머신 러닝 노트 3/17~3/18

1. **선형 회귀 (Linear regression): 독립 변수와 종속 변수 간의 관계를 조사하는 예측 분석의 한 형태. 데이터 세트의 모양이 직선과 가장 비슷할 때 사용.**

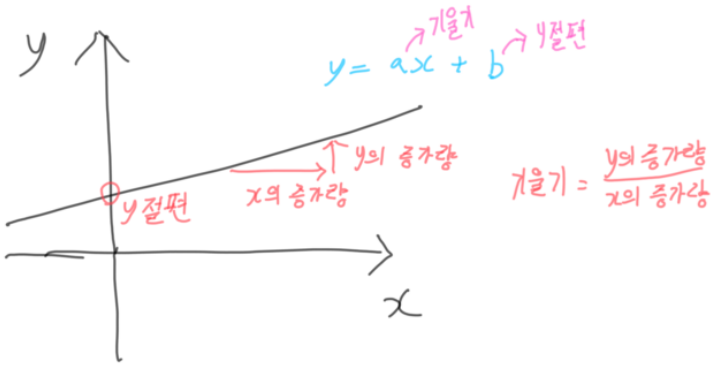
   (변수 관계를 선 그을 수 있다=상관있다=학습가능하다)

   Linear: 데이터를 잘 표현하는 1차 함수, **y=Wx+b**

   regression: 독립변수(x)로 종속변수(y)를 예측

   W: weight(가중치), 직선의 기울기(slope)

   b: bias(편향), x=0 이면 y=b



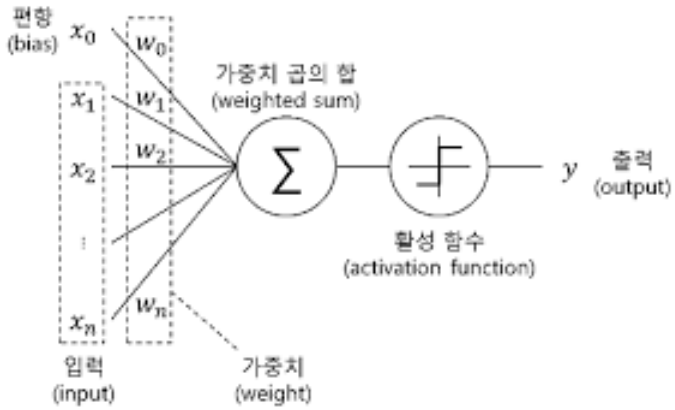

```
y_hat=w*x +b #1차함수
```

2. **2개 데이터 세트의 선형 방정식: y=w1x1+w2x2+b**

$$y = w_1 x_1 + w_2 x_2 + b$$

```
y = x1*a1 + x2*a2 +b #모델

x1=np.random.randint(1,100,size=1000) #(식에 있는 x1값) 1부터 100사이 1000개를 생성
x2=np.random.randint(1,100,size=1000) #(식에 있는 x2값) 1부터 100사이 1000개를 생성
a1=np.random.randint(1,10) #(식에 있는 a1값)
```

```
a2=np.random.randint(-10,2) #(식에 있는 a2값)
b=np.random.randint(1,6) #(식에 있는 b값)
```



n개 데이터 세트의 선형 방정식: y=w0x0+w1x1+w2x2+...+wnxn+b

3. **손실 함수 (MSE:** Mean Square Error**): 현재 모델과 데이터 세트의 실제 지점 사이의 오류를 계산, 이 값을 최소화 시켜야 함. 모델의 성능이 얼마나 떨어지는 지 알려줌.**

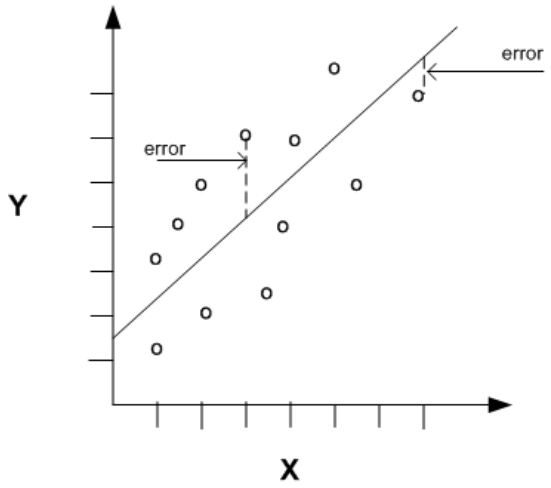
   cost(loss)function=[{(Y실제 값-ŷ (y hat)예측 값)^2음수제거위함}/n총 데이터]

