

Supervised Learning



AI명예학회

SKHU

지도 학습?

Given $(x_1, y_1), (x_2, y_2), \dots, (x_i, y_i)$,
learn a function $f(x)$ to predict y
given x

함수 $f(x)$ 를 학습 하는 것.

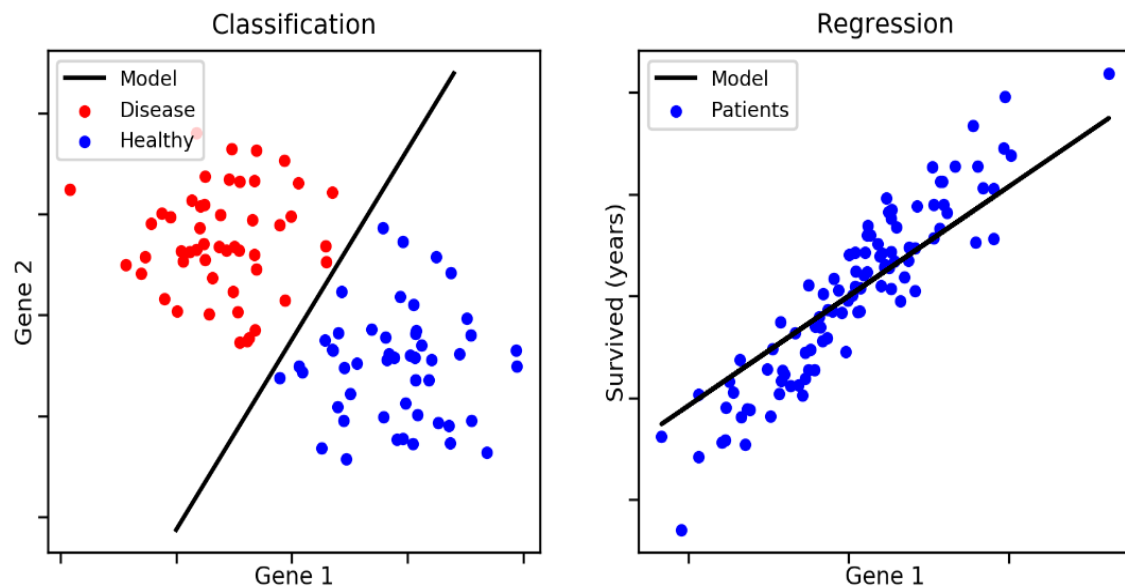
한 데이터에 (x, y) 로 구성.

각각 한 쌍으로 구성되고,

x 가 변수, y 가 정답.

함수의 형태는 어떤 형식으로든
구성. (x^2, x^3 등등)

회귀와 분류의 직관적 이해



classification:

이산적인 범주나 클래스를 예측하는 문제, 확률 모델을 사용해 각 클래스에 대한 확률을 계산하여 예측
ex) 스팸 메시지

Regression:

연속적인 수치 값을 예측하는 문제
ex) 주택 가격 예측

회귀(Regression)

단순회귀식

$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X$$

다중회귀식

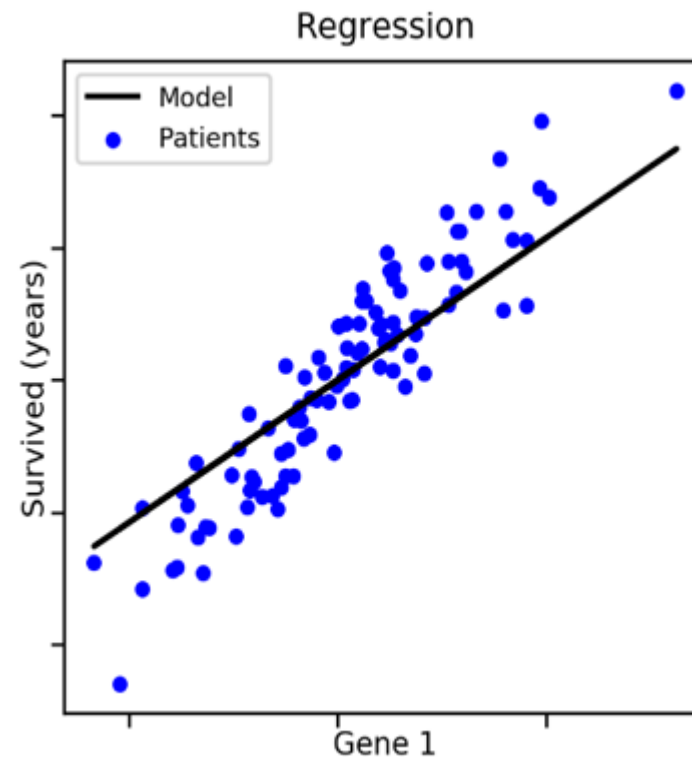
$$\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \cdots + \hat{\beta}_k X_k$$

