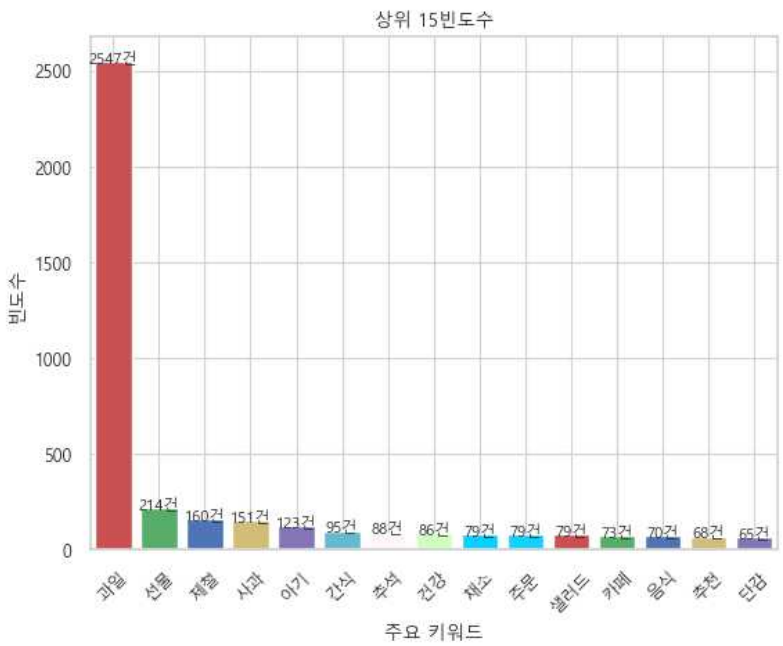
7, '혈당': 7, '인테리어': 7, '한라봉': 7, '급여': 7, '봉지': 7, '농협': 7, '두리안': 7, '입맛': 7, '이바지': 7, '당근': 7, '영업시간': 7, '일요일': 7, '대전': 7, '스토어': 7, '진심': 7, '축하': 7, '부탁': 7, '전문': 7, '레몬': 7, '당일': 7, '만큼': 7, '홍시': 7, '골목': 7, '상인': 7, '투어': 7, '식후': 7, '결혼': 7, '우리나라': 7, '자연': 7, '속초': 7, '우리집': 7, '하노이': 7, '시댁': 7, '동탄': 7, '열매': 7, '정리': 7, '유통': 7, '과정': 7, '추단': 7, '피부': 7, '동물': 7, '생활': 7, '의미': 7, '당뇨': 7, '호텔': 7, '청소': 7, '이마트': 7, '사장': 7, '발견': 7, '느낌': 7, '석류': 7, '스타': 6, '성수': 6, '천혜': 6, '태국': 6}

※ 데이터 시각화

[데이터 클라우드]

```
visual .makeWordCloud ()
```


```
visual2 .makeBarChart ()
```



결 과

과일 > 선물 > 제철 > 사과 > 아기 등의 빈도수를 보였다.

7. 블로그 데이터 클라우드로 보는 과일 트렌드 분석

7.3 최근 일주일간의 관심도 변화

※ 데이터 수집 : 사용한 데이터 정보

출처 : NEVER Developers
자료명 : 블로그 검색 서비스 API

```
import urllib .parse
import urllib .request
import matplotlib .pyplot as plt
from datetime import datetime , timedelta
import time
# 네이버 블로그 검색 결과 수집 함수
def get_result (client_id , client_secret , query , display =100 , start =1 , sort ='sim', max_page =5 ):
   result_list = []
   for page in range (1 , max_page + 1 ):
     enc_text = urllib .parse .quote (query )
     url = f "https://openapi.naver.com/v1/search/blog?query={enc_text }&display={display }&start={(page
-1 )*display + 1 }&sort={sort }"
     request = urllib .request .Request (url )
     request .add_header ("X-Naver-Client-Id", client_id )
     request .add_header ("X-Naver-Client-Secret", client_secret )
     try :
       response = urllib .request .urlopen (request )
       res_code = response .getcode()
       if res_code == 200 :
         response_body = response .read()
         response_json = json .loads (response_body )
         items = response_json .get('items', [])
         if not items :
           break  # No more items
         result_list .extend (items )
       else :
         print (f "Error Code: {res_code }")
         break
     except Exception as e :
       print (f "An error occurred: {e }")
       break
     time .sleep (0.1 )  # sleep 시켜서 요청 부하 방지
   return pd .DataFrame (result_list )
```

