

# 인천 진산과학과 특강 2일차

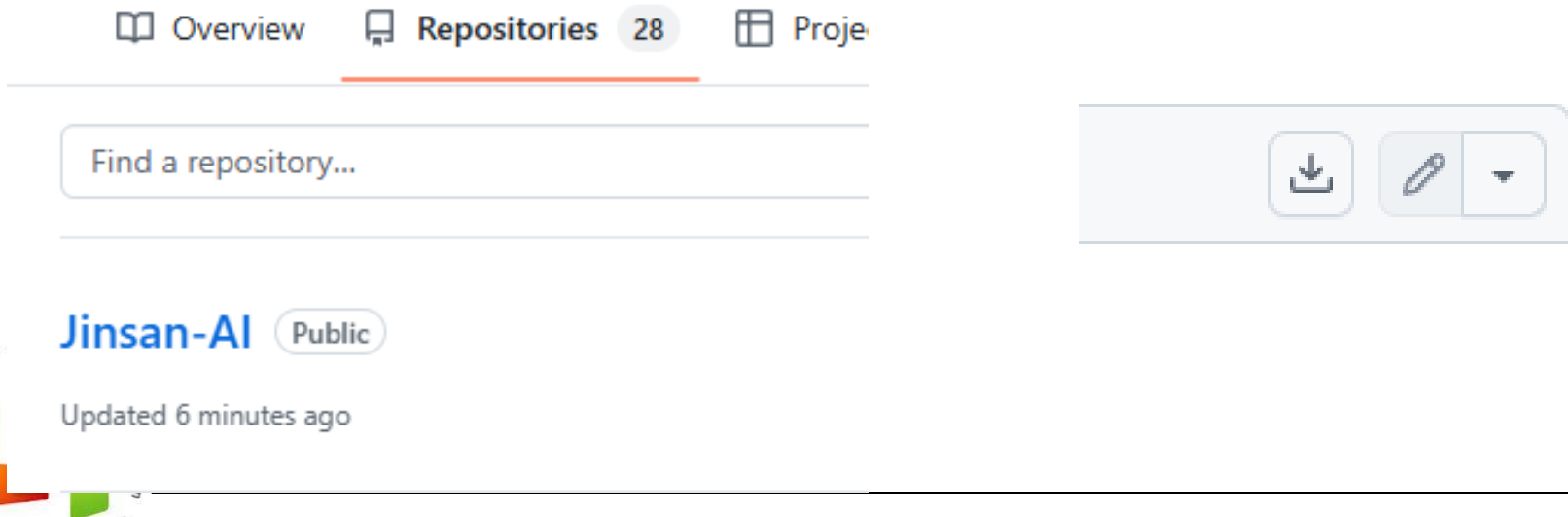
---

**Prof. Jibum Kim (jibumkim@inu.ac.kr)**

**Department of Computer Science & Engineering  
Incheon National University**

# Python 기초

- 오늘 강의 자료 다운 받는 법
- 1. <https://github.com/DevSlem> 주소 침
- 2. Repositories 클릭 후 젤 위의 “Jinsan-AI” 클릭
- 3. “진산과학고\_2일차.pdf” 클릭
- 3. 아래 오른쪽 같이 우측 빨간 박스로 표시된 “다운로드 버튼” 클릭



---

- practice2.ipynb - Colab

---

특강 내용

# 머신 러닝 (ML) 이란?

- 
- **지능 (intelligence)**
  - 본능적이나 자동적으로 행동하는 대신에,  
**생각**하고 **이해**하여 **행동**하는 능력
  - **인공 지능 (AI)**
  - 인간처럼 지능을 가지고 있는 기계

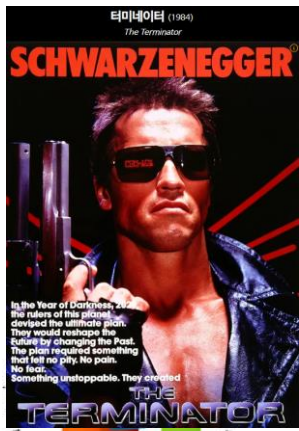
- **인공지능 (AI, Artificial Intelligence)이란?**
- 사람과 유사한 지능을 가지도록 인간의 학습능력, 추론능력, 지각능력 등을 컴퓨터 프로그램으로 실현하는 기술
- 문제 해결을 위해 상황을 인지하고 파악하고 추론하여 답을 얻어내는 인간의 지능을 컴퓨터가 가질 수 있도록 실현한 기술

"기계를 인간 행동의 지식에서와 같이 행동하게 만드는 것"



1956년 존 매커시 교수가 처음으로 인공지능이라는 용어 씀

- **자의식을 가진 기계 – 터미네이터 (1984)**
- 터미네이터가 다쳐서 한 허름한 모텔에서 치료 중
- 청소부가 불쾌한 목소리로 **“이봐 너 거기 안에 뭐라도 숨겼지. 방 안에서 뭐해?”**라고 물음.
- 터미네이터에게는 가능한답변으로 다음과 같은 선택지가 주어짐.  
거절의 표현으로 **정답은? 아래 4개 보기 중에?**
- 진짜 인공지능? 인간처럼 상황에 맞게 대답을 하고 행동