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

제곱합을 데이터 세트의 요소 수로 나누기: 는 기계 학습 규칙으로 인해 수행

💡 변수A=(np.sum((y-y_hat)**2))/y

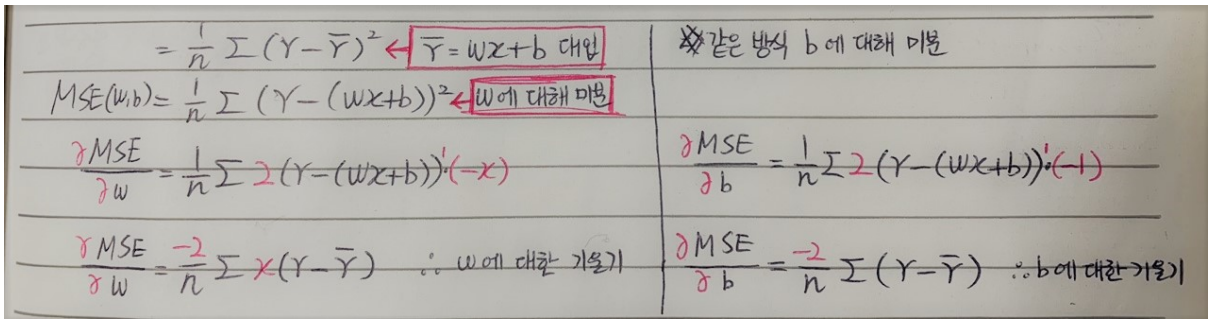
```
...
err=[]
for a in np.arange(-5,13): #start부터 stop 미만까지 step 간격으로 데이터 생성한 후 배열
    y_hat=a*x #위에서 준 식
    tmp=(np.sum((y-y_hat)**2))/y.size #Mean Square Error
    err.append(tmp)
err
```



4. **경사 하강 법 Gradient Descent (GD): 미분 가능한 함수의 최소값을 찾는 반복 최적화 알고리즘, MSE 손실 함수에서 최소값을 찾음, 2차 함수 그래프. 경사 하강 법을 통해 인치 단위로 이동.**

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$$

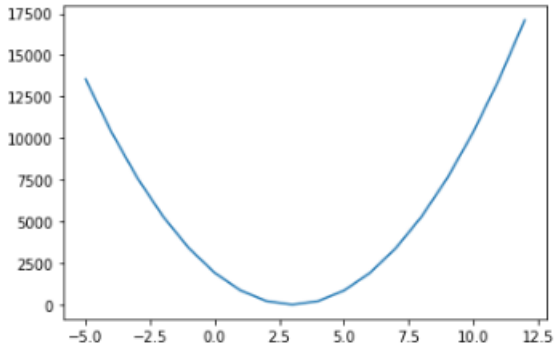
$$MSE(w,b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - (wx_i + b))^2$$



도함수를 사용하면 임의의 순간 지점에서 비선형 그래프의 기울기를 찾을 수 있다. 편미분.
MSE 함수 최소값=**기울기가 0 인** 위치. 단순 참고용입니다. 그냥 넘어가도 됨.

```
a=np.arange(-5,13) ## 1차원 배열
plt.plot(a,err) #경사하강: 스스로 학습가능
```

```
[<matplotlib.lines.Line2D at 0x7f5d8a22cee0>]
```

