

# 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 <sup>town</sup> 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)
```

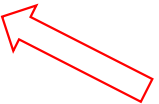
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	LSTAT	MEDV	CAT_MEDV	RM_bin
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	4.98	24.0	0	6
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	9.14	21.6	0	6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	4.03	34.7	1	7
3	0.03237	0.0	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
# # column = '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.
```

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

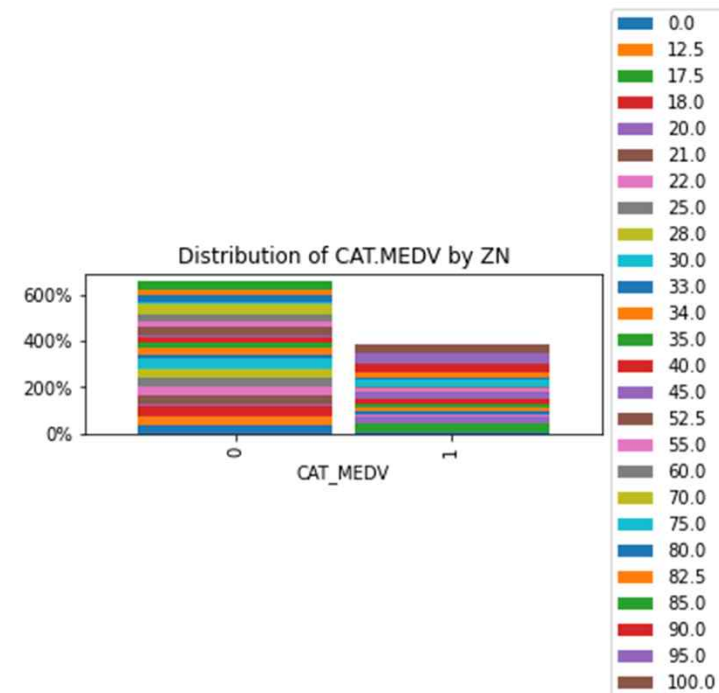


# 범주 축소(1)

## ■ 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.
```

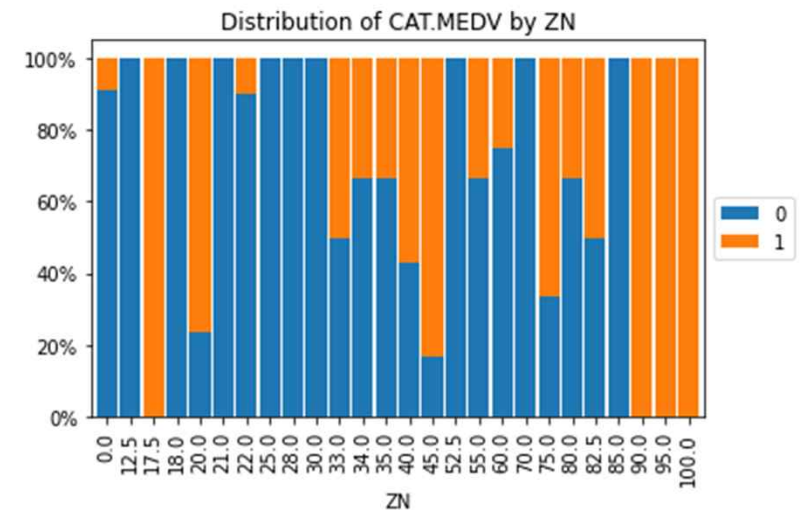


# 범주 축소(1)

## ■ 교차 분석표 (Crosstab)

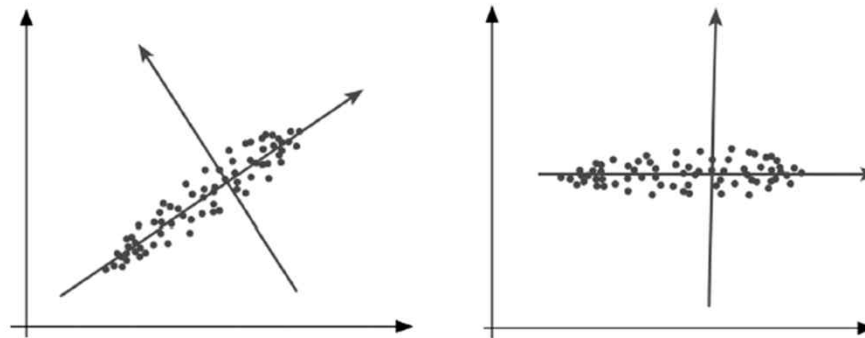
- Pandas에서 범주형 데이터 2개를 비교 분석 할 때 유용한 표

```
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)

