In [1]:

```
02.01 데이터 전처리 기초

[전처리 사용 패키지]

missingno : 결측 데이터 검색
sklearn.impute : 결측 데이터 대체
patsy : 데이터 선택, 변환, 추가, 스케일링
sklearn.preprocessing : 스케일링, 변환


1. 결측 데이터 (msno, sklearn.impute)

1) missingno 패키지
결측 검색 대상 : pandas 데이터프레임 상 NaN(not a number)

주의사항)
    NaN 값 : 부동소수점 실수 자료형에만 존재
        (원래 정수형, 시간자료형에는 존재하지 않음)

    1. 데이터프레임 속 정수값 : Int64Dtye자료형을 명시
    2. 데이터프레임 속 시간값 : parse_dates로 날짜시간형 파싱!
        *그래야 datetime64[ns]자료형이 되어 결측데이터가 NaT(not a time)값이 됨


1. 결측데이터 위치 탐색
df.isnull()
df.isnull().sum()
msno.matrix(df)
msno.bar(df)


2. 결측데이터 대체 (삭제 OR 대체)
- 결측 데이터가 너무 많은 경우 : 해당 데이터 열 전체르 삭제가능 (가짜 데이터 많이 채우기보단..)
- 결측 데이터가 일부인 경우 : 가장 그럴듯한 값으로 대체(imputation)

[삭제]
df.dropna(axis=1) *axis=1, 열단위 결측데이터 존재 열 삭제(행단위 axis=0)
df.dropna(axis=1, thresh=7), 열단위 결측데이터 존재 열 삭제(7개 이상 비결측 값 있으면 남기, th
resh)

[대체]
sklearn 패키지 - SimpleImputer 패키지 ==>> fit_transform 메서드

ex)
from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy = 'most_frequent')
df = pd.DataFrame(imputer.fit_transform(df), columns = df.columns)

*strategy인수 : mean, median, most_frequent
    mean : 데이터가 실수 연속값인 경우, 값의 분포가 대칭적일 때
    median : 데이터가 실수 연속값인 경우, 값의 분포가 비챙적인 경우
    most_frequent : 데이터가 범주값, 정수값인 경우 (가장 많이 나온값으로 채우면 그래도 가장 많이 맞
을 수 있다.)




2. 데이터 가공 (patsy)
```

1) patsy패키지

*design matrix = dmatrix
*formula = 데이터 열 이름 기반으로 구성된 문자열 (1열 + 2열 + 0 = 상수항 생성하지말고 1열과 2열만 가져와라)

[데이터 선택]
dmatrix(formula문자열, data)

[데이터 변형]
dmatrix(formula문자열, data)
*formula 문자열 = 수식을 포함해, 원하는 컬럼대로 기입 (def로 만든 함수등 파이썬이 해석 가능한 모든 명령 문자열로 기입)

ex) dmatrix("x1 + np.log(np.abs(x2))",df)
ex) dmatrix("x1 + x2 + x1:x2", df)
    *(x1:x2 = x1과 x2를 곱한 값 = 상호작용항. 회귀분석 시 반드시 체크해야함)
    *각각 들어갔을 때는 없었던 효과가, 상호작용에서는 보이는 경우가 있기 때문에 반드시 체크!
ex) dmatrix("x1 + x2 + I(x1+x2) + I(x1/x2) + 0", df)
    *상호작용항 제외, 연산값 컬럼을 만들고 싶다면, I() 연산자 활용
    *맨 끝에 +0 은 상수항 만들지 말라는 의미 (상수항은 회귀분석 용 데이터 시 필수이긴 함)

*스케일링 작업 = 선형회기 분석 시, 조건수(condition number)의 영향으로 데이터의 다중공선성이 있을 수 있음
    ==> 이를 잡기 위해 스케일링 작업을 해줘야 함 (일종의 표준정규분포로 정규화하는 작업)
    ==> 평균 0, 표준편차 1로 변환

center(): 평균을 0으로 스케일링
standardize(): 평균을 0으로하고 표준편차를 1로 스케일링
scale(): standardize() 과 같음

ex) dmatrix("center(x1) + 0", df)
    *x1 - mean of x1 = center(x1)
    *center(x1)의 값들은 dm.design_info.factors_infors 내부에 저장됨
    *왜 스케일링 값을 저장하는 가? (18p)

3. 데이터 가공 (sklearn.preprocessing)

1) sklearn.preprocessing
    - StandardScalar 클래스
        scaler = StandardScaler()
        scaler.fit_transform(x)

    - RobustScalar 클래스
        scaler = RobustScaler
        scaler.fit_transform(x)
    - 19p

    - PolynomialFeatures 클래스(21p)

```
- FunctionTransformer 클래스(21,22p)
```

```
  File "<ipython-input-1-adda786f2d64>", line 1
    02.01 데이터 전처리 기초
          ^
SyntaxError: invalid syntax
```