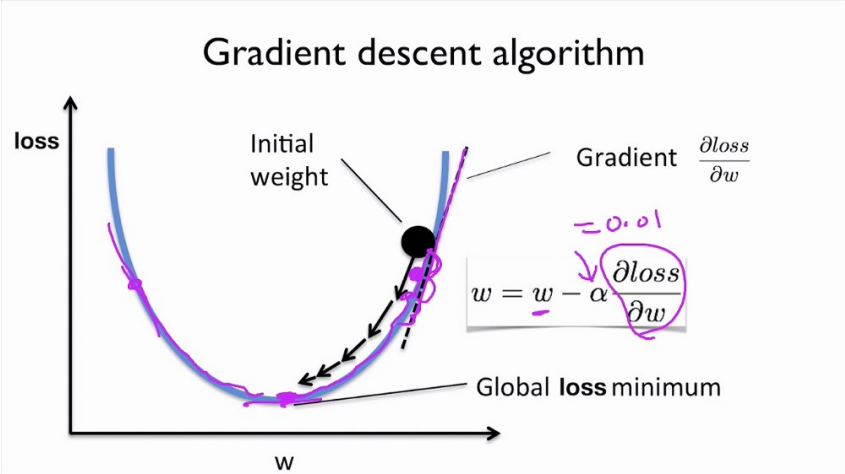


5. **가중치w 및 편향b을 조절: 회귀 모델이 가중치와 편향을 조정할 때 초과하지 않고 더 많은 오류를 생성하지 않도록 작은 숫자 (예 : 0.001)로 정의, (기울기 곱하기 L)로 w와 b을 조정**

   learnning rate(lr, L, Step size): 적당한 보폭의 스칼라값을 곱 다음 지점***을*** 결정.

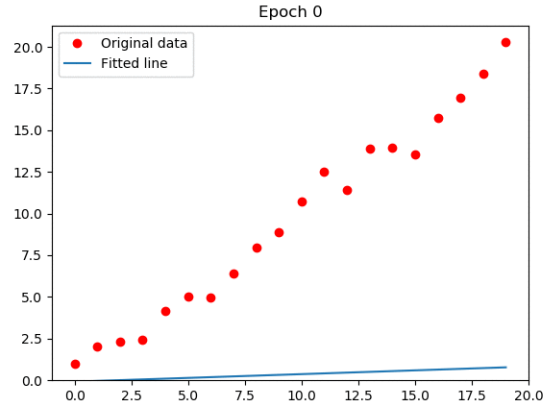
$$w = w - \frac{\partial MSE}{\partial w} * L$$

$$b = b - \frac{\partial MSE}{\partial b} * L$$




```
lr=0.00001
_a = _a - lr*((_a*x-y)*x).mean()
```

6. **Epoch: 전체 데이터 세트를 반복하고 손실 함수를 계산하며 경사 하강 법을 수행하는 기계 학습. 경사하강법은 인치 단위로 이동해서 눈에 띄는 변화를 보려면 기계 학습을 여러번 돌려야 함.**



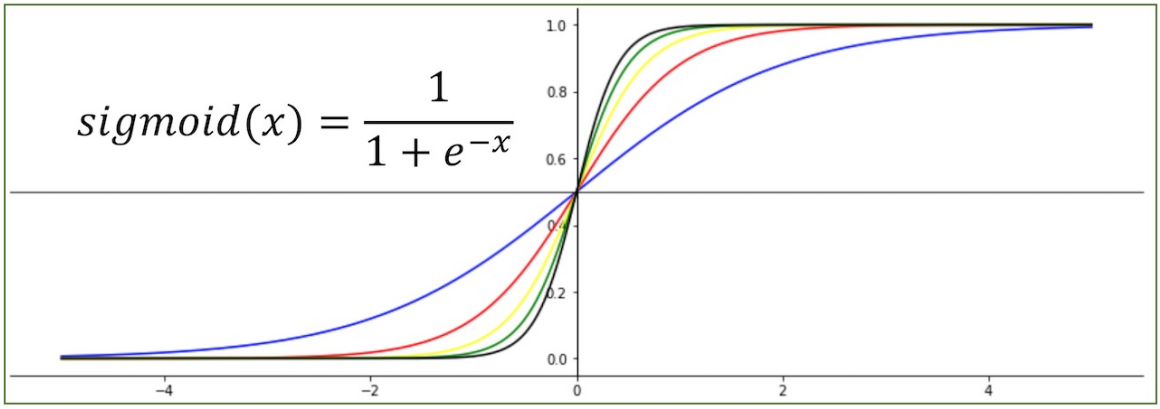

```
epoch = 10000 #학습 한번 하는 주기
for i in range(epoch) ...
```

```
y=x*a+b

x=np.random.randint(10,200,1000)
a=np.random.randint(-5,5)
b=np.random.randint(-5,5)

epoch = 10000
w=np.random.randint(-5,5)
_b=np.random.randint(-8,8)
lr=1e-5
err=[]
for i in range(epoch):
    y_hat=w*x +_b
    w = w-lr*((y_hat-y)*x).mean()
```

```
    _b = _b-lr*(y_hat-y).mean()
print(f"실제기울기는 {a} 이며 예측기울기 {w} \n 실제편차 {b} 예측편차 {_b}")
```