※ 데이터 정제

[텍스트 데이터에서 과일 언급 추출 후 일주일간의 빈도수 계산]

```
# 텍스트 데이터 분석 함수
def analyze_fruit_mentions (client_id , client_secret , queries , fruits , display =100 , max_page =5 ):
   # 날짜에 대한 각 과일의 빈도를 저장할 데이터 프레임
   date_fruit_df = pd .DataFrame (columns =['date'] + fruits )
   # 일주일로 기간 설정
   end_date = datetime .now () - timedelta (days =1 ) # 하루 전
   start_date = end_date - timedelta (days =7 )

   for query in queries :
     result_df = get_result (client_id , client_secret , query , display =display , max_page =max_page )

     for index , row in result_df .iterrows ():
       # 'postdate' 필드에서 날짜 추출
       date_str = row .get ('postdate', '')
       date = datetime .strptime (date_str , '%Y%m %d ')
       # 일주일간의 데이터 수집
```

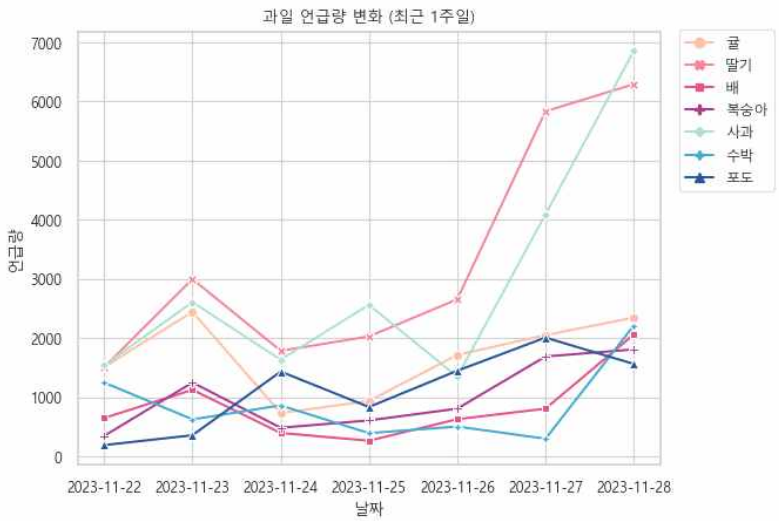
```
        if start_date <= date <= end_date :
          # 데이터 프레임이 존재하지 않을경우 초기화
          if date not in date_fruit_df ['date'].tolist():
            new_row = pd .DataFrame ([[date .strftime ('%Y-%m-%d ')] + [0 ] * len (fruits )], columns =['date'] +
fruits )
            date_fruit_df = pd .concat ([date_fruit_df , new_row ], ignore_index =True )
          # description' 필드의 각 과일에 대한 언급을 계산
          description = row .get ('description', '').lower ()
          for fruit in fruits :
            count = description .count (fruit .lower())
            date_fruit_df .loc [date_fruit_df ['date'] == date .strftime ('%Y-%m-%d '), fruit ] += count
    # 날짜별 각 과일의 빈도를 합산
    grouped_data = date_fruit_df .groupby ('date')[fruits ].sum()
    return grouped_data

query = ['과일', '귤', '딸기', '복숭아', '사과', '수박', '포도', '제철']
fruits = ['귤', '딸기', '배', '복숭아', '사과', '수박', '포도']
grouped_data = analyze_fruit_mentions (client_id , client_secret , query , fruits , max_page =5 )
grouped_data
```