1. Yes/ No? (그래/아니)

2 What? (어쩌라고)

3. Please come back later

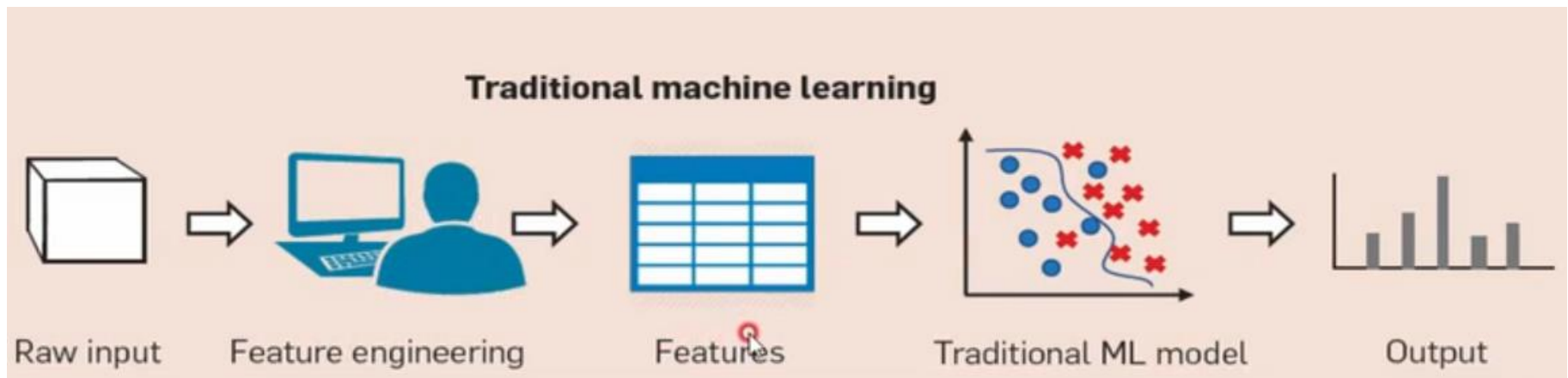
4. 뭐라고? 저리가

- 머신 러닝 (machine learning, 기계 학습)
- 인공지능의 하위 분야
- 규칙을 일일이 프로그래밍하지 않아도 **자동으로 데이터에서 규칙을 학습**하는 알고리즘을 연구하는 분야
- 1) 프로그래머가 명시적으로 코딩하지 않고 자기 개선과 예제를 통해 학습
- 2) 학습으로 얻어진 정보를 기반으로 미래의 임의의 데이터에 대한 결과 (값, 분포)를 예측

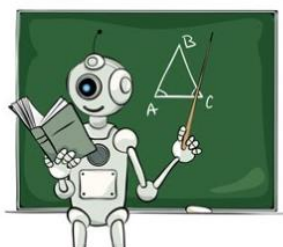


- 전통적인 머신 러닝

- 예: 키와 성별을 주고 몸무게를 예측한다고 하면  
키와 성별이 “feature”



- 머신 러닝의 종류
- 머신 러닝의 학습 방법은 크게 3가지로 분류
- 특강에서는 주로 지도 학습



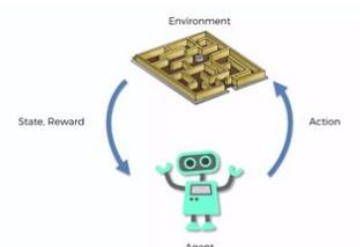
Supervised  
Learning

VS



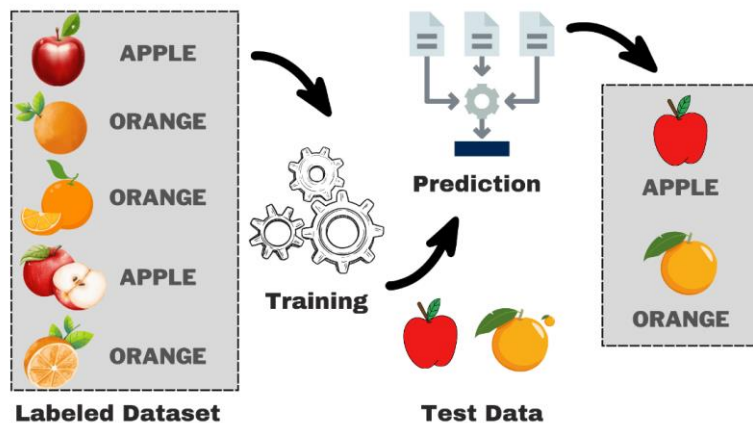
Unsupervised  
Learning

VS



Reinforcement  
Learning

- 지도 학습 (supervised learning)
- 정답을 알려주며 학습시키는 것
- 예: 고양이 사진을 입력 데이터로 주고, 이 사진은 고양이 (정답지 -label data)야
- 크게 분류 (classification)와 회귀 (regression)



## ■ 머신 러닝에 필요한 주요 요소

1. 다양한 형식의 데이터 (동영상, 텍스트, 이미지 등)
2. 알고리즘: 컴퓨터에게 지능을 심어주는 부분.

수학적 통계모델, 의사결정트리 (3일차), 서포트 벡터머신 (오늘), 신경망등

- 대표적인 머신러닝 라이브러리는 **사이킷런 (Scikit-learn)**
- 파이썬 기반으로 배우기 쉽고 사용하기 편리
- scikit-learn: machine learning in Python — scikit-learn 1.5.1 documentation

The screenshot shows the official scikit-learn documentation website for version 1.5.1. The header includes the 'scikit-learn' logo, navigation links (Install, User Guide, API, Examples, Community, More), a search icon, and the version number '1.5.1 (stable)'. The main content area is divided into three columns: Classification, Regression, and Clustering. Each column contains a brief description, applications, and algorithms. The Classification section includes a grid of 16 small plots showing various data distributions. The Regression section features a line plot titled 'Predicted average energy transfer during the week'. The Clustering section shows a scatter plot titled 'Kmeans clustering on the digits dataset (PCA-reduced data)' with centroids marked by white crosses.

**scikit-learn**  
*Machine Learning in Python*

Getting Started Release Highlights for 1.5

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

### Classification

Identifying which category an object belongs to.

**Applications:** Spam detection, image recognition.

**Algorithms:** [Gradient boosting](#), [nearest neighbors](#), [random forest](#), [logistic regression](#), and [more...](#)

### Regression

Predicting a continuous-valued attribute associated with an object.

**Applications:** Drug response, stock prices.

**Algorithms:** [Gradient boosting](#), [nearest neighbors](#), [random forest](#), [ridge](#), and [more...](#)

### Clustering

Automatic grouping of similar objects into sets.

**Applications:** Customer segmentation, grouping experiment outcomes.

**Algorithms:** [k-Means](#), [HDBSCAN](#), [hierarchical clustering](#), and [more...](#)

- 1일차에 배운 “선형 최소 제곱법 ” 은 **회귀분석 (regression)** 방법의 하나로 머신 러닝의 중요한 방법 중의 하나이다
- **Regression:** predicting a continuous-valued attribute associated with an object
- 예: 미래 주식 가격 예측, 미래 drug response, 미래 주택 가격 예측

## scikit-learn

Machine Learning in Python

Getting Started

Release Highlights for 1.6

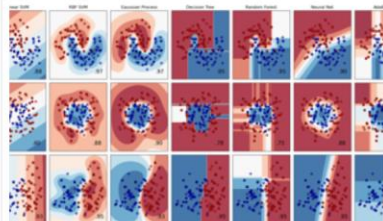
- Simple and efficient
- Accessible to everyl
- Built on NumPy, Sci
- Open source, comm

### Classification

Identifying which category an object belongs to.