7. **Sigmoid 함수: 모든 입력에 대하여 S와 같은 형태로 미분 가능한 0~1 사이의 값을 반환, 분류 문제(Logistic Classification)의 가설과 비용 함수(Cost Function)에 많이 사용**

💡 step function의 경우는 보통 미분 불가능한 사각형 모양의 0 아니면 1 인 함수이지만, sigmoid 함수의 미분에서는 가운데가 무한대 까지 치솟는 델타함수가 됨



식 중요

```
import numpy as np
x=np.arange(1,100)
y=np.linspace(0,1,99) #두 수 사이를 50개의 균일한 간격의 수를 배열로 만들어준다

logi_x = 1/(1+np.exp(-x)) #밑이 자연상수 e인 지수함수(e^x)로 변환

import matplotlib.pyplot as plt
plt.plot(x,y)
plt.plot(logi_x,y)
```

8. **중심극한 정리: 모집단이 정규분포가 아니더라도 표본의 크기가 30이상이 되면 표본평균의 분포가 모집단의 분포와 상관없이 정규분포에 근사**

(어짜피 뽑을 때 가장 많이 모여있는 구간에서 뽑힐 확률이 높기 때문에 뽑힌 표본의 평균도 유사하다.)

💡 PPT 29 통계적 추론을 참고하시오.

9. **결정계수 r_squared** : **0<= R^2 <= 1:** R^2가 1이라면, 이는 **오차^2 = 0인 상황이므로 training error가 0**인 것을 의미

$$R^2 = 1 - \frac{\Sigma(\text{오차}^2)}{\Sigma(\text{편차}^2)}$$

$$\text{오차} = (t_i - y_i),\ t_i : \text{실제값},\ y_i : \text{예측값}$$

$$\text{편차} = (t_i - \overline{t_i}),\ t_i : \text{실제값},\ \overline{t_i} : \text{평균값}$$

```
오차의제곱합=np.sum((y-y_hat)**2)
편차의제곱합=np.sum((y-y.mean())**2)
r_squared=1-(오차의제곱합)/(편차의제곱합)
r_squared #이 값이 0.8이상 나와야함. 그래야 쓸 수 있는 데이터.
#값: -0.49724279238686075
```

10. **공분산: X의 편차와 Y의 편차를 곱한 것의 평균**

$$\text{분산} = \sigma^2 = \frac{\sum (X-\overline{X})^2}{n}$$

$$\text{공분산} = Cov[X, Y] = E[(X-\overline{X})(Y-\overline{Y})] = \sigma_{XY}$$

$$= \frac{\sum (X_i-\overline{X})(Y_i-\overline{Y})}{n}$$

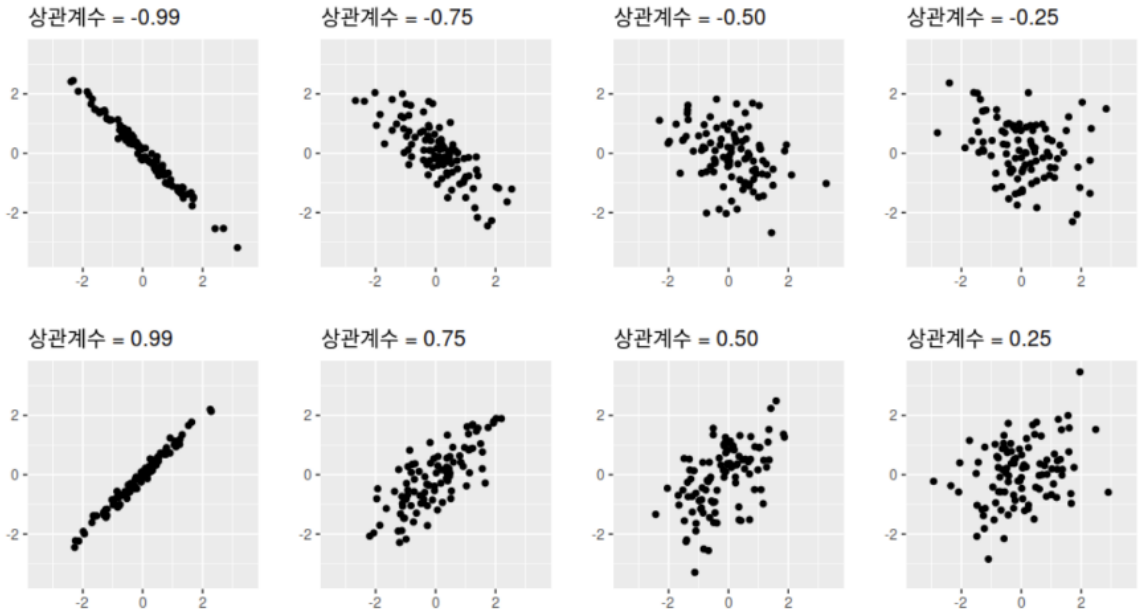
```
#minmax scaling #분산구하고 공분산 구하기

x1=np.random.randint(10000,100000,1000) #초기값
x2=np.random.randint(1,10,1000) #초기값

x1_var=np.sum((x1-x1.mean())**2)/len(x1) #분산
x2_var=np.sum((x2-x2.mean())**2)/len(x2) #분산

cov=np.sum(((x1-x1.mean())*(x2-x2.mean())))/len(x2) #공분산

x1_sc=(max(x1)-x1)/(max(x1)-min(x1)) #스케일링값
x2_sc=(max(x2)-x2)/(max(x2)-min(x2)) #스케일링값
cov_sc=np.sum(((x1_sc-x1_sc.mean())*(x2_sc-x2_sc.mean())))/len(x2_sc)
cov_sc
#값: 0.001085093085106383
```