date	귤	딸기	배	복숭아	사과	수박	포도
2023-11-20	890	495	394	357	785	503	703
2023-11-21	1406	2994	1256	620	2750	713	1094
2023-11-22	1517	1974	936	441	1986	1363	375
2023-11-23	2634	4043	1302	1392	3244	697	574
2023-11-24	1087	2896	495	480	2052	852	2372
2023-11-25	1418	4792	359	803	4949	554	1458
2023-11-26	1797	2602	729	806	1637	518	2150

※ 데이터 시각화

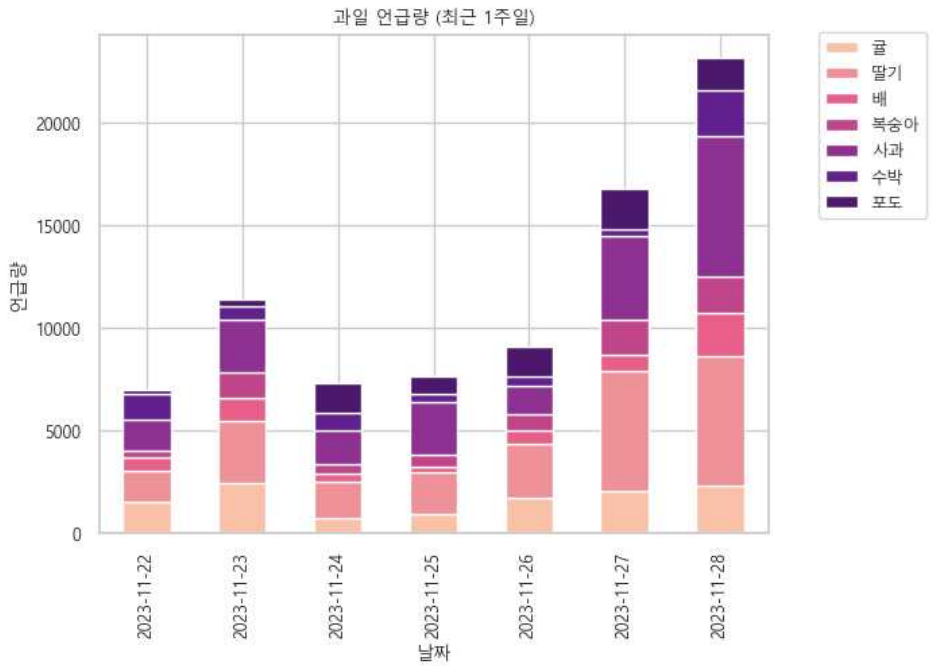
[과일 언급량 추이 시각화]

```
# 선 그래프
colors = ['#f7bba6', '#ed8495', '#e05286', '#a73b8f', '#aadacc', '#44a7cb', '#2a5599']  # 색상 리스트
sns .set(style ='whitegrid', font ='Malgun Gothic', font_scale =.8 , palette =colors )
ax = sns .lineplot(data =grouped_data , markers =True , dashes =False )
ax .legend(loc ='upper left', bbox_to_anchor =(1.02 , 1.02 ))  # 범례 위치 조정
ax .set(xlabel ='날짜', ylabel ='언급량',title ='과일 언급량 변화 (최근 1주일)')
```



[과일 언급량 추이 시각화]

```
# 누적 막대 차트
colors = ['#f8c1a8', '#ef9198', '#e8608a', '#c0458a', '#8f3192', '#63218f', '#4b186c', '#33104a' ]
grouped_data .plot (kind ='bar', stacked =True , color =colors )
plt .title ('과일 언급량 (최근 1주일)')
plt .xlabel ('날짜')
plt .ylabel ('언급량')

# 범례 위치와 레이아웃 조정
plt .legend (loc ='upper right', bbox_to_anchor =(1.25 , 1.02 ))
plt .show ()
```



결 과

대체로 딸기와 사과의 언급량이 가장 높았으며 증가 추세를 보였다. 이외의 과일들도 대체로 증가 추세를 보이고 있다. 과일 전체의 언급량도 점점 증가하는 추세를 보이며, 마지막 날인 28일에 가장 높은 언급량을 보였다.

