

데이터마이닝

- 차원 축소 -



2023.03.20 (MON)

동덕여자대학교 HCI사이언스

파이썬 패키지 불러오기

필요한 파이썬 라이브러리 불러오기

```
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
from sklearn import preprocessing
import matplotlib.pyplot as plt
```

예제 데이터

■ 보스턴 주택 데이터(BostonHousing.csv)

- <https://github.com/reisanar/datasets/blob/master/BostonHousing.csv>

■ 아침 식사용 시리얼 데이터(Cereals.csv)

- <https://github.com/reisanar/datasets/blob/master/Cereals.csv>

■ 와인 데이터(wine.csv)

- <https://gist.github.com/tijptjik/9408623>

데이터 요약과 차원 축소

■ 데이터 요약

- 데이터 요약은 데이터 탐색의 중요한 구성 요소
- 요약 통계(평균, 중앙값 등) 및 그래픽 요약
- 일반적인 요약 통계
 - 평균, 중앙값, 최소값, 최댓값, 표준편차, 개수 및 백분율

■ 차원 축소

- 고차원의 원본 데이터를 저차원의 부분 공간으로 투영하여 데이터를 축소하는 방법
- 정확도의 희생을 최소로 하여 독립 변수 또는 입력 변수의 차원을 축소하는 방법을 찾는 것
- 요인 선택(factor selection) 또는 특징 추출(feature extraction)
- 데이터의 정보를 더 작은 하위 집합으로 압축하는 데 유용
- 유사한 범주를 결합하여 범주형 변수를 줄일 수 있음

차원 축소 방법

- 주어진 데이터에 도메인 지식을 적용해 범주를 제거하거나 결합하기
- 데이터 요약을 사용해 변수 간 중복 정보를 검출하고 불필요한 변수 및 범주를 제거하거나 합치기
- 데이터 변환 기술을 사용해 범주형 변수를 수치형 변수로 변환하기
- 주성분 분석(PCA) 같은 자동화된 차원 축소 기술을 사용하기
 - 주성분 분석은 원래 수치 데이터셋을 더 적은 변수에 대부분의 원래 정보를 포함하는 원래 데이터의 더 작은 가중 평균 셋으로 변환

예제 1: 보스턴 주택 데이터

```
bostonHousing_df = dmba.load_data('BostonHousing.csv')
bostonHousing_df = bostonHousing_df.rename(columns={'CAT', 'MEDV': 'CAT_MEDV'})
bostonHousing_df.head(9)
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT	MEDV	CAT_MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	4.98	24.0	0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	9.14	21.6	0
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	4.03	34.7	1
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	2.94	33.4	1
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	5.33	36.2	1
5	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222	18.7	5.21	28.7	0
6	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	12.43	22.9	0
7	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311	15.2	19.15	27.1	0
8	0.21124	12.5	7.87	0	0.524	5.631	100.0	6.0821	5	311	15.2	29.93	16.5	0

보스턴 주택 데이터셋의 변수 설명

CRIM	범죄율
ZN	25,000제곱피트 이상의 부지에 대해 구획된 주거용 토지의 비율
INDUS	비소매업이 차지하는 토지 비율
CHAS	찰스강 인접 여부(1=인접, 0=비인접)
nox	10ppm당 일산화질소
RM	주택의 평균 방 개수
AGE	1940년 이전에 건축된 주택에 사는 비율
DIS	보스턴 5대 상업 지구와의 거리
RAD	고속도로 진입 용이성 정도
TAX	재산세율(10,000달러당)
PTRATIO	시 ^{town} 별 학생 대 교사 비율
LSTAT	저소득층 비율
MEDV	주택 가격의 중앙값(단위: 1,000달러)
CAT_MEDV	주택 가격의 중앙값이 3만 달러 이상인지 여부(1=이상, 0=미만)

요약 통계(1)

```
bostonHousing_df.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	12.653063
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	7.141062
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	1.730000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	6.950000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	11.360000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	16.955000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	37.970000

```
print('Mean : ', bostonHousing_df.CRIM.mean())
print('Std. dev : ', bostonHousing_df.CRIM.std())
print('Min : ', bostonHousing_df.CRIM.min())
print('Max : ', bostonHousing_df.CRIM.max())
print('Median : ', bostonHousing_df.CRIM.median())
print('Length : ', len(bostonHousing_df.CRIM))

print('Number of missing values : ', bostonHousing_df.CRIM.isnull().sum())
```

```
Mean : 3.6135235573122535
Std. dev : 8.601545105332487
Min : 0.00632
Max : 88.9762
Median : 0.25651
Length : 506
Number of missing values : 0
```

요약 통계(1)

모든 변수에 대해서 mean, standard dev., min, max, median, length, and missing values 계산

```
pd.DataFrame({'mean': bostonHousing_df.mean(),  
             'sd': bostonHousing_df.std(),  
             'min': bostonHousing_df.min(),  
             'max': bostonHousing_df.max(),  
             'median': bostonHousing_df.median(),  
             'length': len(bostonHousing_df),  
             'miss.val': bostonHousing_df.isnull().sum(),  
             })
```

	mean	sd	min	max	median	length	miss.val
CRIM	3.613524	8.601545	0.00632	88.9762	0.25651	506	0
ZN	11.363636	23.322453	0.00000	100.0000	0.00000	506	0
INDUS	11.136779	6.860353	0.46000	27.7400	9.69000	506	0
CHAS	0.069170	0.253994	0.00000	1.0000	0.00000	506	0
NOX	0.554695	0.115878	0.38500	0.8710	0.53800	506	0
RM	6.284634	0.702617	3.56100	8.7800	6.20850	506	0
AGE	68.574901	28.148861	2.90000	100.0000	77.50000	506	0
DIS	3.795043	2.105710	1.12960	12.1265	3.20745	506	0
RAD	9.549407	8.707259	1.00000	24.0000	5.00000	506	0
TAX	408.237154	168.537116	187.00000	711.0000	330.00000	506	0
PTRATIO	18.455534	2.164946	12.60000	22.0000	19.05000	506	0
LSTAT	12.653063	7.141062	1.73000	37.9700	11.36000	506	0
MEDV	22.532806	9.197104	5.00000	50.0000	21.20000	506	0
CAT_MEDV	0.166008	0.372456	0.00000	1.0000	0.00000	506	0

요약 통계(2)

```
bostonHousing_df.corr().round(3)
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT	MEDV	CAT_MEDV
CRIM	1.000	-0.200	0.407	-0.056	0.421	-0.219	0.353	-0.380	0.626	0.583	0.290	0.456	-0.388	-0.152
ZN	-0.200	1.000	-0.534	-0.043	-0.517	0.312	-0.570	0.664	-0.312	-0.315	-0.392	-0.413	0.360	0.365
INDUS	0.407	-0.534	1.000	0.063	0.764	-0.392	0.645	-0.708	0.595	0.721	0.383	0.604	-0.484	-0.366
CHAS	-0.056	-0.043	0.063	1.000	0.091	0.091	0.087	-0.099	-0.007	-0.036	-0.122	-0.054	0.175	0.109
NOX	0.421	-0.517	0.764	0.091	1.000	-0.302	0.731	-0.769	0.611	0.668	0.189	0.591	-0.427	-0.233
RM	-0.219	0.312	-0.392	0.091	-0.302	1.000	-0.240	0.205	-0.210	-0.292	-0.356	-0.614	0.695	0.641
AGE	0.353	-0.570	0.645	0.087	0.731	-0.240	1.000	-0.748	0.456	0.506	0.262	0.602	-0.377	-0.191
DIS	-0.380	0.664	-0.708	-0.099	-0.769	0.205	-0.748	1.000	-0.495	-0.534	-0.232	-0.497	0.250	0.119
RAD	0.626	-0.312	0.595	-0.007	0.611	-0.210	0.456	-0.495	1.000	0.910	0.465	0.489	-0.382	-0.198
TAX	0.583	-0.315	0.721	-0.036	0.668	-0.292	0.506	-0.534	0.910	1.000	0.461	0.544	-0.469	-0.274
PTRATIO	0.290	-0.392	0.383	-0.122	0.189	-0.356	0.262	-0.232	0.465	0.461	1.000	0.374	-0.508	-0.443
LSTAT	0.456	-0.413	0.604	-0.054	0.591	-0.614	0.602	-0.497	0.489	0.544	0.374	1.000	-0.738	-0.470
MEDV	-0.388	0.360	-0.484	0.175	-0.427	0.695	-0.377	0.250	-0.382	-0.469	-0.508	-0.738	1.000	0.790
CAT_MEDV	-0.152	0.365	-0.366	0.109	-0.233	0.641	-0.191	0.119	-0.198	-0.274	-0.443	-0.470	0.790	1.000

요약 통계(3)

```
bostonHousing_df = dmba.load_data('BostonHousing.csv')
bostonHousing_df = bostonHousing_df.rename(columns={'CAT. MEDV': 'CAT_MEDV'})
bostonHousing_df.CHAS.value_counts()
```