11. **상관계수: 두 연속형 변수의 선형 상관관계 강도 측정, -1≤상관계수≤1**

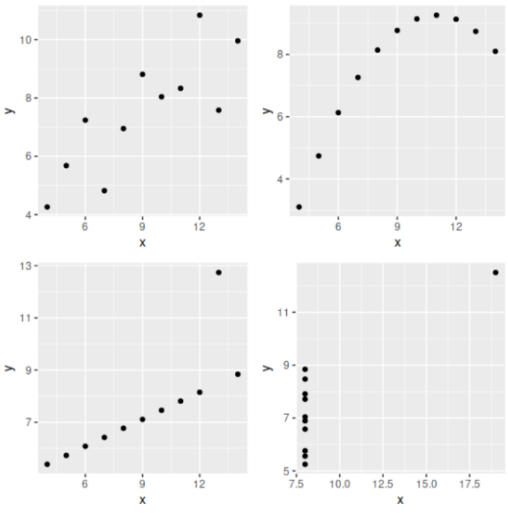
+1: 양의 선형 상관 관계

0: 상관 관계 없음

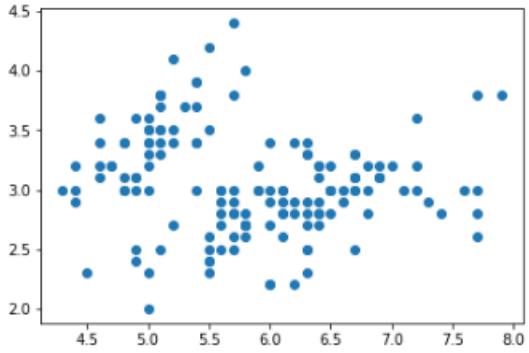
-1: 음의 선형 상관 관계



💡 참고용: 다음 그래프는 모두 상관계수 0.82이지만, 나타내는 관계는 다르다.

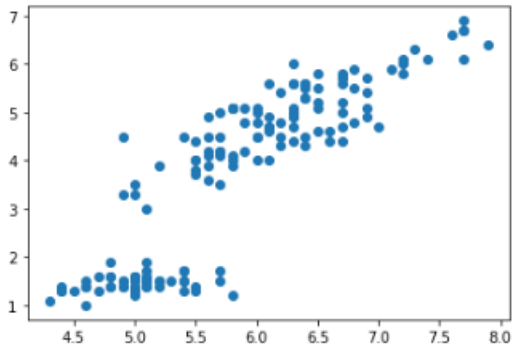


```
#iris데이터셋 분류하고 상관관계파악

from sklearn.datasets import load_iris
iris=load_iris()["data"]
plt.scatter(iris[:,0],iris[:,1])
```



상관관계 없음

```
from sklearn.datasets import load_iris
iris=load_iris()["data"]
plt.scatter(iris[:,0],iris[:,2])
```