8. 결론

1. 연령대별 섭취량 분석

✓ 연령대가 증가할수록 과일 섭취량도 증가하는 경향을 보였다.
✓ 특히 70대 이상 연령층에서 가장 높은 과일 섭취량을 나타냈다.
✓ 30대 이하의 연령대에서는 대체로 섭취량이 감소하는 추세를 보였다.

2. 소득계층별 섭취량 분석

✓ 2016년부터 2020년까지는 소득 수준과 관계없이 과일 섭취량이 감소하는 경향을 보였으나, 2020년에 전 소득 계층에서 과일 섭취량이 대폭 감소했다.
✓ 소득 수준별로 큰 차이 없이 과일 섭취량이 변동하며, 중위계층이 1위를 차지하였다.

3. 성별 섭취량 분석

✓ 여성의 과일 섭취량이 남성보다 높다.
✓ 모든 연도에서 여성의 과일 섭취량이 남성을 상회하며, 2020년에는 두 성별 모두 큰 폭으로 감소하는 추세를 보인다.

4. 과일의 가격 변동과 식품물가와의 상관관계

✓ 전반적으로 모든 과일의 평균 가격이 상승했다.
✓ 특히 2016년 대비 2021년에 가장 큰 상승을 보인 과일은 '배'이다.
✓ '딸기'와 '수박'은 최소값과 최대값의 차이가 크게 나타난다.
✓ 대부분의 과일이 양의 상관 관계를 가지며, 특히 '딸기'와 식품물가의 상관 관계가 매우 높다.
✓ '수박'은 음의 상관 관계를 가지며, 다른 과일들과는 반대로 움직이는 경향을 보인다.

5. 과일별 판매량 및 소비량

✓ '사과'의 한 달 판매량이 가장 높다.
✓ 판매량이 많은 순위는 '사과' 다음으로 '감귤', '딸기', '포도' 등의 순을 보인다.
✓ '복숭아'의 소비량은 감소 추세를 보인다.

6. 과일 재배지 분포

✓ 경상남도가 가장 높은 생산 분포를 보이며, 경상북도, 전라남도, 경기도 등이 그 뒤를 잇는다.
✓ 수도권과 부산, 광주 등의 지역에서는 낮은 생산 분포를 보인다.

7. 과일 특성 분류 모델

✓ 종경과 횡경이 비례하는 분포를 보이며, 배가 가장 큰 비율을 차지한다.
✓ 종경, 횡경, 산도, 당도 등이 0.0~0.4 사이에 가장 많이 분포하며, 횡경과 종경이 밀접한 관계를 보인다.
✓ KNN 분류 알고리즘을 사용했을 때 로지스틱 회귀보다 더 높은 정확도를 나타낸다.

8. 소비자의 과일 선호도 및 관심도

✓ 소비자들의 과일 선호도는 '배'가 가장 높았으며, '딸기'와 '사과'도 높은 순위를 차지한다.
✓ 최근 일주일간의 관심도는 대체로 상승 추세이며, '딸기'와 '사과'의 언급량이 가장 높다.

총 평

과일 섭취량은 연령, 소득 수준, 성별 등에 영향을 받고 있으며, 과일 가격과 식품물가와의 상관 관계도 존재한다.
지역별로는 경상남도가 과일 생산에서 선도적인 역할을 하며, 과일 특성 분류 모델과 판매량 분석 등을 통해 소비자의 과일 선호도 및 판매 동향을 파악할 수 있다고 생각했다. 최근 관심도에서는 '딸기'와 '사과'가 주목받고 있으며,
이러한 트렌드를 고려하여 과일 시장에서의 전략 수립이 가능할 것 같다.