```
0    471
1     35
```

변수 하나로 취합

Name: CHAS, dtype: int64

35개 지역은 CHAS 값이 "1" (찰스강 경계에 인접)

요약 통계(4)

```
# pd.cut 메서드를 사용하여 변수에 대해 크기 1의 bins 생성
# 기본적으로 메서드는 범주형 변수를 생성 (0, 7].
# 인수 labels=False는 대신 정수를 결정. 예) 0.
# pd.cut(X, bins, labels). bins=[start, end] => (미포함, 포함).
# bin 구간 대비 작거나 큰 수. if bin 첫 번째 구간보다 작으면 --> NaN, 마지막 구간보다 크면 --> Nan
bostonHousing_df['RM_bin'] = pd.cut(bostonHousing_df.RM, range(0, 10), labels=False)
bostonHousing_df.head(5)
```

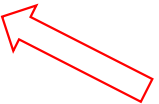
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT	MEDV	CAT_MEDV	RM_bin
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	4.98	24.0	0	6
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	9.14	21.6	0	6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	4.03	34.7	1	7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	2.94	33.4	1	6
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	5.33	36.2	1	7

요약 통계(4)

```
# 비닝된 RM 및 CHAS로 MEDV의 평균을 계산  
# groupby 방법을 사용하여 데이터 프레임을 그룹화한 다음 분석을 MEDV로 제한하고 각 그룹에 대한 평균을 결정  
bostonHousing_df.groupby(['RM_bin', 'CHAS'])['MEDV'].mean()
```

RM_bin	CHAS	
3	0	25.300000
4	0	15.407143
5	0	17.200000
	1	22.218182
6	0	21.769170
	1	25.918750
7	0	35.964444
	1	44.066667
8	0	45.700000
	1	35.950000

Name: MEDV, dtype: float64



여러 변수에 대해 레코드를 취합하고
요약 통계량(도수, 평균값, 중앙값 등)을 계산

상관 분석

- 변수의 중복 탐지에 좋음
- 상관 계수 표에 대한 히트맵을 사용해 강한 상관관계가 있는 변수들을 쉽게 식별 가능

피벗 테이블

```
# pivot_table() 함수를 사용해 피벗 테이블 작성
bostonHousing_df = dmba.load_data('BostonHousing.csv')
bostonHousing_df = bostonHousing_df.rename(columns={'CAT', 'MEDV': 'CAT_MEDV'})
# 사이즈 1의 빈 생성
bostonHousing_df['RM_bin'] = pd.cut(bostonHousing_df.RM, range(0, 10), labels=False)

# pivot_table()을 사용하여 데이터를 재구성하고 피벗 테이블을 생성
# pivot : 회전 축, Pivot Table : 표로 정리해서 보고자 하는 관점의 축을 잡고 표로 만들어 주는 기능
# pdf1 = pd.pivot_table(df, # 피벗할 데이터프레임
# # index = 'class', # 행 위치에 들어갈 열
# # columns = 'Gender', # 열 위치에 들어갈 열
# # values = 'age', # 데이터로 사용할 열
# # aggfunc = 'mean') # 데이터 집계 함수
pd.pivot_table(bostonHousing_df, values='MEDV', index=['RM_bin'], columns=['CHAS'],
               aggfunc=np.mean, margins=True)
```

피벗 테이블 작성

CHAS	0	1	All
RM_bin			
3	25.300000	NaN	25.300000
4	15.407143	NaN	15.407143
5	17.200000	22.218182	17.551592
6	21.769170	25.918750	22.015985
7	35.964444	44.066667	36.917647
8	45.700000	35.950000	44.200000
All	22.093843	28.440000	22.532806

찰스강과 접하지 않는 방이 8개인 지역은 MEDV = 45.7

피벗 테이블

```
col = ['Machine', 'Country', 'Grade', 'Price', 'Count']
data = [['TV', 'Korea', 'A', 1000, 3],
        ['TV', 'Korea', 'B', 800, 8],
        ['TV', 'Korea', 'B', 800, 2],
        ['TV', 'Japan', 'A', 1300, 5],
        ['TV', 'Japan', 'A', 1300, 1],
        ['PC', 'Korea', 'B', 1500, 6],
        ['PC', 'Korea', 'A', 2000, 9],
        ['PC', 'Japan', 'A', 3000, 3],
        ['PC', 'Japan', 'B', 2500, 3]]
df = pd.DataFrame(data=data, columns=col)
print(df)
```

	Machine	Country	Grade	Price	Count
0	TV	Korea	A	1000	3
1	TV	Korea	B	800	8
2	TV	Korea	B	800	2
3	TV	Japan	A	1300	5
4	TV	Japan	A	1300	1
5	PC	Korea	B	1500	6
6	PC	Korea	A	2000	9
7	PC	Japan	A	3000	3
8	PC	Japan	B	2500	3

```
# index를 Machine, Country로 하고 columns를 Grade로 설정하고 Count값들을 np.sum으로 합계를 계산한 결과를
# Pivot_table()을 사용하여 출력하라.
print(df.pivot_table(values='Count', index=['Machine', 'Country'], columns='Grade', aggfunc=np.sum))
```

		Grade	A	B
Machine	Country			
PC	Japan		3.0	3.0
	Korea		9.0	6.0
TV	Japan		6.0	NaN
	Korea		3.0	10.0

범주 축소(1)

■ 교차 분석표 (Crosstab)

- Pandas에서 범주형 데이터 2개를 비교 분석 할 때 유용한 표

```
# Crosstab :  
# pd.crosstab(index, columns, values=None, rownames=None, colnames=None, aggfunc=None,  
#             margins=False, margins_name='All', dropna=True, normalize=False)  
# index (행으로 그룹화할 값), columns (열로 그룹화할 값)은 필수 입력  
# 두 변수의 교차 표 작성을 위해 pd.crosstab 메소드를 사용  
# 두 번째 단계에서는 개수를 열을 따라 백분율로 변환  
tbl = pd.crosstab(bostonHousing_df.CAT_MEDV, bostonHousing_df.ZN)  
propTbl = tbl / tbl.sum()  
propTbl.round(2)
```

	ZN	0.0	12.5	17.5	18.0	20.0	21.0	22.0	25.0	28.0	30.0	...	55.0	60.0	70.0	75.0	80.0	82.5	85.0	90.0	95.0	100.0
CAT_MEDV																						
0	0.91	1.0	0.0	1.0	0.24	1.0	0.9	1.0	1.0	1.0	1.0	...	0.67	0.75	1.0	0.33	0.67	0.5	1.0	0.0	0.0	0.0
1	0.09	0.0	1.0	0.0	0.76	0.0	0.1	0.0	0.0	0.0	0.0	...	0.33	0.25	0.0	0.67	0.33	0.5	0.0	1.0	1.0	1.0

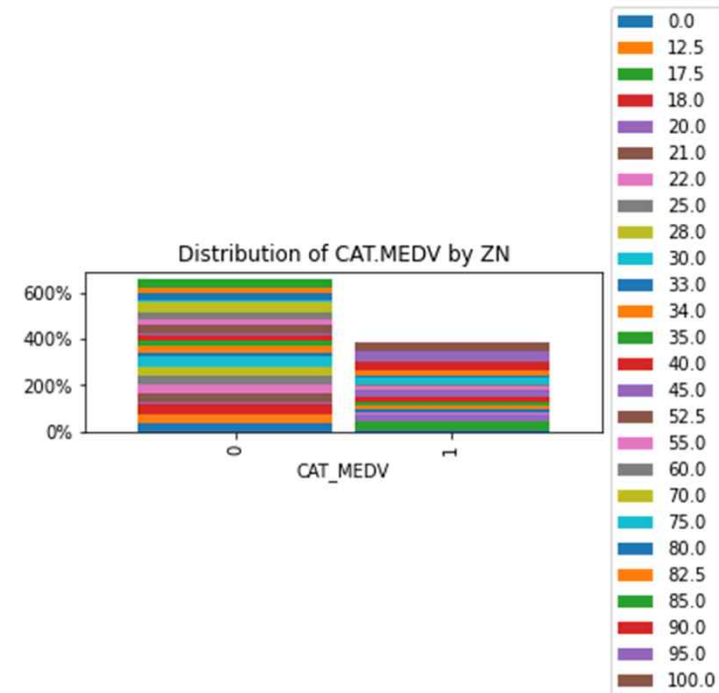
2 rows × 26 columns

범주 축소(1)

■ 교차 분석표 (Crosstab)

- Pandas에서 범주형 데이터 2개를 비교 분석 할 때 유용한 표