입력 X 를 받아서 출력 Y (연속적)를 예측하는 것.

여기서 $\hat{\beta}_0$ 는 절편값.

$\hat{\beta}_k$ 는 가중치(파라미터 값)

예측값과 실제값 차이를 최소화 하는 것이 관건.



$y \in \mathbb{R}$ (실수 값)

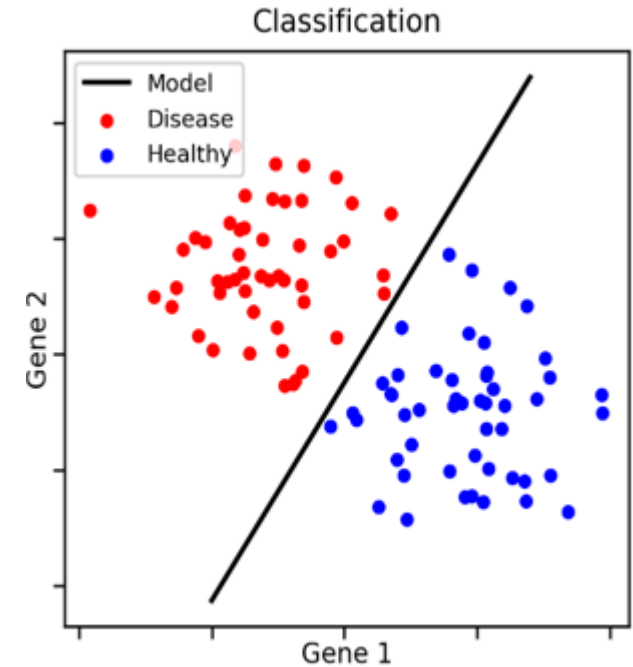
연속적인 수치 값을 예측하는 것

분류(classification)

$$P(y = C_k | \mathbf{x}) = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + \dots + w_n x_n + b)}} \quad P(y = 1 | \mathbf{x}) = \frac{1}{1 + e^{-z}}$$

입력 X 받아서 Y값(이산적) 예측 하는 것.

$y \in \{C_1, C_2, \dots, C_k\}$ (이산적인 클래스 값)



이산적인 범주(클래스) 예측 문제.
확률을 계산하여 예측

회귀(Regression)

-Linear Regression

-Ridge/Lasso

-Polynomial Regression

등등

회귀(Regression)

- Linear Regression

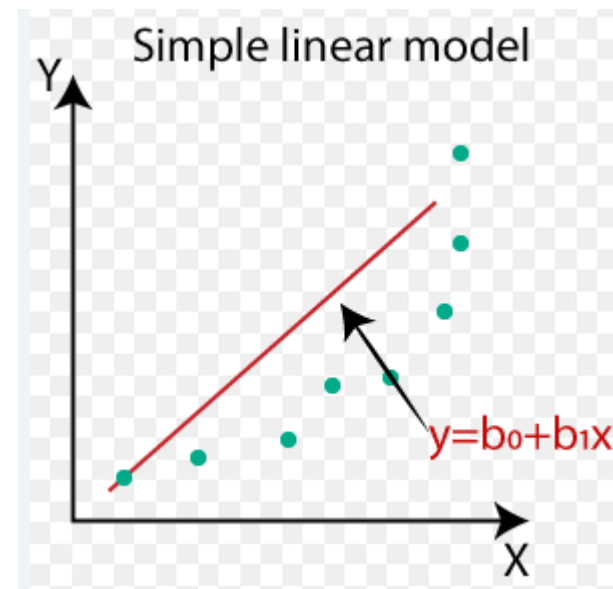
$$y = w_1x_1 + w_2x_2 + \dots + w_nx_n + b = \mathbf{w}^T \mathbf{x} + b$$