**Applications:** Spam detection, image recognition.

**Algorithms:** [Gradient boosting](#), [nearest neighbors](#), [random forest](#), [logistic regression](#), and more...

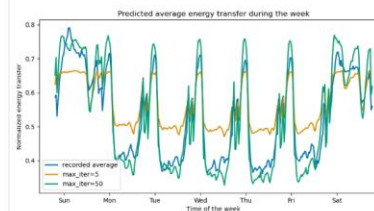


### Regression

Predicting a continuous-valued attribute associated with an object.

**Applications:** Drug response, stock prices.

**Algorithms:** [Gradient boosting](#), [nearest neighbors](#), [random forest](#), [ridge](#), and more...



---

## 특강 내용

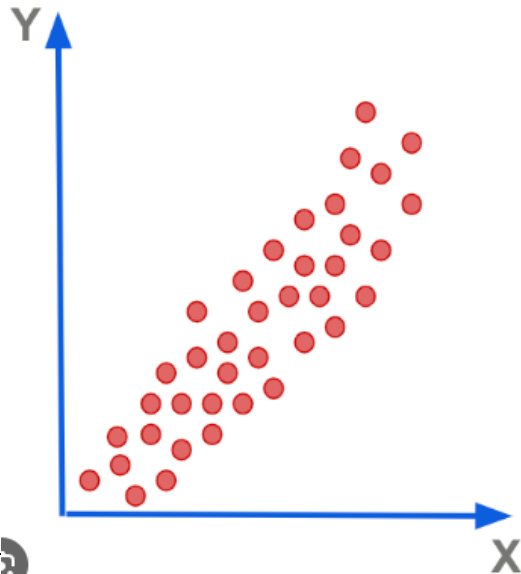
# ML 기반 Linear regression (선형 회귀 분석)

- 1일차에서 “최소제곱법 ” 을 이용한 선형 회귀분석을 배웠다
- 이번에는 ML을 이용한 선형 회귀 분석에 대해서 배워본다
- Linear regression (선형 회귀 분석)
- 주어진 데이터를 가장 잘 fit (근사화)하는 직선 형태의 회귀선을 찾는 방법
- 주어진 n개의 이산 데이터들  $(x_i, y_i)$ 이 있다
- 목표: Fitting a “best” line,  $y = ax + b$ , “best”?

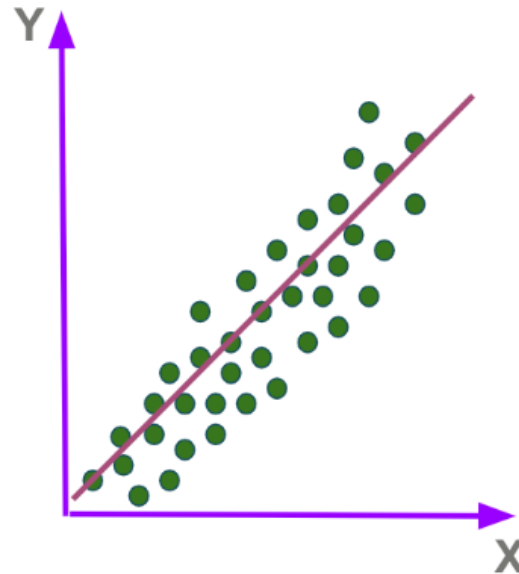


- 1일차에서 산점도 그래프에 대해서 배웠다.
- 데이터를 분석 시에 두 변수 사이에 양의 상관관계를 갖고 직선으로 근사화가 가능하다면 **linear regression**으로 데이터를 근사화 가능하다

Correlation



Linear Regression



- **손실 함수 (loss function):** 머신 러닝 모델의 예측값과 실제값 간의 차이를 수치적으로 나타낸 함수
- 모델이 얼마나 잘못 예측했는지를 측정, 이를 바탕으로 모델을 개선
- **Linear regression에서의 손실 함수**
- $MSE = J = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \frac{1}{n} \sum_{i=1}^n (y_i - (ax_i + b))^2$
- $y_i$  : 실제 값,  $\hat{y}_i$  : 예측값

- **목표:** ML 모델이 손실 함수인 MSE를 최소화도록 학습, 즉, MSE를 최소화하는 직선 찾기 (a와 b 찾기)
- **How?** 경사하강법 사용하여 **다변수 손실함수인 MSE를 최소화하도록** a와 b를 update
- $a := a - \alpha \frac{\partial J}{\partial a}, b := b - \alpha \frac{\partial J}{\partial b}$
- 여기서  $\alpha$ : 학습률 (step size),  $J = \frac{1}{n} \sum_{i=1}^n (y_i - (ax_i + b))^2$

- ML 기반 Linear regression 학습 방법

- 1. 초기화: 직선  $y = ax + b$ 의  $a$ 와  $b$ 를 임의로 초기화
- 2. 반복

현재  $a$  와  $b$  값으로  $\frac{\partial J}{\partial a}$  와  $\frac{\partial J}{\partial b}$  계산

경사하강법으로  $a$  와  $b$  값 업데이트:  $a := a - \alpha \frac{\partial J}{\partial a}$ ,  $b := b - \alpha \frac{\partial J}{\partial b}$

손실함수 값 확인. 계속? 중단?

- 3. 수렴: 손실 함수의 변화량이 충분히 작아지면  $a$ 와  $b$  값 출력

- 실습:
- 1.  $y = 3x + 7$ 에 **noise**를 추가한 데이터를 100개 생성
- **시드 (seed)값?** 난수를 컴퓨터에서 생성시에 초기 상태를 결정하는값. 컴퓨터가 실제로는 진짜로 무작위로 숫자를 생성하지 않고 “가짜 난수”를 생성. 이때 시드값이 같으면 생성되는 난수도 같음
- `np.random.rand`: 0~1사이 무작위 난수 생성

```
# 1. 데이터 생성
np.random.seed(42) # 재현성을 위한 시드 설정
n_samples = 100

#  $y = 3x + 7 + \text{noise}$  형태의 데이터 생성
X = np.random.rand(n_samples) * 10 # 0 ~ 10 사이의 랜덤 값
true_slope = 3
true_intercept = 7
noise = np.random.randn(n_samples) # 노이즈 추가
y = true_slope * X + true_intercept + noise
```