```
#누적 막대 차트에서 비율을 플로팅
ax = propTbl.plot(kind='bar', stacked=True, width=0.9)
ax.set_yticklabels(['{:, .0%}'.format(x) for x in ax.get_yticks()])
plt.title('Distribution of CAT.MEDV by ZN')
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.tight_layout()
plt.show()
# x축, y축에 대한 설정이 거꾸로 되어 있다.
```

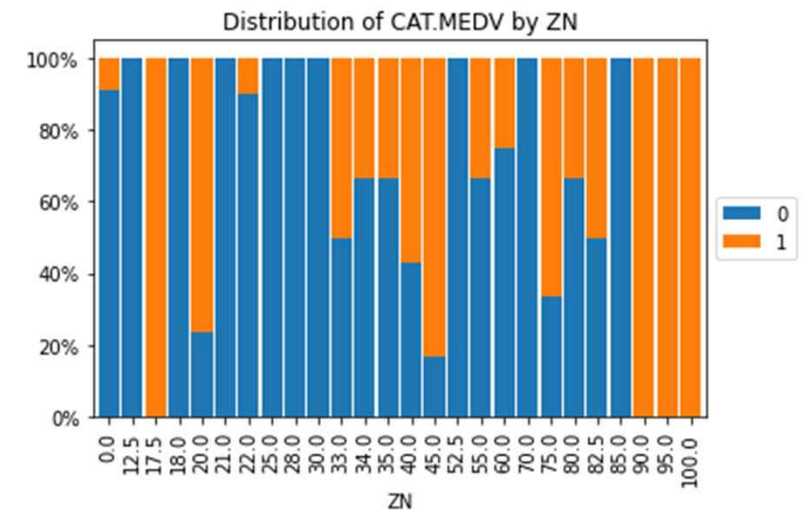


범주 축소(1)

■ 교차 분석표 (Crosstab)

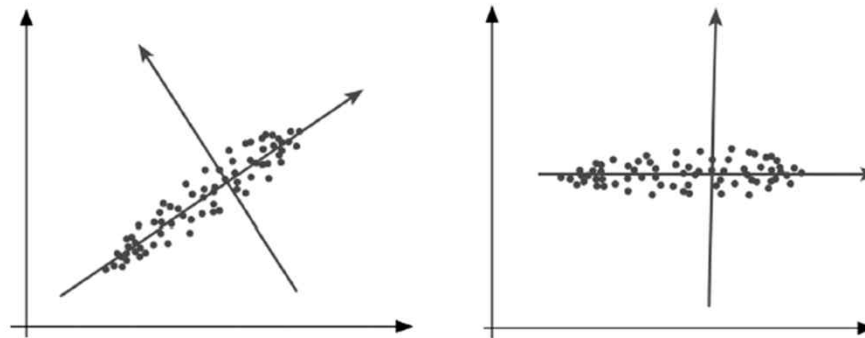
- Pandas에서 범주형 데이터 2개를 비교 분석 할 때 유용한 표