In [ ]:

02.02 범주형 데이터 처리

*실수형 데이터 = 연속적, 크기 및 가치비교가 가능

*대부분의 데이터 분석 모형 = 숫자만 입력으로 받을 수 있음. 따라서, 범주형 데이터는 숫자로 변환해야함
*범주형 데이터의 변형
    1. 더미변수화
    2. 카테고리 임베딩

1. 더미변수화 (풀랭크 / 축소랭크)

1) 더미변수
    더미변수 : 0, 1 (이진 변수), 특징 유무만 표현 ex)남자/여자
    더미변수의 갯수 = 클래스의 갯수 (2p, 혈액형)

2) (풀랭크) patsy 패키지를 사용한 더미변수화 (dmatrix)
    *주의사항 : + 0을 추가하면, 풀랭크 방식으로 더미변수 생성
    *dmatrix("x + 0", df)
    *dmatrix("C(x) + 0",df)
        C연산자 => 1. 데이터가 범주형 값이지만, 정수로 표시된 경우, 범주형 값임을 명시적으로 지정
        C연산자 => 2. 더미변수의 순서도 바꿀 수 있음

3) (축소랭크) 더미변수화
    *주의사항 : +0이 없음
    *하나의 범주값을 기준값으로 지정(항상 1), 추가적인 특성을 나타내는 더미변수 1 출력
    *기준이 되는 더미변수는 이름이 'Intercept'가 됨. (5p)

4) 두개의 범주형 변수가 있는 경우

    - 통합 축소형 방식
    - 상호작용 방식

    1. 통합축소 방식(7p)

    2. 상호작용 방식(7p) -> 카테고리 변수 생성

2. 카테고리 임베딩
    - 범주값 대신 범주값의 특성을 나타내는 연속값/연속값 벡터를 사용
    - 운동선수의 이름을 나타내는 범주값 -> 나이, 연봉, 신체능력치 등으로 대체

- 지역명을 나타내는 범주값 -> 지역의 면적, 인구수 등으로 대체

In [ ]:

04.01 회귀분석 예제

make_regression 예제(4.1.2)
- 가상의 회귀분석 문제를 만들어줌

- 2개의 독립변수가 있다면, 1개만 종속변수와 상관관계 갖도록 설계 가능
- 두 변수가 독립이 아닌 다중공산성 설계 가능

make_regression() 명령은 내부적으로 다음 과정을 거쳐 가상의 데이터를 만듬

1. 독립변수 데이터 행렬 X를 무작위로 만든다.
2. 종속변수와 독립변수를 연결하는 가중치 벡터 w를 무작위로 만든다.
3. X와 w를 내적하고 y절편 b 값을 더하여 독립변수와 완전선형인 종속변수 벡터 y_0를 만든다.
4. 기댓값이 0이고 표준편차가 noise인 정규분포를 이용하여 잡음 epsilon를 만든다.
5. 독립변수와 완전선형인 종속변수 벡터 y_0에 잡음 epsilon을 더해서 종속변수 데이터 y 를 만든다.

$$y = wTx + b + \epsilon$$

In [ ]:

04.02 선형회귀분석의 기초

회귀분석 : 종속변수의 값을 실제값과 가장 비슷하게 출력하는 f(x)함수를 찾는 과정
*f(x)가 선형이면, 선형회귀분석이다.

- f(x)의 계수 = 가중치 벡터 = 선형회귀모형의 모수(parameter) (1p 상단)

1) 상수항 결합
- y = w0.Tx0, 상수항도 포함해 내적으로 선형회귀분석 식을 간단히 하기 위함
- statsmodels패키지 - add_constant 함수
- 1p, 2p 상단

2) 최소자승법 (OLS, Ordinary Least Square)
- 잔차제곱합(RSS)를 최소화하는 가중치 벡터를 구하는 방법
*벡터의 크기 = L2norm으로. 잔차벡터의 norm최소화
= norm은 각 원소들의 제곱 합에 squared = RSS도출

3) 직교방정식 (normal equation) (3p 하단)
- 그레디언트가 0벡터가 되는 관계를 나타냄

- 2가지 성질 (4p)
1. 모형에 상수항 존재 시, SUM(잔차벡터의 원소) = 0 (잔차 평균은 0)
2. x 데이터의 평균값에 대한 예측 = y 데이터의 평균값

4) Numpy를 이용한 선형 회귀분석
- sklearn의 make_retression 으로 선형회귀 모델, 데이터 생성
- np.linalg.inv(넘파이 선형대수 기능)으로 OLS해를 직접 구하기
-> 잡음 때문에 같지는 않지만 비슷한 가중치 벡터 구할 수 있음

5) scikit-learn 패키지를 이용한 선형회귀분석

6) statsmodels 패키지를 이용한 선형회귀분석 (주로 이걸 사용, 8p)