y: 출력값

x = (x_1, x_2, ..., x_n) # 입력 변수

w = (w_1, w_2, ..., w_n) # 입력변수의 가중치

b: 절편값(바이어스)



x값은 데이터에 따라 정해져 있고,

결국 w값(파라미터)을 맞추는게
관건

회귀(Regression)

- Linear Regression

손실함수(loss function) : 예측한 값이랑 실제 값 사이의 차이를 수치적으로 알 수 있게 해주는

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - (\mathbf{w}^T \mathbf{x}_i + b))^2$$

N: 데이터 수

y_i : 실제 값

x_i : 입력 값

$\mathbf{w}^T \mathbf{x}_i + b$: 예측된 값.

제공은 +인지 - 인지 상관없이 조금이라도 차이 나면 더 돋보이게 할려고.

회귀(Regression)

- Polynomial Regression(다항 회귀)

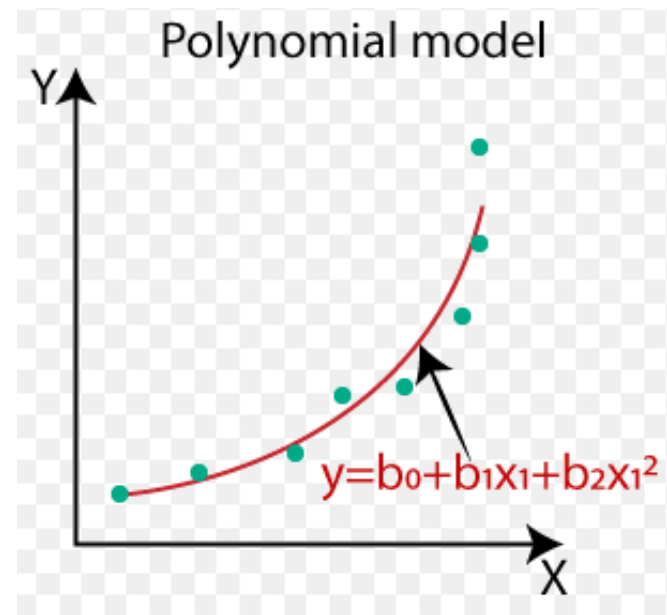
$$y = w_0 + w_1x + w_2x^2 + w_3x^3 + \dots + w_dx^d$$

y: 출력 변수

x: 입력 변수

w_0, w_1, \dots, w_d : 가중치

d: 차수



동일하게 선형 회귀 범주에 속함.

차수가 작고 높고의 문제는

overfitting, underfitting 문제와 직결

회귀(Regression)

-Ridge/Lasso

규제(regularization): overfitting 방지하려고

회귀(Regression)

-Ridge regression == L2 regularization

$$J(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda \sum_{j=1}^n w_j^2$$

$J(\mathbf{w})$: 손실함수

$(y_i - \mathbf{w}^T \mathbf{x}_i)^2$: 예측 값이랑 실제 값 차이 제곱

λ : 규제 강도 조절 파라미터 값

$\sum_{j=1}^n w_j^2$: L2 규제 항 # 가중치 제곱

$$\lambda \sum_{j=1}^n w_j^2$$

가중치가 클수록 제곱값도 같이 커진다.

그러므로 전체 $J(\mathbf{w})$ 값 자체가 커짐.