```
ax = propTbl.transpose().plot(kind='bar', stacked=True, width=0.9)
ax.set_yticklabels(['{:, .0%}'.format(x) for x in ax.get_yticks()])
plt.title('Distribution of CAT.MEDV by ZN')
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
plt.tight_layout()
plt.show()
```

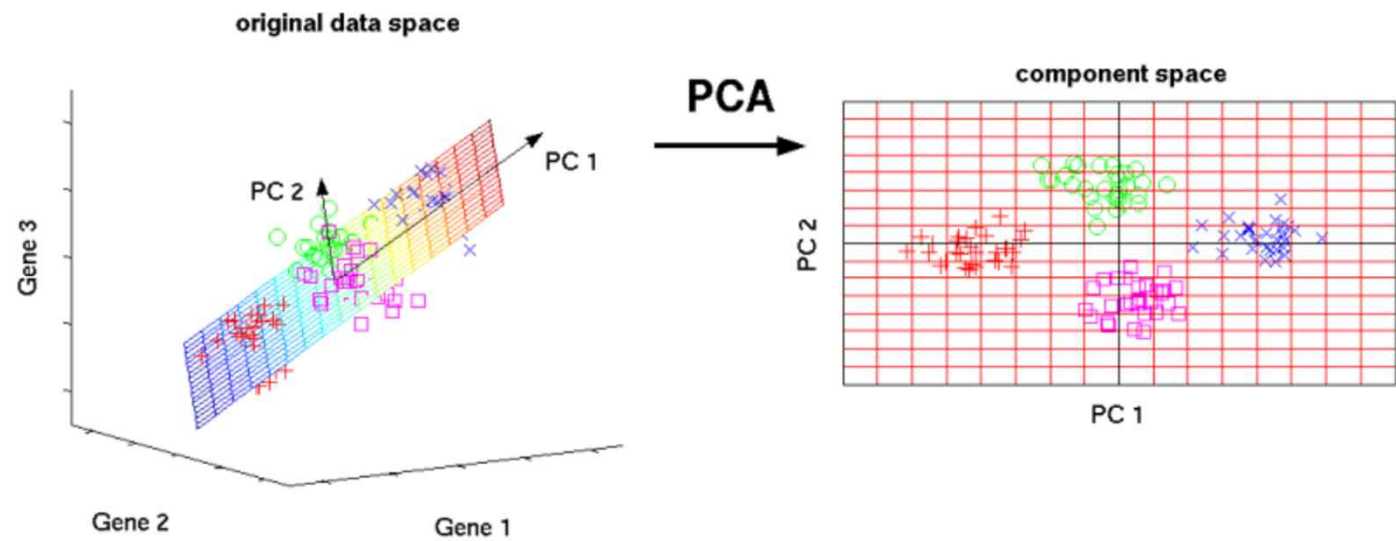
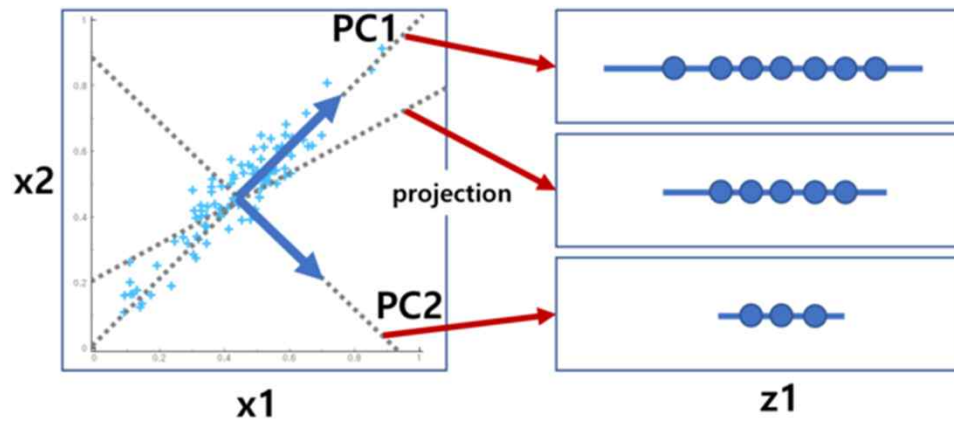


주성분 분석(PCA)

- 원 데이터의 분포를 최대한 보존하면서 고차원 공간의 데이터들을 저차원 공간으로 변환하는 기법
- 독립 변수 간 선형 결합을 통해 주성분을 만들어 변수의 수를 줄이는 비지도학습 방법론. 주성분 분석은 차원 감소, 영상 인식, 노이즈 제거 등에 활용
- 서로 상관관계를 갖는 설명 변수들의 선형 결합을 이용해 상호 독립적인 주성분(principal components)이라는 인공 변수를 만들기 위한 분석 방법
- 주성분 분석은 복잡한 다차원 데이터를 저차원으로 변환해 더 쉽게 이해할 수 있도록 변환. 따라서, parameter의 수가 많을 때의 차원 축소에 유용
 - ➔ 각 주성분은 주어진 설명 변수의 정보를 최대한 반영하는 것을 목적으로 함



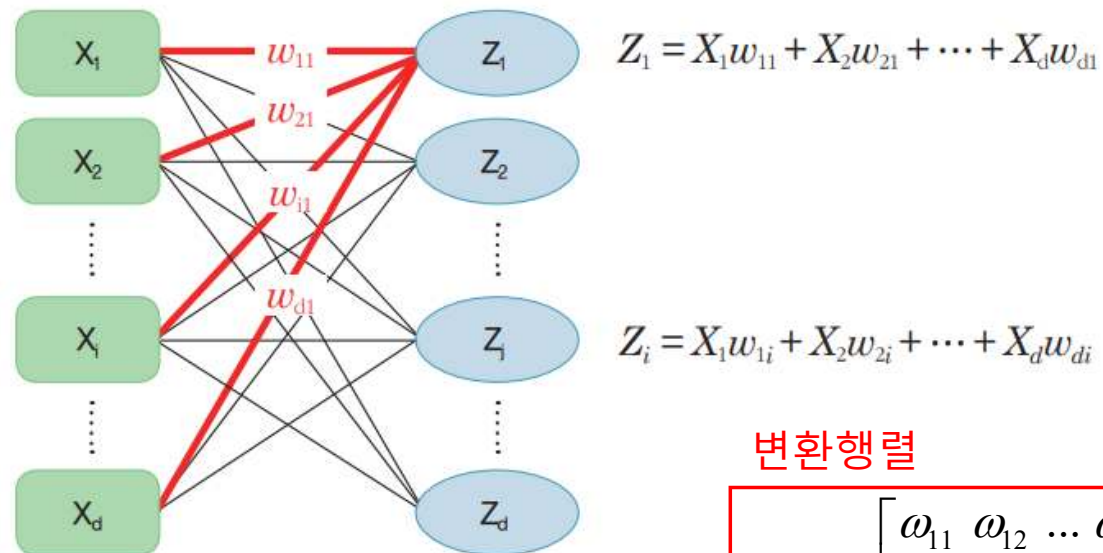
주성분 분석(PCA)



주성분 분석(PCA)

- 선형 변환 : 주성분의 축으로 표현되는 데이터(Z) == 기존 변수들의 데이터(X)를 선형 변환

포본	X_1	X_2
1	50	73
2	65	75
3	75	80
4	80	82
5	95	85



$$\mathbf{Z} = \mathbf{XW}$$

변환행렬

$$\mathbf{W} = \begin{bmatrix} \omega_{11} & \omega_{12} & \dots & \omega_{1d} \\ \omega_{21} & \omega_{22} & \dots & \omega_{2d} \\ \dots & \dots & \dots & \dots \\ \omega_{d1} & \omega_{d2} & \dots & \omega_{dd} \end{bmatrix}$$

주성분 분석(PCA)

■ 주성분

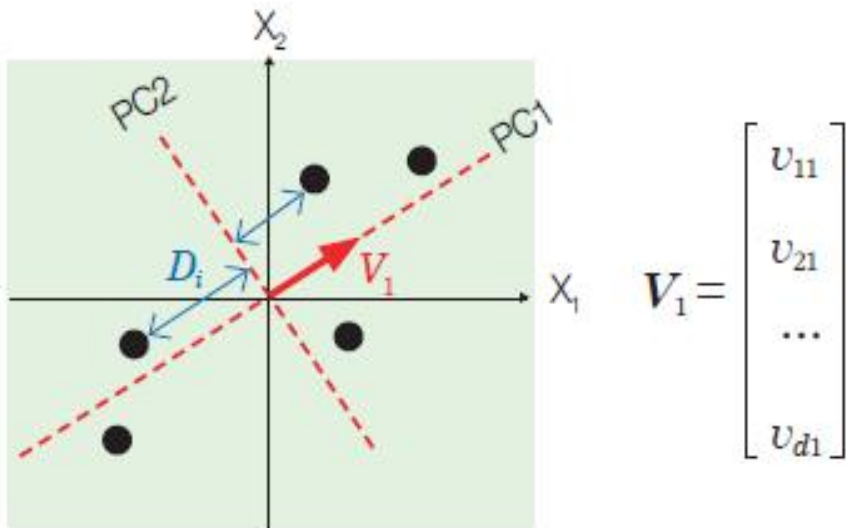
\mathbf{X} : 원 행렬의 각 열에 해당하는 변수 x_i 를 평균은 0, 표준편차는 1이 되도록 표준화

\mathbf{V}_i : 주성분 방향의 단위 벡터

\mathbf{XV}_i : 각 표본들을 주성분 방향으로 사영한 값

$\text{Var}(\mathbf{XV}_i)$: 주성분 방향의 분산

Σ : 공분산 행렬



$$\text{Var}(\mathbf{XV}_i) = \mathbf{V}_i^T \Sigma \mathbf{V}_i$$

$$\Sigma = \begin{bmatrix} \text{Var}(X_1) & \text{Cov}(X_1, X_2) & \cdots & \text{Cov}(X_1, X_d) \\ \text{Cov}(X_2, X_1) & \text{Var}(X_2) & \cdots & \text{Cov}(X_2, X_d) \\ \cdots & \cdots & \cdots & \cdots \\ \text{Cov}(X_d, X_1) & \text{Cov}(X_d, X_2) & \cdots & \text{Var}(X_d) \end{bmatrix}$$

주성분 분석(PCA)

■ 최대화 문제

단위벡터의 크기가 1인 제약조건을 갖으며, $\text{Var}(\mathbf{XV}_i) = \mathbf{V}_i^T \boldsymbol{\Sigma} \mathbf{V}_i$ 를 최대로 하는 최대화 문제

$$\begin{aligned} \max \mathbf{V}_i^T \boldsymbol{\Sigma} \mathbf{V}_i \\ \text{subject to } ||\mathbf{V}_i|| = 1 \end{aligned}$$

이를 만족하는 조건을 찾기 위해 라그랑주 승수법(Lagrange Multiplier)으로 풀면, $\boldsymbol{\Sigma}$ 에 대해 \mathbf{V}_i 가

$$\boldsymbol{\Sigma} \mathbf{V}_i = \lambda_i \mathbf{V}_i$$

를 만족할 때 분산이 최대가 됨
 λ_i 와 \mathbf{V}_i 는 $\boldsymbol{\Sigma}$ 에 대한 고유벡터와 고유값을 의미함

주성분 분석(PCA)

■ 주성분으로의 변환 행렬

$$\mathbf{Z} = \mathbf{XV} (= \mathbf{XW})$$

\mathbf{V} 는 \mathbf{v}_i 를 열벡터로 하는 행렬
주성분 로딩: \mathbf{v}_i 의 원소

■ 분산 비율과 특성 추출

고유벡터와 고유값은 각각 d 개 존재

$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d$ 라고 하면, 주성분 PC_i 에 해당하는 분산은 λ_i 임

분산의 총합 대비 주성분 PC_i 의 상대적 분산 비율:
$$\frac{\lambda_i}{\sum_{i=1}^d \lambda_i}$$

특성 추출:

- ① 분산 비율이 임계치(일반적으로 90%)를 넘는 최소 수
- ② scree plot에서 elbow method 적용: 분산 비율의 기울기가 완만해지는 elbow point로 정함

주성분 분석(PCA) : 예제 1

■ 데이터 작성 및 표준화

$$x'_i = \frac{x_i - \mu_x}{\sigma_x}$$

표본	원데이터		표준화	
	X_1	X_2	X_1	X_2
1	50	73	$x_{11} = -1.5299$	$x_{12} = -1.3553$
2	65	75	$x_{21} = -0.5322$	$x_{22} = -0.9035$
3	75	80	$x_{31} = 0.1330$	$x_{32} = 0.2259$
4	80	82	$x_{41} = 0.4656$	$x_{42} = 0.6776$
5	95	85	$x_{51} = 1.4634$	$x_{52} = 1.3553$

주성분 분석(PCA) : 예제 1

■ 공분산 행렬

$$\text{Cov}(X_i, X_j) = \frac{1}{n} \sum_{k=1}^n (x_{ik} - \mu_i)(x_{jk} - \mu_j)$$

$$\Sigma = \begin{bmatrix} 1 & 0.9766 \\ 0.9766 & 1 \end{bmatrix}$$

주성분 분석(PCA) : 예제 1

■ 고유 값과 고유 벡터

$$\Sigma V = \lambda V$$

고윳값

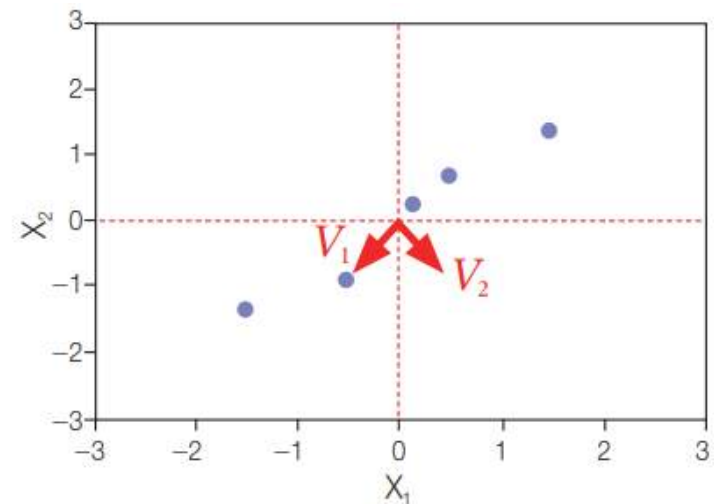
$$\lambda_1 = 1.9776, \lambda_2 = 0.0224$$

고유벡터

$$V_1 = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \text{ (또는 } \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix})$$

$$V_2 = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \text{ (또는 } \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix})$$

주성분1(PC1): V_1 방향
주성분2(PC2): V_2 방향



주성분 분석(PCA) : 예제 1

■ 주성분 추출

$$PC1 : \frac{\lambda_1}{\lambda_1 + \lambda_2} = \frac{1.9776}{1.9776 + 0.0224} = 0.9888$$

$$PC2 : \frac{\lambda_2}{\lambda_1 + \lambda_2} = \frac{0.0224}{1.9776 + 0.0224} = 0.0112$$

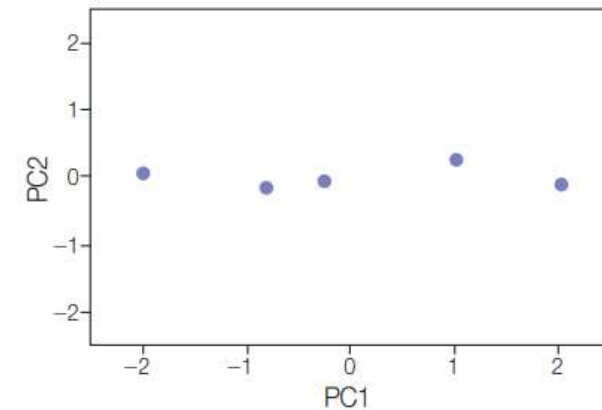
주성분1의 분산 비율이 임계치(90%) 이상이기 때문에 주성분 1만 추출

주성분 분석(PCA) : 예제 1

■ 주성분으로의 데이터 변환

$$Z = \begin{bmatrix} -1.5299 & -1.3553 \\ -0.5322 & -0.9035 \\ 0.1330 & 0.2259 \\ 0.4656 & 0.6776 \\ 1.4634 & 1.3553 \end{bmatrix} \begin{bmatrix} -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 2.0401 & -0.1235 \\ 1.0152 & 0.2626 \\ -0.2538 & -0.0656 \\ -0.8084 & -0.1499 \\ -1.9931 & 0.0765 \end{bmatrix}$$

No	원데이터		표준화		주성분	
	X ₁	X ₂	X ₁	X ₂	Z ₁	Z ₂
1	50	73	-1.5299	-1.3553	2.0401	-0.1235
2	65	75	-0.5322	-0.9035	1.0152	0.2626
3	75	80	0.1330	0.2259	-0.2538	-0.0656
4	80	82	0.4656	0.6776	-0.8084	-0.1499
5	95	85	1.4634	1.3553	-1.9931	0.0765

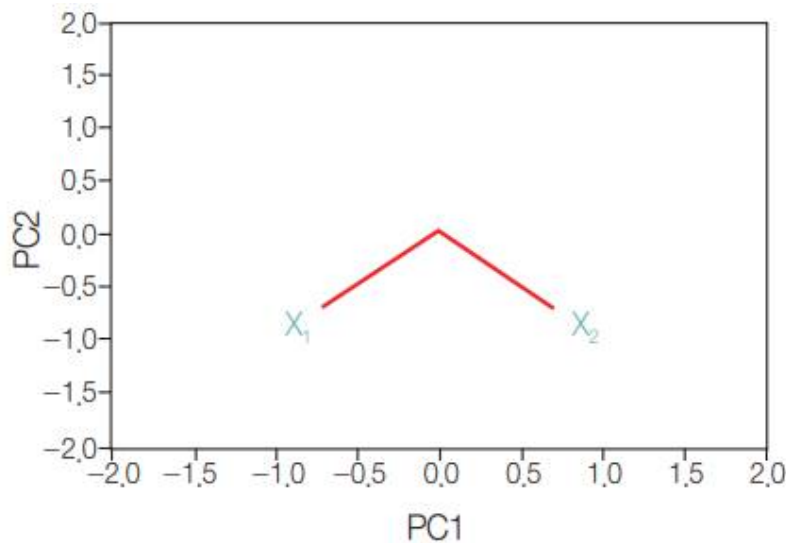


주성분 분석(PCA) : 예제 1

■ 시각화와 해석

고유벡터: 주성분의 로딩에 해당하며,
기존변수들이 주성분으로의 변환에 미치는 영향도

loading plot: 기존변수들이 주성분에 미치는 정도를 나타낸 그래프 크기는 고유벡터의 값과 동일



\mathbf{v} 의 1열인 고유벡터 \mathbf{v}_1 은 x_1 과 x_2 변수의 값을 주성분1(PC1)로 변환 시, $-\frac{1}{\sqrt{2}}$ 만큼 동일한 영향을 미침

\mathbf{v} 의 2열인 고유벡터 \mathbf{v}_2 는 x_1 과 x_2 변수의 값을 주성분2 (PC2)로 변환 시, 각각 $\frac{1}{\sqrt{2}}$ 과 $-\frac{1}{\sqrt{2}}$ 만큼 영향을 미침

예제 2: 아침 식사용 시리얼 데이터