## ■ 2. 손실함수 및 gradient 계산

### ■ 손실함수 (MSE):

```
# 손실 함수(MSE) 계산 함수
def compute_loss(X, y, a, b):
    y_pred = a * X + b
    loss = np.mean((y - y_pred) ** 2)
    return loss
```

### ■ $\frac{\partial J}{\partial a}$ 와 $\frac{\partial J}{\partial b}$ 계산

### ■ (13페이지 참고)

```
# 기울기 계산 함수
def compute_gradients(X, y, a, b):
    n = len(X)
    y_pred = a * X + b
    da = -2 / n * np.sum(X * (y - y_pred))
    db = -2 / n * np.sum(y - y_pred)
    return da, db
```

### ■ 3. 경사하강법 수행

- $a := a - \alpha \frac{\partial J}{\partial a}$ ,  $b := b - \alpha \frac{\partial J}{\partial b}$ , 학습률 (learning rate)은 0.01로 정함
- **Epoch**: ML 모델이 학습하는 동안 훈련 데이터를 한번 모델에 입력하고, 모델의 파라미터를 업데이트하는 과정을 1 epoch라고 한다
- 충분히 MSE Loss가 줄 정도로 epoch 크기를 정하자

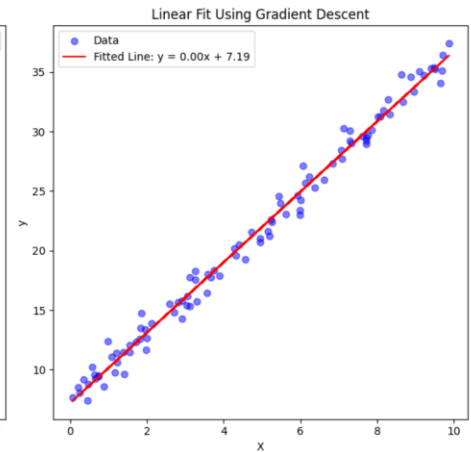
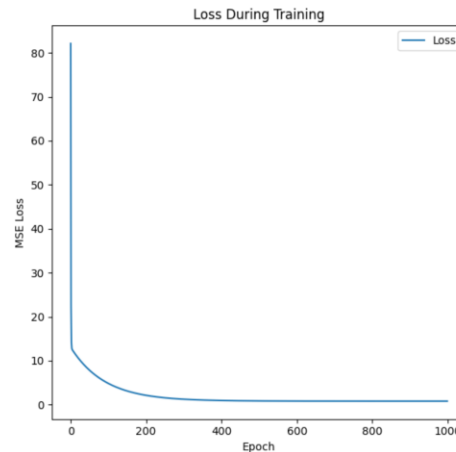
```
# Gradient Descent 학습
loss_history = []
for epoch in range(n_epochs):
    da, db = compute_gradients(X, y, a, b)
    a -= learning_rate * da
    b -= learning_rate * db

    # 손실 계산 및 저장
    loss = compute_loss(X, y, a, b)
    loss_history.append(loss)

    # 100번마다 로그 출력
    if (epoch + 1) % 100 == 0:
        print(f"Epoch {epoch + 1}/{n_epochs}, Loss: {loss:.4f}, w: {w:.4f}, b: {b:.4f}")
```

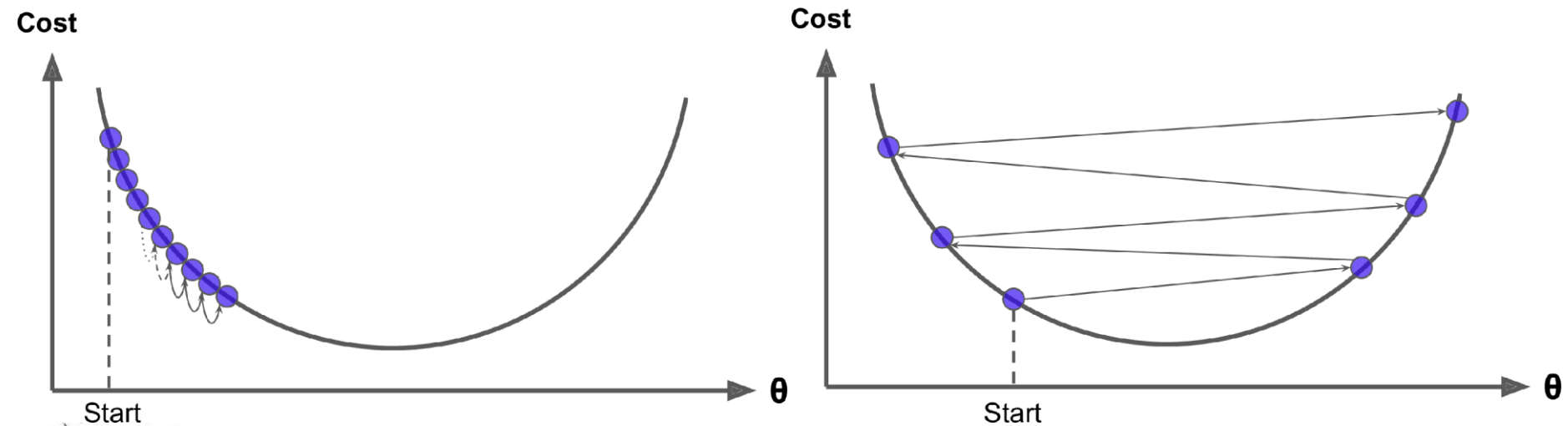
- 4. 손실 함수 및 fit 결과 시각화
- Epoch이 증가함에 따라서 손실함수 (MSE loss) 감소 확인
- Fit된 함수가 처음에 기대한  $y=3x+7$ 과 거의 유사함 확인

Epoch 100/1000, Loss: 4.8783, w: 3.5361, b: 3.4280  
Epoch 200/1000, Loss: 2.1452, w: 3.2878, b: 5.0437  
Epoch 300/1000, Loss: 1.2466, w: 3.1454, b: 5.9701  
Epoch 400/1000, Loss: 0.9513, w: 3.0637, b: 6.5012  
Epoch 500/1000, Loss: 0.8541, w: 3.0169, b: 6.8058  
Epoch 600/1000, Loss: 0.8222, w: 2.9901, b: 6.9804  
Epoch 700/1000, Loss: 0.8117, w: 2.9747, b: 7.0805  
Epoch 800/1000, Loss: 0.8083, w: 2.9659, b: 7.1379  
Epoch 900/1000, Loss: 0.8071, w: 2.9608, b: 7.1709  
Epoch 1000/1000, Loss: 0.8068, w: 2.9579, b: 7.1897