회귀(Regression)

-Lasso regression == L1 regularization

$$J(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N (y_i - \mathbf{w}^T \mathbf{x}_i)^2 + \lambda \sum_{j=1}^n |w_j|$$

회귀(Regression)

-Ridge/Lasso

L1

주로 feature selection에 사용
변수 많을 때, 중요 변수만 남기는 방식

- 절대값이 선형적으로 증가하는 효과 부여
- 그래서 가중치 0이 되는 경우 생김

L2

모든 feature 다 쓰면서 가중치 조절해서 모델 overfitting 방지

- 제곱값이 가중치 크면 더 큰 손실, 작으면 크게 변화 X
- 그래서 0에 가까워 지기만 하고 딱 0은 되지는 않음.

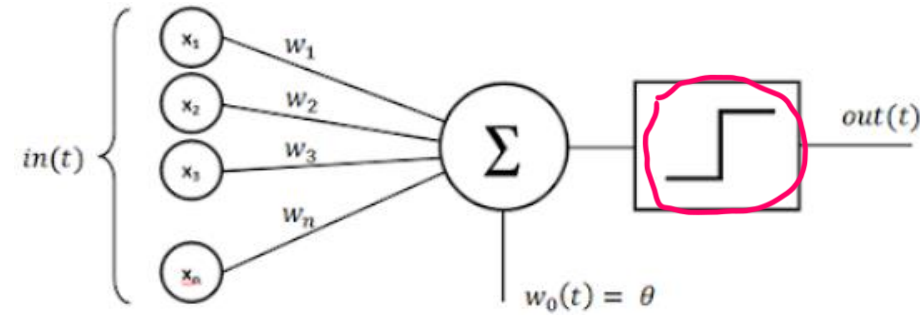
분류(Classification)

- Perceptron

$$f(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \rightarrow \text{단위 계단 함수 (activation function)}$$

결국 이진 분류에 사용.

굉장히 초창기 분류 문제



$$y = \sum_{i=1}^n w_i x_i + b$$

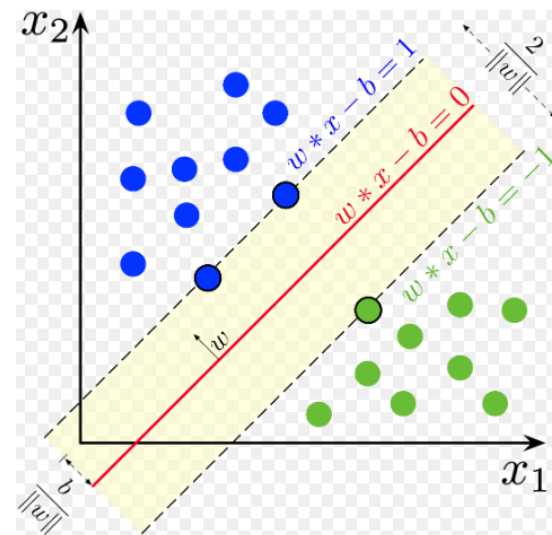
분류(Classification)

- SVM(support vector machine)

최대 마진 찾는 것

마진: 두 클래스 사이의 간격

➔ 이걸 최대화 해서 두 클래스 잘 구분하는
'결정경계' 만드는 것



Decision Hyperplane: 두 클래스 분리하는 선. 2차원에선 선, 3차원에선 평면

Margine: 가장 가까운 데이터 포인트랑 Decision Hyperplane 사이의 거리 # 이 마진을 최대화 하는게 관건

Support vector: Decision Hyperplane에 가장 가까운 데이터 포인트들

$$w \cdot x + b = 0$$

decision
hyperplane
정의 식

$$M = \frac{1}{||w||}$$

분류(Classification)

- **Logistic/softmax Regression**

말만 회귀

이진 분류 / 다중 분류

분류(Classification)