```
cereals_df = dmba.load_data('Cereals.csv')
cereals_df.head(10)
```

	name	mfr	type	calories	protein	fat	sodium	fiber	carbo	sugars	potass	vitamins	shelf	weight	cups	rating
0	100%_Bran	N	C	70	4	1	130	10.0	5.0	6.0	280.0	25	3	1.00	0.33	68.402973
1	100%_Natural_Bran	Q	C	120	3	5	15	2.0	8.0	8.0	135.0	0	3	1.00	1.00	33.983679
2	All-Bran	K	C	70	4	1	260	9.0	7.0	5.0	320.0	25	3	1.00	0.33	59.425505
3	All-Bran_with_Extra_Fiber	K	C	50	4	0	140	14.0	8.0	0.0	330.0	25	3	1.00	0.50	93.704912
4	Almond_Delight	R	C	110	2	2	200	1.0	1.0	1.0	10.0	0	1	1.00	0.50	10.000000
5	Apple_Cinnamon_Cheerios	G	C	110	2	2	180	1.5	1.0	1.0	10.0	0	1	1.00	0.50	10.000000
6	Apple_Jacks	K	C	110	2	0	125	1.0	1.0	1.0	10.0	0	1	1.00	0.50	10.000000
7	Basic_4	G	C	130	3	2	210	2.0	1.0	1.0	10.0	0	1	1.00	0.50	10.000000
8	Bran_CheX	R	C	90	2	1	200	4.0	1.0	1.0	10.0	0	1	1.00	0.50	10.000000
9	Bran_Flakes	P	C	90	3	0	210	5.0	1.0	1.0	10.0	0	1	1.00	0.50	10.000000

mfr	시리얼 제조 업체(American Home Food Products, General Mills, Kellogg 등)
type	저온용 또는 고온용
calories	1회분에 대한 칼로리
protein	단백질(g)
fat	지방(g)
sodium	나트륨(mg)
fiber	식이섬유(g)
carbo	복합 탄수화물(g)
sugars	설탕(g)
potass	칼륨(mg)
vitamins	비타민과 미네랄: FDA의 권장 비율로서 0, 25 또는 100을 나타냄
shelf	디스플레이 선반(바닥에서부터 1, 2, 3으로 세어나감)
weight	1회분의 무게(Ounces)
cups	1회분에 제공되는 컵의 수
rating	소비자 보고서에 의한 시리얼 평점

예제 2: 아침 식사용 시리얼 데이터

```
cereals_df = dmba.load_data('Cereals.csv')
pcs = PCA(n_components=2)
pcs.fit(cereals_df[['calories', 'rating']])
```

```
PCA(n_components=2)
```

```
# 구성 요소의 중요성은 설명된 분산을 사용하여 평가할 수 있음
pcsSummary = pd.DataFrame({'Standard deviation': np.sqrt(pcs.explained_variance_),
                           'Proportion of variance': pcs.explained_variance_ratio_,
                           'Cumulative proportion': np.cumsum(pcs.explained_variance_ratio_)})

pcsSummary
pcsSummary.index = ['PC1', 'PC2']
pcsSummary.round(4)
```

	Standard deviation	Proportion of variance	Cumulative proportion
PC1	22.3165	0.8632	0.8632
PC2	8.8844	0.1368	1.0000

```
pcsSummary = pd.DataFrame({'Standard deviation': np.sqrt(pcs.explained_variance_),
                           'Proportion of variance': pcs.explained_variance_ratio_,
                           'Cumulative proportion': np.cumsum(pcs.explained_variance_ratio_)})

pcsSummary = pcsSummary.transpose()
pcsSummary.columns = ['PC1', 'PC2']
pcsSummary.round(4)
```

	PC1	PC2
Standard deviation	22.3165	8.8844
Proportion of variance	0.8632	0.1368
Cumulative proportion	0.8632	1.0000

예제 2: 아침 식사용 시리얼 데이터

```
# pcs의 components_ 필드는 개별 구성 요소를 제공할  
# 이 행렬의 열은 주성분 PC1, PC2로 정의할 수 있음. 행은 입력 행렬, 칼로리 및 등급에서 찾은 변수  
pcsComponents_df = pd.DataFrame(pcs.components_.transpose(), columns=['PC1', 'PC2'],  
                                index=['calories', 'rating'])  
pcsComponents_df
```

	PC1	PC2
calories	-0.847053	0.531508
rating	0.531508	0.847053