- **학습률 선택의 중요성**
- 너무 작으면: 수렴이 매우 오래 걸림
- 너무 크면: 발산함



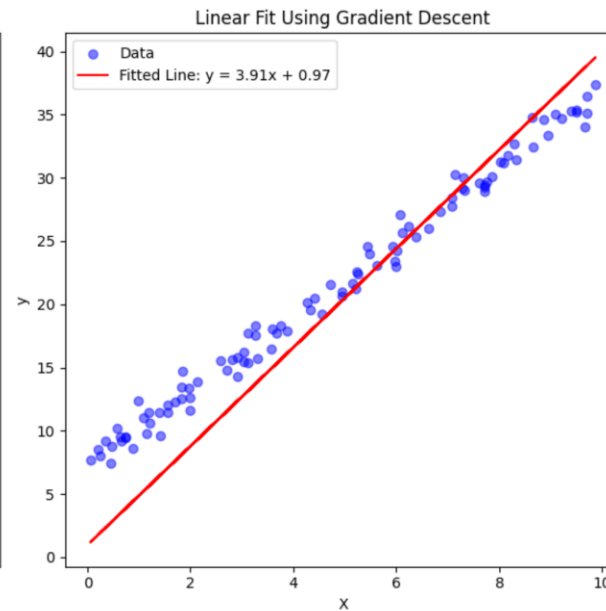
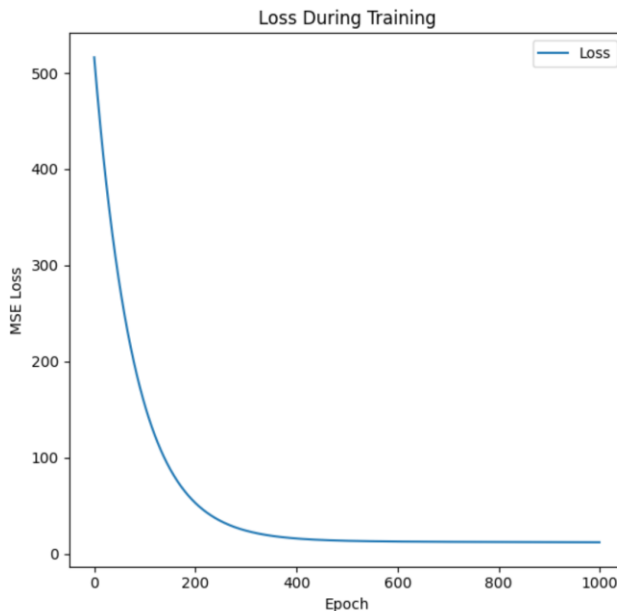
## ■ 학습률 선택의 중요성

### ■ 1. 학습률을 매우 작게 줄여봄

손실 함수가 느리게 감소, fitting도 덜 됨

```
learning_rate = 0.0001
```

```
n_epochs = 1000
```



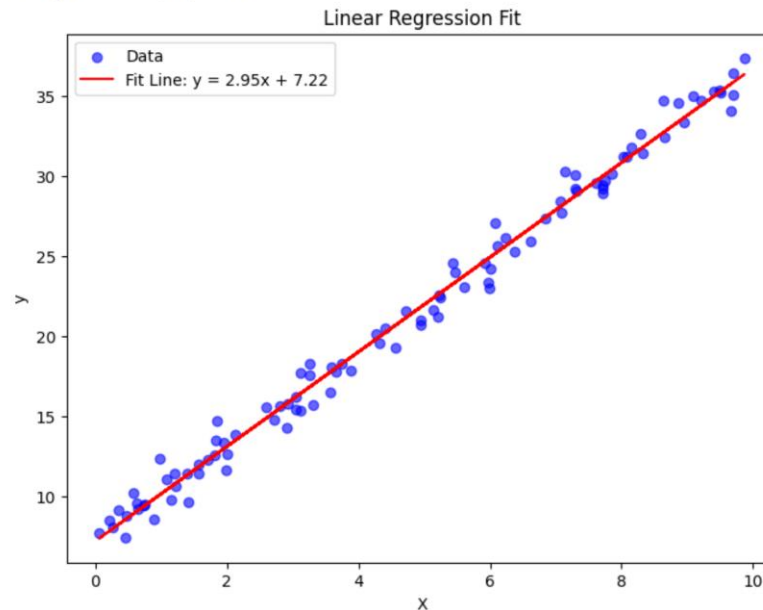
## ■ 2. 학습률을 매우 크게 해보자. 발산?

```
learning_rate = 0.1  
n_epochs = 1000
```