- Logistic Regression

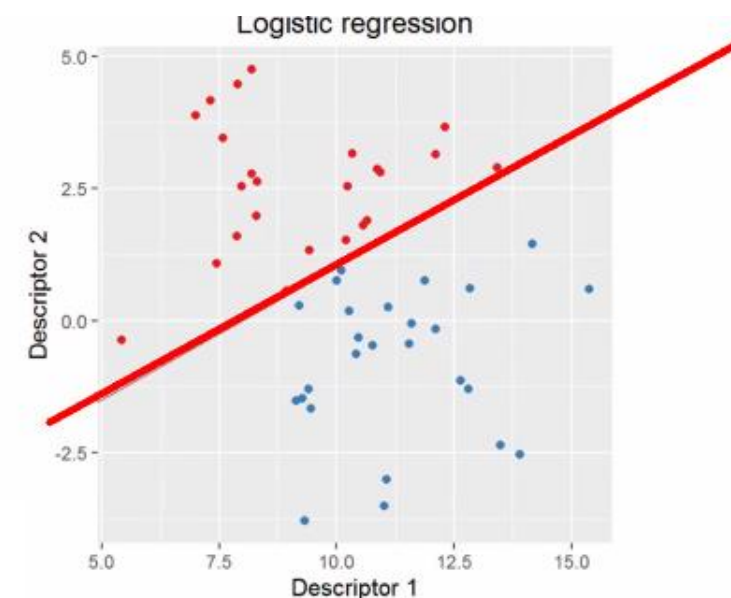
$$z = w^T x + b$$

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

선형회귀와 비슷하게 선형 결합 형태

but,
출력값을 그대로 쓰지 않고, 시그모이드
함수 사용하여 확률값으로 변환

$$\hat{y} = \begin{cases} 1 & \text{if } \sigma(z) \geq 0.5 \\ 0 & \text{if } \sigma(z) < 0.5 \end{cases}$$



분류(Classification)

- Softmax Regression

$$z_k = w_k^T x + b_k$$

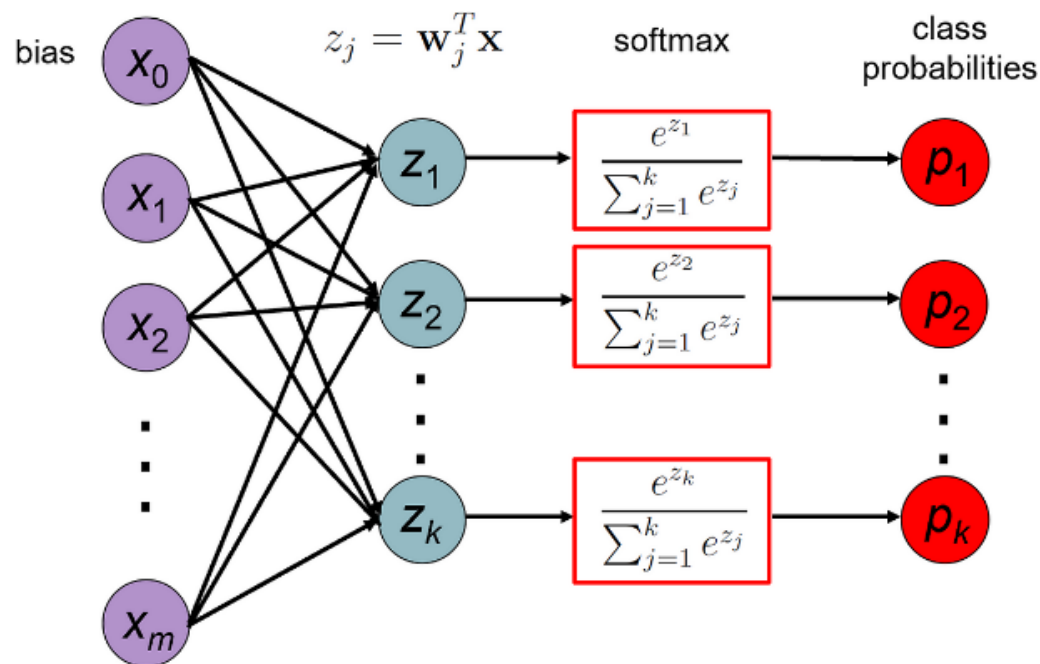
$$P(y = k | x) = \frac{e^{z_k}}{\sum_{j=1}^K e^{z_j}}$$

결국 z_k 의 확률값을 반환.