```
# Transform 방법을 사용하여 점수를 얻음  
scores = pd.DataFrame(pcs.transform(cereals_df[['calories', 'rating'])),  
                      columns=['PC1', 'PC2'])  
scores.head()
```

	PC1	PC2
0	44.921528	2.197183
1	-15.725265	-0.382416
2	40.149935	-5.407212
3	75.310772	12.999126
4	-7.041508	-5.357686

pcs 모델을 transform() 메서드를 사용하여 데이터 프레임의 칼로리 및 등급 열을 변환
'PC1' 및 'PC2' 열이 있는 변환된 데이터에서 점수라는 새 데이터 프레임을 만들

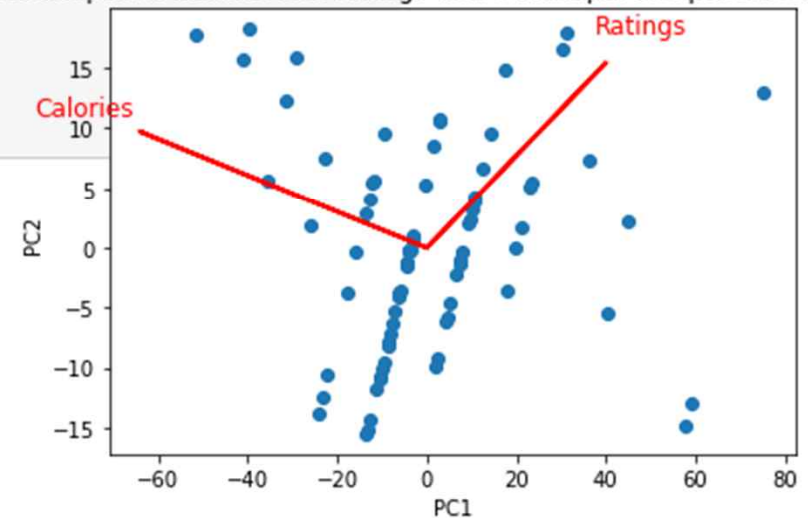
예제 2: 아침 식사용 시리얼 데이터

```
# Plot the scatterplot with principal component axes
fig, ax = plt.subplots()
ax.scatter(scores.PC1, scores.PC2)
ax.set_xlabel('PC1')
ax.set_ylabel('PC2')
ax.set_title('Scatterplot of Calories and Ratings with 2 Principal Component Directions')

# Add the original variables as arrows
for i, var in enumerate(pcs.components_.transpose()):
    ax.arrow(0, 0, var[0]*max(scores.PC1), var[1]*max(scores.PC2), head_width=0.05, head_length=0.1, linewidth=2,
            color='red')
    if i == 0:
        ax.text(var[0]*max(scores.PC1)*1.2, var[1]*max(scores.PC2)*1.2,
                'Calories', color='red', ha='center', va='center', fontsize=12)
    elif i == 1:
        ax.text(var[0]*max(scores.PC1)*1.2, var[1]*max(scores.PC2)*1.2,
                'Ratings', color='red', ha='center', va='center', fontsize=12)

plt.show()
```

Scatterplot of Calories and Ratings with 2 Principal Component Directions



예제 2: 아침 식사용 시리얼 데이터

```
# 처음 두 개(이름 및 mfr)를 제외한 시리얼_df 데이터 프레임의 모든 열에서 PCA를 수행
# 누락된 값이 있는 행을 모두 삭제합니다.
# 3단계에서와 같이 PCA 결과 요약물 작성하되 목록 이해 및 문자열 형식을 사용하여 열 이름을 'PC1', 'PC2' 등으로 지정
# 데이터 프레임을 소수점 이하 4자리로 반올림
pcs = PCA()
pcs.fit(cereals_df.iloc[:, 3:].dropna(axis=0))
pcsSummary_df = pd.DataFrame({'Standard deviation': np.sqrt(pcs.explained_variance_),
                              'Proportion of variance': pcs.explained_variance_ratio_,
                              'Cumulative proportion': np.cumsum(pcs.explained_variance_ratio_)})
pcsSummary_df = pcsSummary_df.transpose()
pcsSummary_df.columns = ['PC{}'.format(i) for i in range(1, len(pcsSummary_df.columns) + 1)]
pcsSummary_df.round(4)
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13
Standard deviation	83.7641	70.9143	22.6437	19.1815	8.4232	2.0917	1.6994	0.7796	0.6578	0.3704	0.1864	0.063	0.0
Proportion of variance	0.5395	0.3867	0.0394	0.0283	0.0055	0.0003	0.0002	0.0000	0.0000	0.0000	0.0000	0.000	0.0
Cumulative proportion	0.5395	0.9262	0.9656	0.9939	0.9993	0.9997	0.9999	1.0000	1.0000	1.0000	1.0000	1.000	1.0

예제 2: 아침 식사용 시리얼 데이터

```
# 데이터 세트의 주성분을 포함하는 pcsComponents_df라는 판다스 데이터 프레임을 생성
# pcs.components_ 속성에는 각각 원래 변수의 선형 조합인 주성분이 포함되어 있음
# 속성은 (n_components, n_features)의 형태를 가짐. n_components는 주성분의 수이고 n_features는 원래 변수의 수
# (이 코드에서는 구성 요소 수 지정 안함)
# 기본 형태를 전치하는 .transpose() 메서드를 사용하여 pcs.components_ 행렬을 전치시켜서 표현.
# 전치된 행렬을 기준으로 DataFrame을 생성하고 pcsSummary_df.columns 속성을 사용하여 열 이름을 할당
# 원래 데이터 세트인 cereals_df.iloc[:, 3:].columns의 열 이름을 사용하여 DataFrame에 행 레이블을 할당
# PC 축 처음부터 5개만 보여줌
pcsComponents_df = pd.DataFrame(pcs.components_.transpose(), columns=pcsSummary_df.columns,
                                index=cereals_df.iloc[:, 3:].columns)
pcsComponents_df.iloc[:, :5]
```

	PC1	PC2	PC3	PC4	PC5
calories	-0.077984	-0.009312	0.629206	-0.601021	0.454959
protein	0.000757	0.008801	0.001026	0.003200	0.056176
fat	0.000102	0.002699	0.016196	-0.025262	-0.016098
sodium	-0.980215	0.140896	-0.135902	-0.000968	0.013948
fiber	0.005413	0.030681	-0.018191	0.020472	0.013605
carbo	-0.017246	-0.016783	0.017370	0.025948	0.349267
sugars	-0.002989	-0.000253	0.097705	-0.115481	-0.299066
potass	0.134900	0.986562	0.036782	-0.042176	-0.047151
vitamins	-0.094293	0.016729	0.691978	0.714118	-0.037009
shelf	0.001541	0.004360	0.012489	0.005647	-0.007876
weight	-0.000512	0.000999	0.003806	-0.002546	0.003022
cups	-0.000510	-0.001591	0.000694	0.000985	0.002148
rating	0.075296	0.071742	-0.307947	0.334534	0.757708

주성분 분석(PCA)

```
# 시리얼 데이터 셋에서 13개의 수치형 변수를 모두 사용해 주성분 5개를 분석해보고자 함
# 처음 두 개(이름 및 mfr)를 제외한 시리얼_df 데이터 프레임의 모든 열에서 PCA를 수행
# 누락된 값이 있는 행을 모두 삭제합니다.
# 3단계에서와 같이 PCA 결과 요약물 작성하되 목록 이해 및 문자열 형식을 사용하여 열 이름을 'PC1', 'PC2' 등으로 지정
# 데이터 프레임을 소수점 이하 4자리로 반올림
pcs = PCA()
pcs.fit(cereals_df.iloc[:, 3:].dropna(axis=0))
pcsSummary_df = pd.DataFrame({'Standard deviation': np.sqrt(pcs.explained_variance_),
                              'Proportion of variance': pcs.explained_variance_ratio_,
                              'Cumulative proportion': np.cumsum(pcs.explained_variance_ratio_)})
pcsSummary_df = pcsSummary_df.transpose()
pcsSummary_df.columns = ['PC{}'.format(i) for i in range(1, len(pcsSummary_df.columns) + 1)]
pcsSummary_df.round(4)
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13
Standard deviation	83.7641	70.9143	22.6437	19.1815	8.4232	2.0917	1.6994	0.7796	0.6578	0.3704	0.1864	0.063	0.0
Proportion of variance	0.5395	0.3867	0.0394	0.0283	0.0055	0.0003	0.0002	0.0000	0.0000	0.0000	0.0000	0.000	0.0
Cumulative proportion	0.5395	0.9262	0.9656	0.9939	0.9993	0.9997	0.9999	1.0000	1.0000	1.0000	1.0000	1.000	1.0

주성분 분석(PCA)