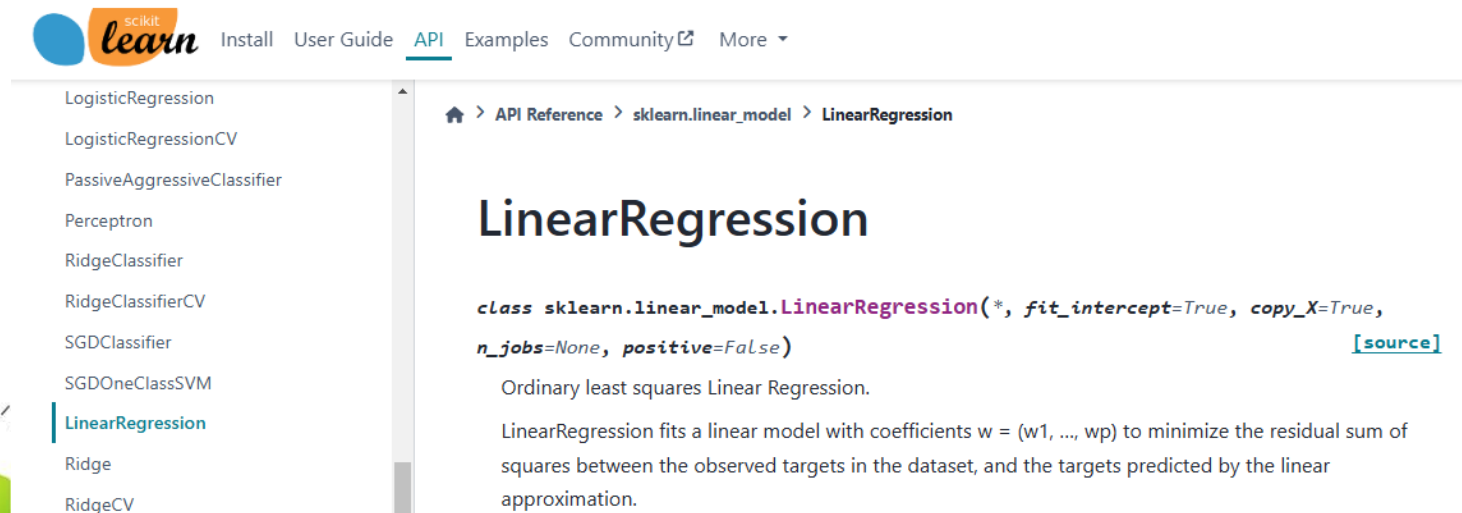
```
Epoch 100/1000, Loss: 72531917266224610080983233697725100417889617166378716012445672001974038982163915  
Epoch 200/1000, Loss: 10326321330181877118725139337048725614084113138165464911956009757955011759762671  
Epoch 300/1000, Loss: inf, w: -21322017683700482410929991630909728819612497299509836277538861831869353  
Epoch 400/1000, Loss: inf, w: -80451943143694985819513142081323067163469925180792252696441619556938098  
Epoch 500/1000, Loss: nan, w: nan, b: nan  
Epoch 600/1000, Loss: nan, w: nan, b: nan  
Epoch 700/1000, Loss: nan, w: nan, b: nan  
Epoch 800/1000, Loss: nan, w: nan, b: nan  
Epoch 900/1000, Loss: nan, w: nan, b: nan  
Epoch 1000/1000, Loss: nan, w: nan, b: nan  
<ipython-input-16-3bfecf17c7c8>:25: RuntimeWarning: overflow encountered in square  
    loss = np.mean((y - y_pred) ** 2)  
<ipython-input-16-3bfecf17c7c8>:40: RuntimeWarning: invalid value encountered in scalar subtract  
    a -= learning_rate * da
```

- Scikit-learn에는 linear regression을 수행하는 함수 **“LinearRegression”**이 있다. 이를 사용하면 쉽게 linear regression 수행 가능. Scikit-learn 결과와 비교

기울기 (slope): 2.9540  
절편 (intercept): 7.2151  
Mean Squared Error (MSE): 0.8066



- [https://scikit-learn.org/1.5/modules/generated/sklearn.linear\\_model.LinearRegression.html](https://scikit-learn.org/1.5/modules/generated/sklearn.linear_model.LinearRegression.html)
- 어떤 파라미터가 있는지, 설명도 나와있음.
- 대부분 기본 파라미터가 좋음



The screenshot shows the scikit-learn website's API reference for the `LinearRegression` class. The left sidebar lists various models, with `LinearRegression` selected. The main content area displays the class signature: `class sklearn.linear_model.LinearRegression(*, fit_intercept=True, copy_X=True, n_jobs=None, positive=False)` with a `[source]` link. Below the signature, it states: "Ordinary least squares Linear Regression. LinearRegression fits a linear model with coefficients  $w = (w_1, \dots, w_p)$  to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation."

## ■ Scikit-learn에서의 linear regression

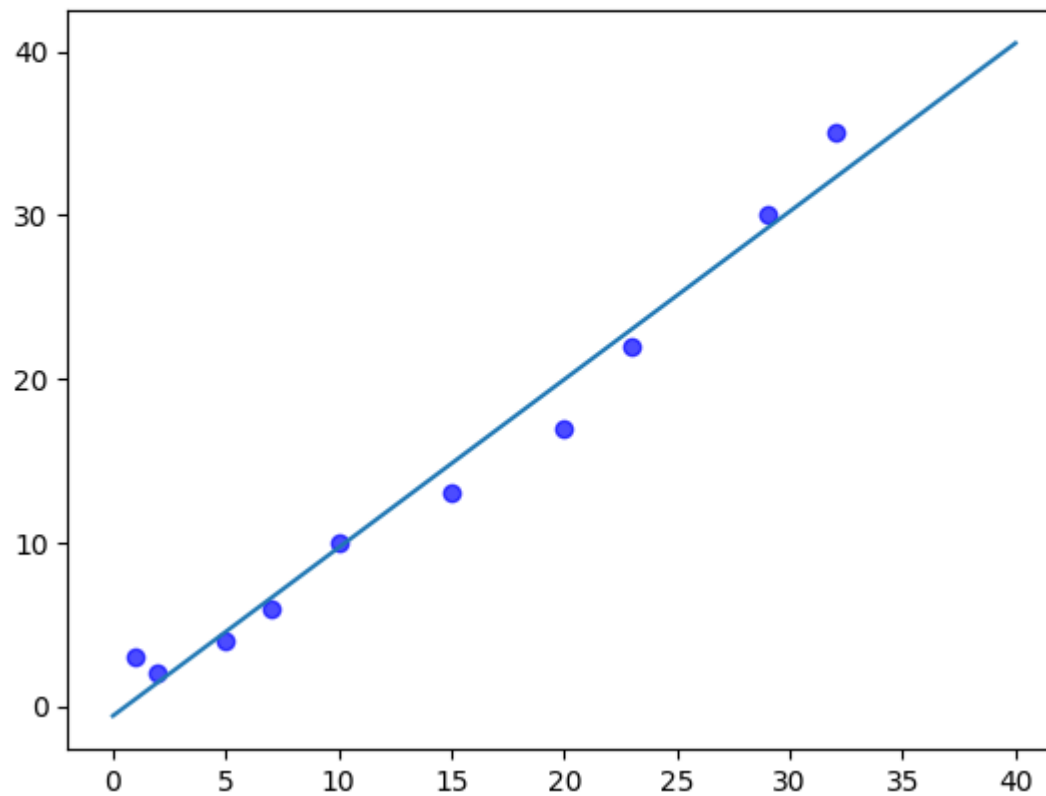
```
# 2. Linear Regression 모델 학습  
model = LinearRegression()  
model.fit(X, y) # 모델 훈련
```