모든 클래스의 확률 합이 1

동일한 선형 결합 형태

그래서 로지스틱 회귀의 연장선 # 클래스 2개라면 동일



Tree Model

- Decision Tree Classifier # 분류
- Random Forest Classifier # 분류
- Decision Tree Regressor # 회귀
- Gradient Boosting Regressor # 회귀

등등

Tree Model

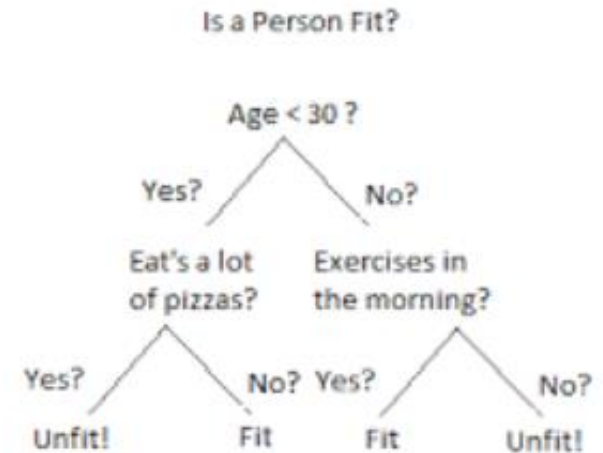
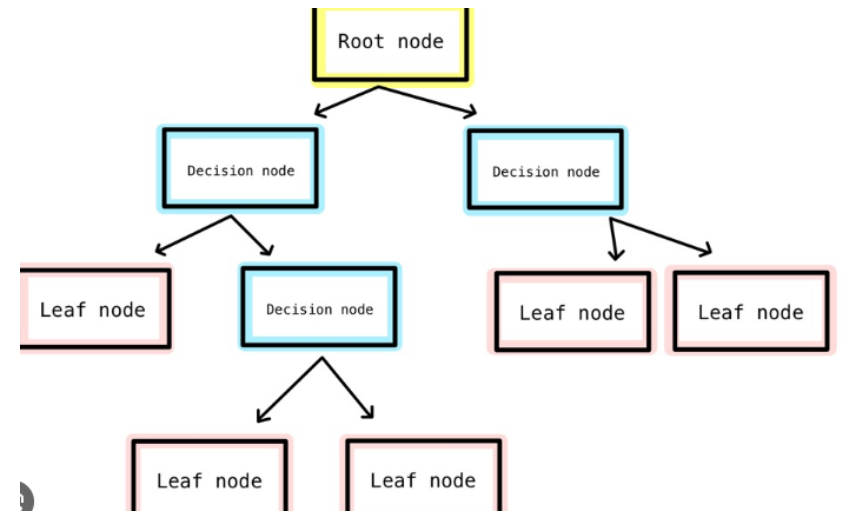
- Decision Tree Classifier # 분류