```
# 데이터 세트의 주성분을 포함하는 pcsComponents_df라는 판다스 데이터 프레임을 생성
# pcs.components_ 속성에는 각각 원래 변수의 선형 조합인 주성분이 포함되어 있음
# 속성은 (n_components, n_features)의 형태를 가짐. n_components는 주성분의 수이고 n_features는 원래 변수의 수
# (이 코드에서는 구성 요소 수 지정 안함)
# 기본 형태를 전치하는 .transpose() 메서드를 사용하여 pcs.components_ 행렬을 전치시켜서 표현.
# 전치된 행렬을 기준으로 DataFrame을 생성하고 pcsSummary_df.columns 속성을 사용하여 열 이름을 할당
# 원래 데이터 세트인 cereals_df.iloc[:, 3:].columns의 열 이름을 사용하여 DataFrame에 행 레이블을 할당
# PC 중 처음부터 5개만 보여줌
pcsComponents_df = pd.DataFrame(pcs.components_.transpose(), columns=pcsSummary_df.columns,
                                index=cereals_df.iloc[:, 3:].columns)

pcsComponents_df.iloc[:, :5]
```

	PC1	PC2	PC3	PC4	PC5
calories	-0.077984	-0.009312	0.629206	-0.601021	0.454959
protein	0.000757	0.008801	0.001026	0.003200	0.056176
fat	0.000102	0.002699	0.016196	-0.025262	-0.016098
sodium	-0.980215	0.140896	-0.135902	-0.000968	0.013948
fiber	0.005413	0.030681	-0.018191	0.020472	0.013605
carbo	-0.017246	-0.016783	0.017370	0.025948	0.349267
sugars	-0.002989	-0.000253	0.097705	-0.115481	-0.299066
potass	0.134900	0.986562	0.036782	-0.042176	-0.047151
vitamins	-0.094293	0.016729	0.691978	0.714118	-0.037009
shelf	0.001541	0.004360	0.012489	0.005647	-0.007876
weight	-0.000512	0.000999	0.003806	-0.002546	0.003022
cups	-0.000510	-0.001591	0.000694	0.000985	0.002148
rating	0.075296	0.071742	-0.307947	0.334534	0.757708

주성분 분석(PCA) : 정규화

■ 정규화

- 이 예에서 나트륨이 첫 번째 주성분에 지배적
- 정규화(또는 표준화)는 각각의 원래 변수를 분산이 1인 표준화된 변수로 대체하는 것으로, 정규화(표준화)를 통해 모든 변수에 동등한 중요성 부여

주성분 분석(PCA) : 정규화

■ 정규화

- 정규화를 위해 preprocessing.scale 활용
- 총 변동 90% 이상을 설명하기 위해서는 7개의 주성분이 필요함
- 첫 두개의 주성분이 총 변동의 52% 설명 → 2개로 축소하면 많은 정보를 손실할 가능성 큼

```
# scikit-learn의 전처리 기능을 사용하여 PCA 전에 데이터 표준화
pcs = PCA()
pcs.fit(preprocessing.scale(cereals_df.iloc[:, 3:]).dropna(axis=0))
pcsSummary_df = pd.DataFrame({'Standard deviation': np.sqrt(pcs.explained_variance_),
                              'Proportion of variance': pcs.explained_variance_ratio_,
                              'Cumulative proportion': np.cumsum(pcs.explained_variance_ratio_)})
pcsSummary_df = pcsSummary_df.transpose()
pcsSummary_df.columns = ['PC{}'.format(i) for i in range(1, len(pcsSummary_df.columns) + 1)]
pcsSummary_df.round(4)
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11	PC12	PC13
Standard deviation	1.9192	1.7864	1.3912	1.0166	1.0015	0.8555	0.8251	0.6496	0.5658	0.3051	0.2537	0.1399	0.0
Proportion of variance	0.2795	0.2422	0.1469	0.0784	0.0761	0.0555	0.0517	0.0320	0.0243	0.0071	0.0049	0.0015	0.0
Cumulative proportion	0.2795	0.5217	0.6685	0.7470	0.8231	0.8786	0.9303	0.9623	0.9866	0.9936	0.9985	1.0000	1.0

주성분 분석(PCA) : 정규화

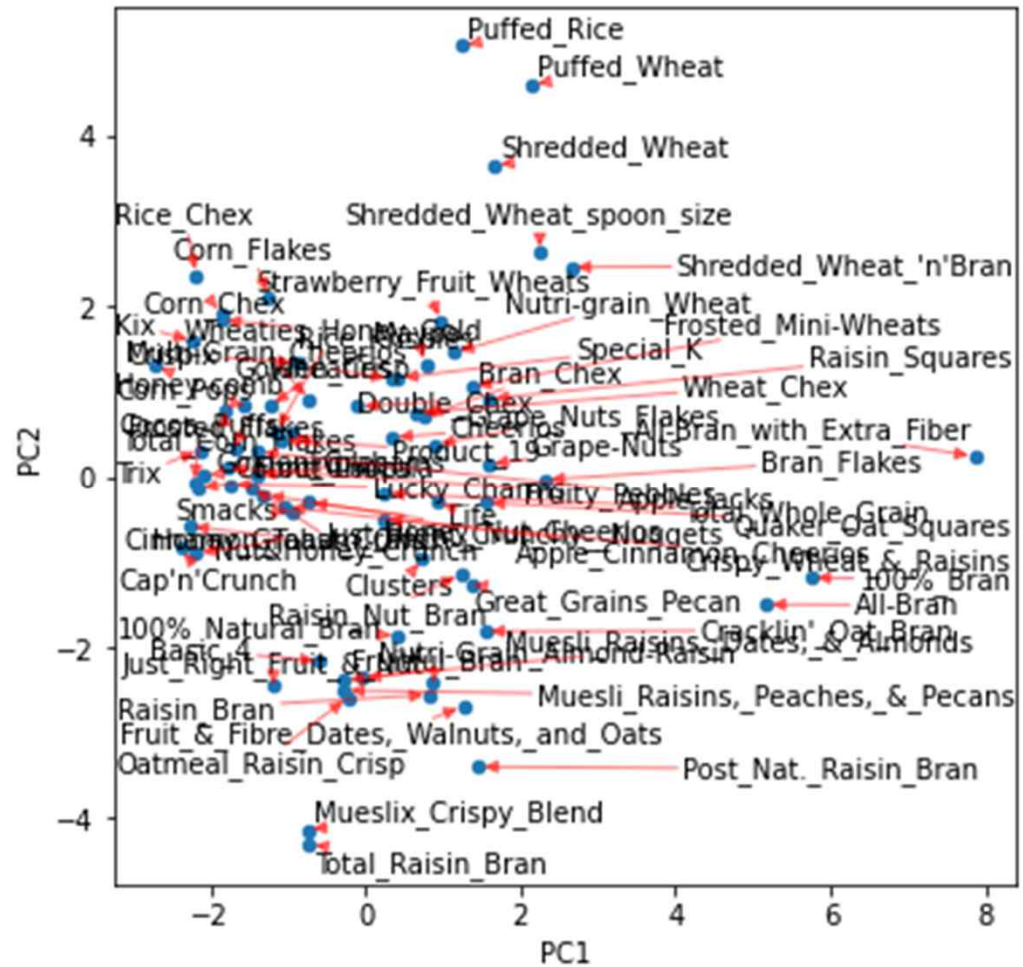
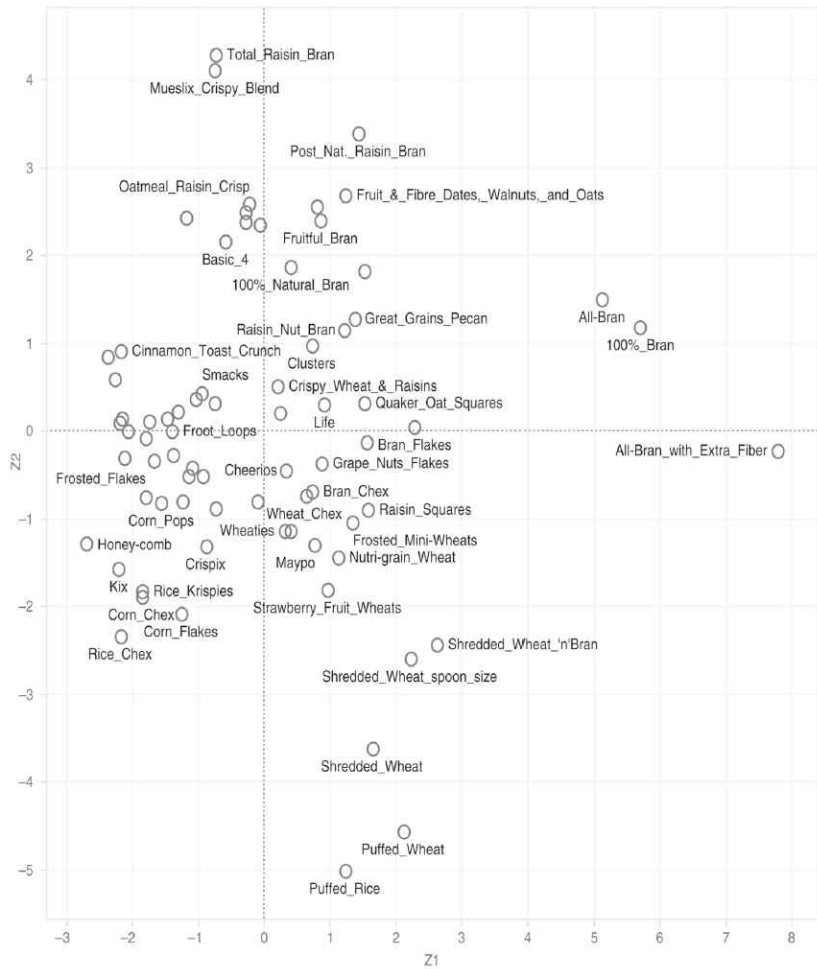
■ 정규화