## ■ 실습 문제 1

- 다음 데이터에 대해서 이 데이터를
- 1. Pandas 의 dataframe으로 저장하자
- 2. 산점도 그래프를 그려보자
- 예: `plt.scatter(df['Feature1'], df['Feature2'], color='blue', alpha=0.7)`
- 3. Scikit-learn의 linear regression 함수를 사용하여 fit하자
- 4. fit한 직선을 그려보자 (예: `np.linspace`사용)
- 5.  $x=30$ 일때의 값을 예측해보자

X	1	2	5	7	10	15	20	23	29	32
y	3	2	4	6	10	13	17	22	30	35

기울기 (slope): 1.0274  
절편 (intercept): -0.5945  
x=30일때의 값은:  
30.227321237993593



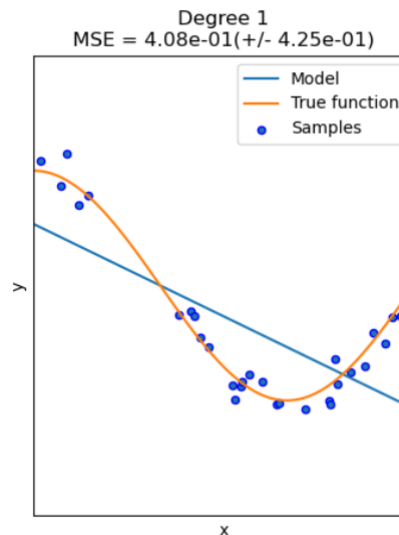


---

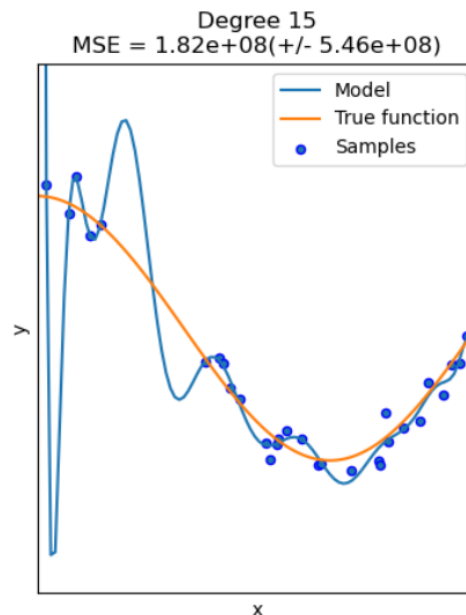
특강 내용

# Overfitting vs underfitting

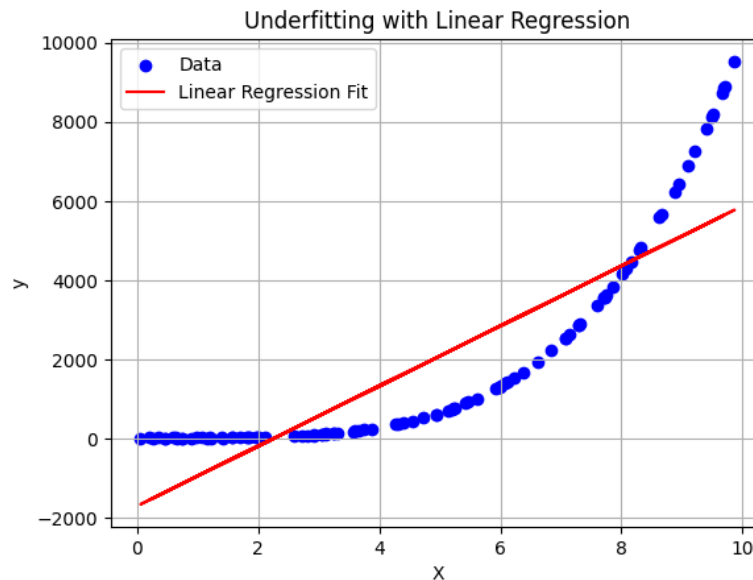
- **Underfitting:** ML 모델이 학습 데이터에 대해서 충분히 학습하지 못한 상태
- **문제점:** 현재 학습 데이터에 대해서도 낮은 성능
- 예: underfitting된 이유는 1차 직선으로 충분하지 않고 더 고차 함수 필요



- **Overfitting:** ML 모델이 학습 데이터에 지나치게 맞추어 학습한 상태
- 이유: 현재 학습 데이터에 맞춰서 너무 고차 다항식 함수로 fit
- **문제점:** 미래 데이터 예측에는 실패 (일반화 성능 약함)

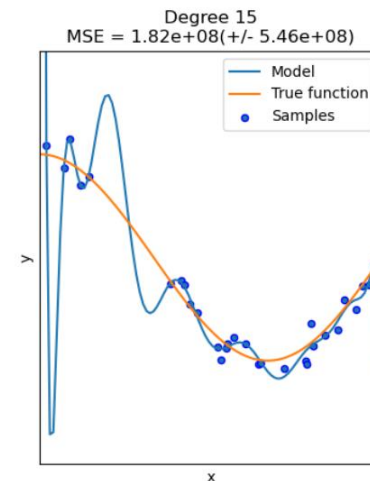
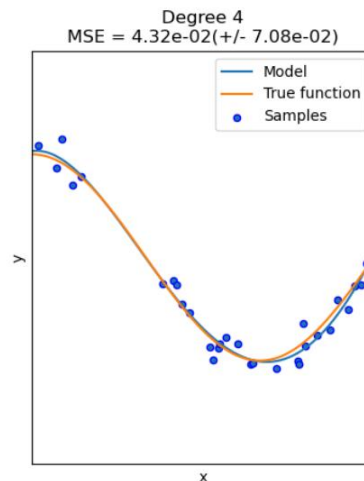
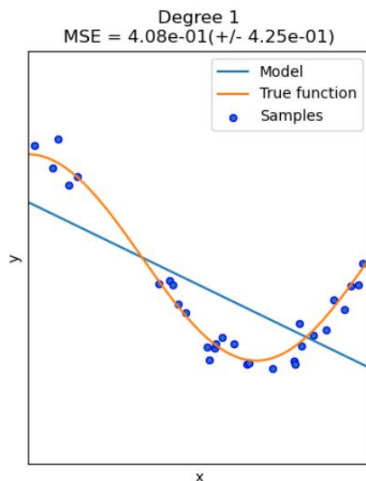


- 실습: UNDER FITTING 예
- 이런 경우에는 직선이 아닌 고차 다항식으로 FIT할 수 있다



- (실습)적절한 fitting: 가운데 처럼 4차 다항식으로 fitting한 경우
- Underfitting vs. Overfitting — scikit-learn 1.6.0 documentation
- 1차, 4차, 15차로 regression

```
n_samples = 30  
degrees = [1, 4, 15]
```