상관관계 있음

상관계수P: 0.05(5%)와 같은 임계치를 정하고, p값이 그보다 작을 경우 "통계적으로 유의미하다"라고 함.

```
#cov상관계수 구하기
from sklearn.datasets import load_iris
data=load_iris()
x1=iris[:,0]
x2=iris[:,1]

x1=(x1-min(x1))/(max(x1)-min(x1))
x2=(x2-min(x2))/(max(x2)-min(x2))

np.sum(np.cov(x1,x2)/np.std(x1)*np.std(x2))
#값: 0.06005971071763204 #별로 유의미하지 않은 데이터
```

12. **분석방법**(ppt 39)

>숫자형변수

F-Test: 두 분산 간의 차이, 비율 검정

T-Test: 두 집단 간 평균의 차이가 있나 없나

ANOVA: 세 집단 이상 간 평균의 차이가 있나 없나

선형회귀: 변수 하나 일 때

다중회귀: 변수 여러 개 일 때

>범주형변수

카이제곱검정: 설명 시

로지스틱회귀: 분석 시

>>>ppt2장 데이터분석

13. Trend_데이터 분석

딥러닝: 깊이가 아니라 넓게 분석, 영향 미쳐도 쓸모없으면 빼야함, 단순한게 가장 좋음

데이터분석 중요5: Accuracy, Interpretable, Speed, Scalable, simple

14. data science

데이터는 빈 값 없이 무조건 다 차야 함

변하는 값 말고 고정 값 줌(나이 대신 생년월일)

없는 데이터는 행/열 날리고 모델링

```
#데이터 만들기
import pandas as pd
x=load_iris()['data'] #있는 데이터 중 꽃 데이터 불러옴
dt=pd.DataFrame(x)
dt.columns=load_iris()['feature_names']
dt["Species"]=np.where(load_iris()["target"]==0,"setosa",
            np.where(load_iris()["target"]==1,"vesicolor","verginica")) #일치하는 종 데이터만 들고오기

dt.to_csv("iris.csv") #csv파일로 저장

#데이터 정리 #독립변수에 이상이 있어서 날려버릴거임
data=pd.read_csv("iris.csv")
data=data.iloc[:,1:]
X=data.iloc[:,:-1]
y=data.iloc[:,-1]
X=X.values
```

```
y=y.values

#학습시키기
from sklearn.model_selection import train_test_split #x,y를 나눠주는애
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2)
from sklearn.svm import SVC #학습시키기
model=SVC()
model.fit(X_train,y_train) #학습끝

#학습잘됐는지확인
y_hat=model.predict(X_test) #예측값
sum(y_hat==y_test)/len(y_test) #몇개맞췄는지/나누기갯수 =퍼센트 #1이 나오면 다 맞음

from sklearn.tree import DecisionTreeClassifier #트리 분류 모델 형성에 각 특성이 얼마나 작용했는지 평가하는 지표
from sklearn.neighbors import KNeighborsClassifier #가장 가까운 훈련 데이터 포인트 하나를 최근접 이웃으로 찾아 예측에 사용
dt_model=DecisionTreeClassifier()
dt_model.fit(X_train,y_train)

knn=KNeighborsClassifier()
knn.fit(X_train, y_train)

dt_model.score(X_test,y_test) #2개를 하나로 만들어줌
```

15. 기계학습ML: 가설을 세우고/ 가설에 맞는 함수를 컴퓨터 스스로 만들어내는 것, 사람방식학습, 학습 결과를 설명 가능(결과의 원인을 설명할 수 있다면 결과도 바꿀 수 있음)
16. AI: 규칙기반 전문가 시스템
17. 지도학습/비지도학습/강화학습: clustering is good for everything

```
#클러스터링
from sklearn.cluster import KMeans

data = pd.read_csv("iris.csv")
data=data.iloc[:,1:] #앞에거만 들고오기
data=data.iloc[:,:-1]

x=data["sepal length (cm)"]  #변수 하나만 들고오기

kmeans = KMeans(4)

#x=x.values.reshape(-1,1)
#x.shape

kmeans.fit(x)

kmeans.labels_ #그룹화
```

18. 데이터마이닝: 데이터 안 규칙/패턴 찾아내는 것, 가설이 맞나 틀리나 검증하면 됨
19. 6myths: o,o,x,x,?,x
20. 빅데이터 알고리즘 연결: 가공된 데이터가 분석의 성패를 좌우(데이터 모델링 관점에 있어 도메인(Domain)은 데이터 표준화 측면에서 매우 중요한 요소)