- Pos/Neg 별 가중치 해석

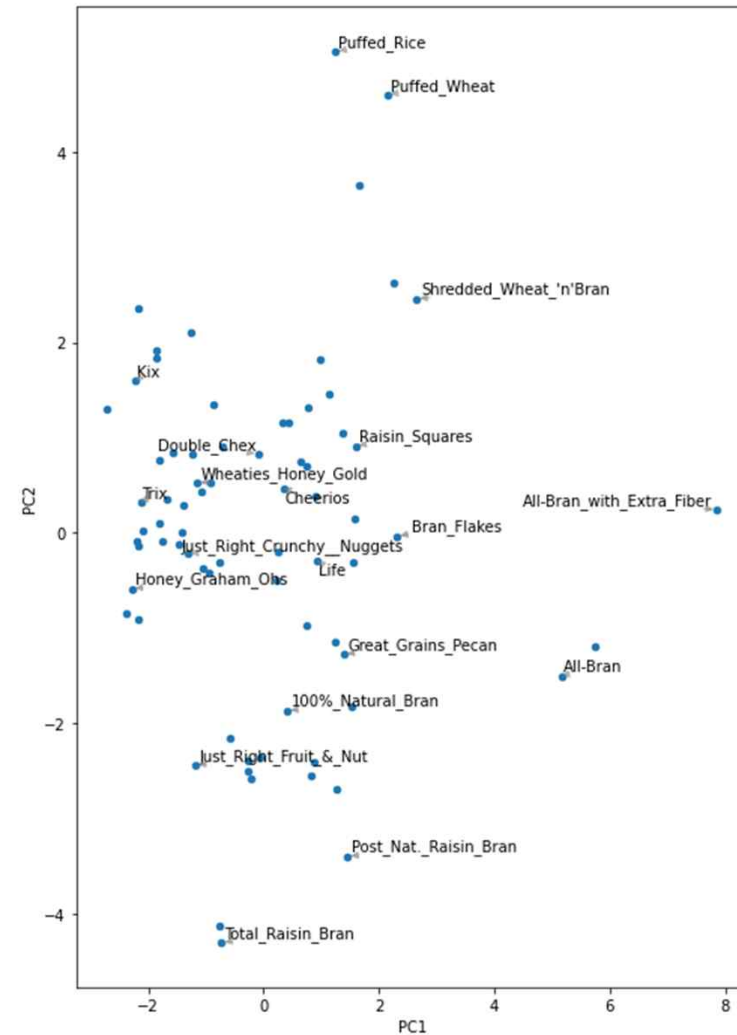
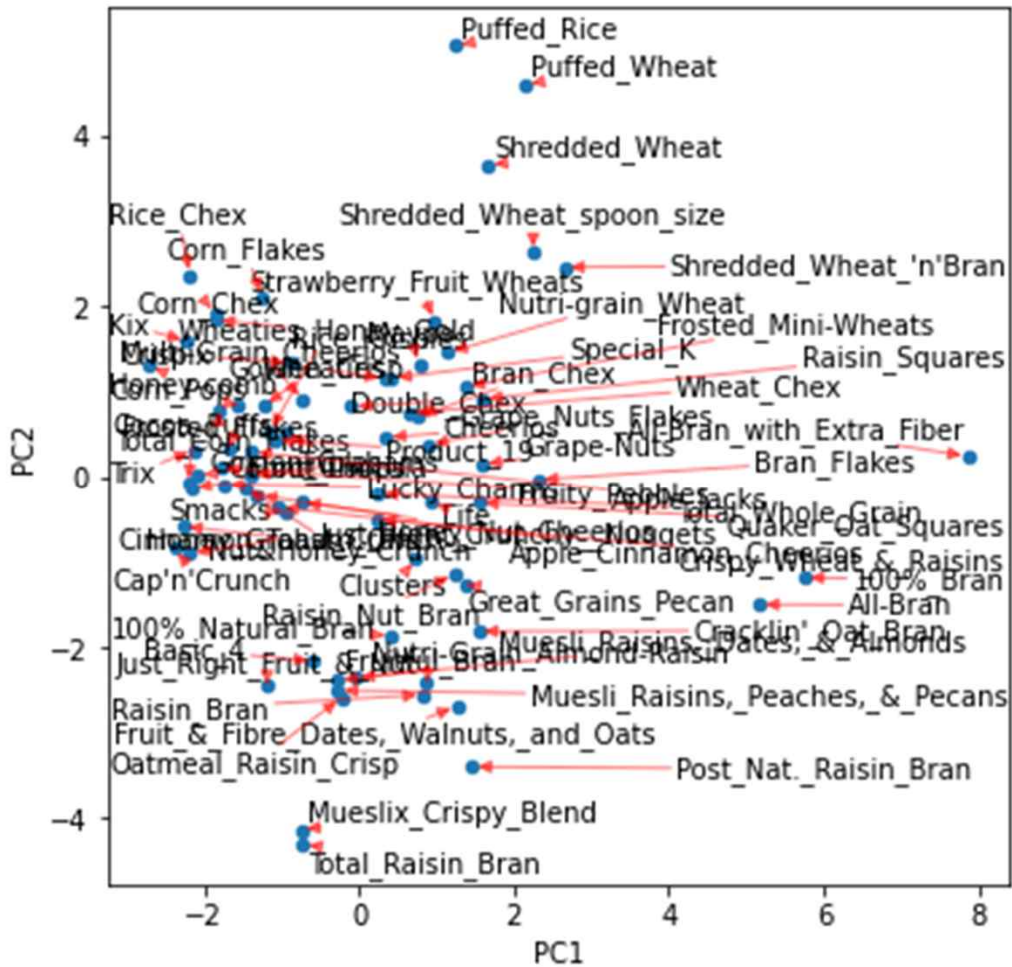
```
pcsComponents_df = pd.DataFrame(pcs.components_.transpose(), columns=pcsSummary_df.columns,  
                                index=cereals_df.iloc[:, 3:].columns)  
pcsComponents_df.iloc[:, :7]
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7
calories	-0.299542	-0.393148	0.114857	-0.204359	0.203899	0.255906	0.025595
protein	0.307356	-0.165323	0.277282	-0.300743	0.319749	-0.120752	-0.282705
fat	-0.039915	-0.345724	-0.204890	-0.186833	0.586893	-0.347967	0.051155
sodium	-0.183397	-0.137221	0.389431	-0.120337	-0.338364	-0.664372	0.283703
fiber	0.453490	-0.179812	0.069766	-0.039174	-0.255119	-0.064244	-0.112325
carbo	-0.192449	0.149448	0.562452	-0.087835	0.182743	0.326393	0.260468
sugars	-0.228068	-0.351434	-0.355405	0.022707	-0.314872	0.152082	-0.227985
potass	0.401964	-0.300544	0.067620	-0.090878	-0.148360	-0.025154	-0.148808
vitamins	-0.115980	-0.172909	0.387859	0.604111	-0.049287	-0.129486	-0.294276
shelf	0.171263	-0.265050	-0.001531	0.638879	0.329101	0.052044	0.174834
weight	-0.050295	-0.450309	0.247138	-0.153429	-0.221283	0.398774	-0.013921
cups	-0.294636	0.212248	0.140000	-0.047489	0.120816	-0.099461	-0.748567
rating	0.438378	0.251539	0.181842	-0.038316	0.057584	0.186145	-0.063445

주성분 분석(PCA) : 시각화



주성분 분석(PCA) : 시각화



주성분 분석(PCA) : 분류와 예측에 사용

- 학습 데이터를 사용해 예측 변수에 PCA 적용
- 분석 결과로부터 예측 변수로 사용할 주성분의 개수 결정
- 모델에서 (축소된 개수의) 주성분 점수를 예측 변수로 사용
- 검증 데이터셋은 학습 데이터로부터 계산된 가중치를 변수들에 적용함으로써 주성분 점수를 얻어 예측 변수로 사용

회귀 모델/분류 트리와 회귀 트리를 사용한 차원 축소

■ 회귀 모델을 사용한 차원 축소

- 예측 변수를 줄이기 위해 변수 선택 절차 사용
 - 선형 회귀 모델을 사용한 예측
 - 로지스틱 회귀 모델을 사용한 분류
- 회귀 모델을 사용한 유사 범주 결합

■ 분류 트리와 회귀 트리를 이용한 차원 축소

- 트리 차트에 나타나지 않은 (수치형 또는 범주형) 예측 변수들은 제거하거나 결합

Q & A