데이터 특성 기준으로 의사결정 규칙 만들어서 분류 하는 모델

루트 노드: 데이터 처음 나누는 지점

결정 노드: 데이터 특성에 따라 분기되는 지점

리프 노드: 더 이상 분할 안되는 종단 지점. 최종 클래스 결정



Tree Model

- Decision Tree Classifier # 분류

$$\text{Gini}(D) = 1 - \sum_{i=1}^C P_i^2$$

Gini는 불순도 지표. → 데이터가 얼마나 혼합되어 있는지

0이면 완벽하게 한 클래스에 속한 것, 0.5에 가까우면 클래스 여러 개 혼합 된 상태

P_i : 클래스 i 에 속할 확률

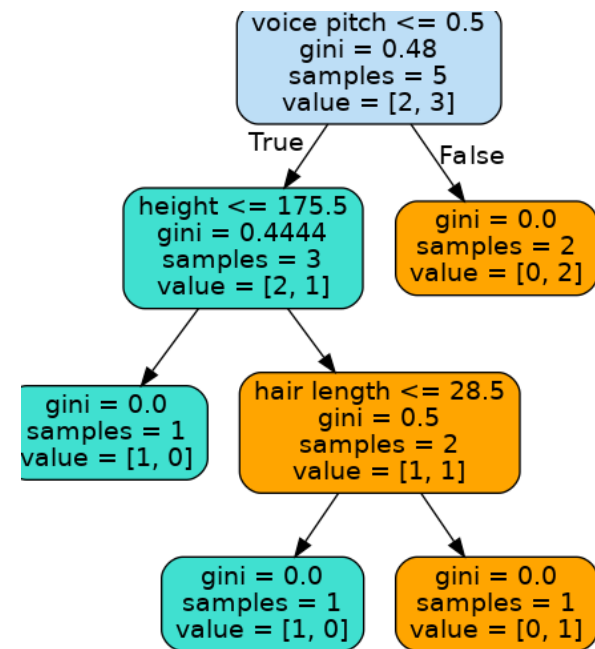
C : 클래스 개수

$$\text{Entropy}(D) = - \sum_{i=1}^C P_i \log_2 P_i$$

D : 데이터셋

C : 클래스 수

P_i : 클래스 i 에 속할 확률



Gini와 동일한 분류 지표

Tree Model

- Decision Tree Regressor # 회귀

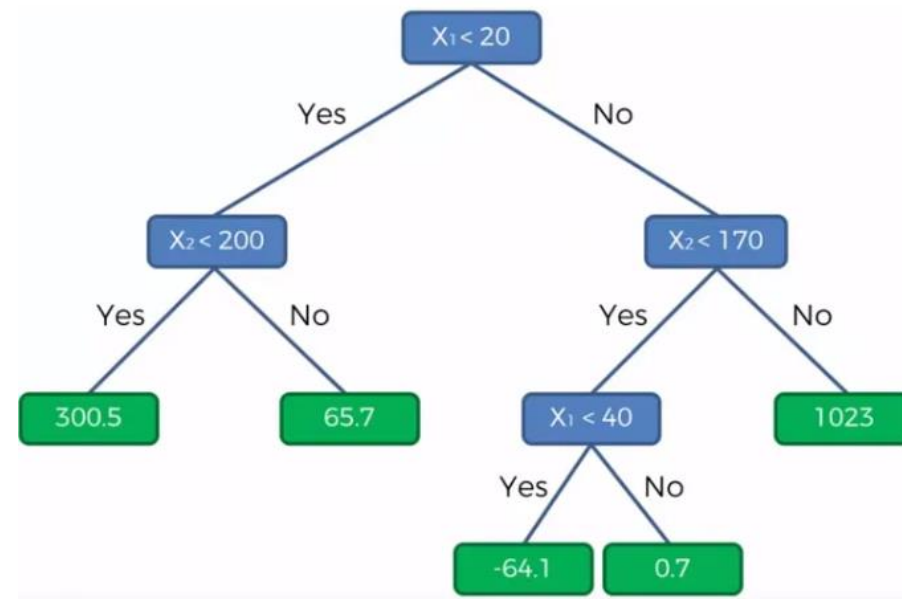
데이터를 여러 구간으로 나눠서 각 구간 내에서 예측값 내는 것

각 노드 개념은 동일.

but, 리프노드에서 평균이나 중앙값을 최종 예측값으로 내는 차이 존재

Decision Tree Classification에서는 Gini, Entropy를 사용.

Decision Tree Regressor에서는 MSE나 분산(데이터가 얼마나 잘 몰려있는지)을 지표로 주로 사용



$$\text{Variance}(D) = \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2$$

Tree Model

- Decision Tree Classification / Regressor

Tree가 너무 깊을 때, Overfitting 주의

Tree가 너무 얇을 때, Underfitting 주의

- 가지 깊이를 제한, 혹은 가지 몇 개를 제거.
- 앙상블 기법(Random Forest, Gradient Boosting) 적용으로 이를 방지.
- 혹은 Cross Validation으로 살펴보기.

Assignment

- 회귀 문제에서의 손실함수 더 찾아볼 것
- 수업 내용 정리
- 오늘 나온 내용, sklearn 라이브러리 활용 후 적용
- 분류문제 데이터: IRIS
- 회귀문제 데이터: Diabetes

→ (폴더명: 2주차_3기_@@@
→ 한 폴더 안에
→ 손실함수 더 찾은것 & 수업 내용 정리,
→ .ipynb 파일)

하고 Pull Request