---

특강 내용

# EDA (탐색적 데이터 분석)

- **Iris 데이터셋**
- 머신 러닝 입문자에게 많이 사용되는 데이터셋으로 꽃 종류를 **분류**
- **Feature (특징):** sepal length (cm), sepal width (cm), petal length (cm), petal width (cm)
- **Target class: 3개**, Setosa, Versicolor, Virginica
- 총 **150개**의 데이터, 각 class 별로 50개의 데이터



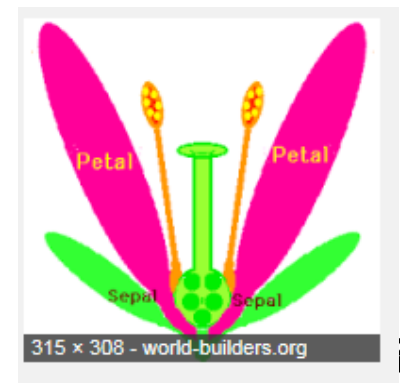
Iris setosa



Iris versicolor



Iris virginica



---

•예) Iris 데이터의 일부분 (120개)

Sepal 길이	Sepal 높이	Petal 길이	Petal 높이	분류
5.1	3.5	1.4	0.2	Iris. Setosa (0)
4.9	3.0	1.4	0.2	Iris. Setosa (0)
7.0	3.2	4.7	1.4	Iris.versicolor (1)
6.4	3.2	4.5	1.5	Iris.versicolor (1)
5.9	3.0	5.1	1.8	Iris.virginica (2)
...	...	...	...	...

---

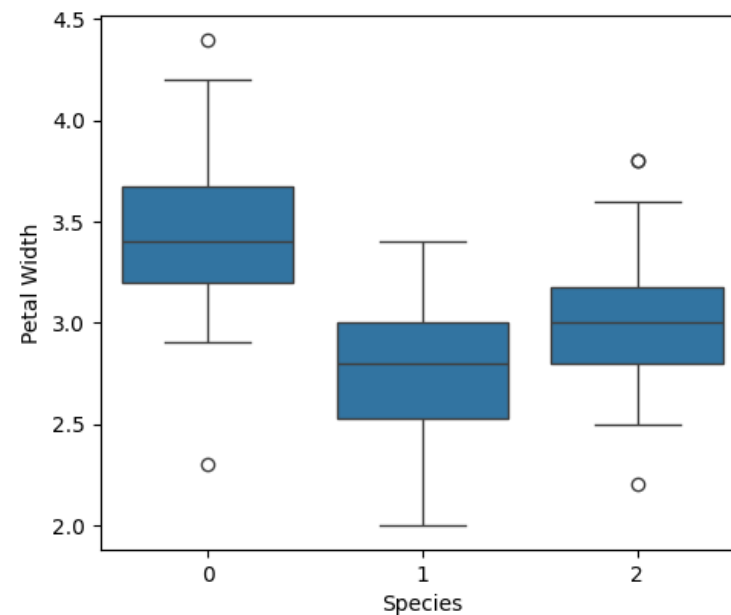
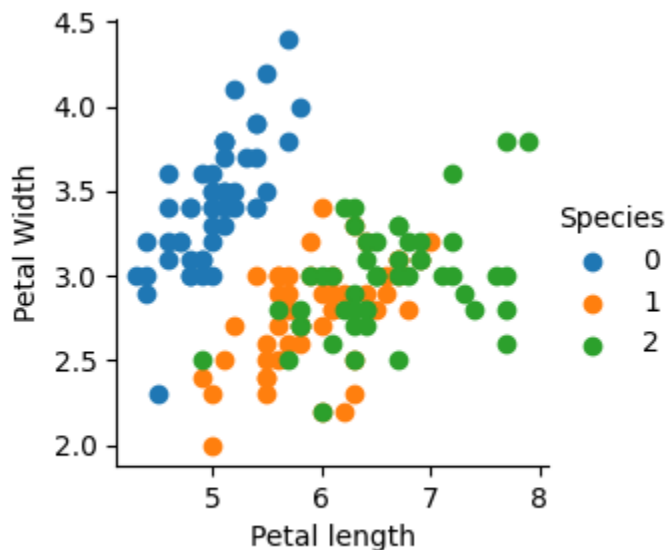


- 
- 탐색적 데이터 분석 (EDA)
  - 목적
    - 1. 데이터 이해: 데이터 분포, 평균, 분산 등 기본 통계 파악
    - 2. 이상치 및 결측치 탐지
    - 3. 패턴 파악: 변수 간 상관관계 및 패턴 시각화
    - 4. 분석 방향 탐색
  - 방법: 기초 통계량 확인, 데이터 시각화, 결측치 및 이상치 처리, 상관관계 분석

- 실습
- Iris dataset에 대한 EDA 수행
- 많은 경우 데이터를 불러온 후에 Pandas의 dataframe으로 저장하여 사용
- 이유: 시각화 및 분석이 쉬움

```
# iris dataset 불러오고 pandas의 dataframe으로 만듦
dataset=load_iris()
data=pd.DataFrame(dataset['data'],columns=["Petal length","Petal Width","Sepal Length","Sepal Width"])
```

- Box plot과 산점도 그래프로 데이터 분포 파악 가능
- 또한, 이상치 파악 가능. Iris dataset은 이상치가 많은 데이터는 아님



## ■ 실습 문제 2

## ■ Wine quality dataset에 대하여 EDA를 수행해보자

## ■ Wine Quality - UCI Machine Learning Repository

1. 고정 산도 (Fixed Acidity): 비휘발성 산의 양 (주로 타르타르산)
2. 휘발성 산도 (Volatile Acidity): 아세트산 등의 휘발성 산 비율 (맛에 영향)
3. 구연산 (Citric Acid): 신맛을 더해주는 천연 산 성분
4. 잔류 설탕 (Residual Sugar): 발효 후 남은 당분 (단맛에 영향)
5. 염화물 (Chlorides): 소금 성분
6. 자유 아황산 (Free Sulfur Dioxide): 미생물 억제와 산화를 방지하는 데 사용
7. 총 아황산 (Total Sulfur Dioxide): 자유 및 결합된 아황산의 총량
8. 밀도 (Density): 와인의 밀도 (알코올, 당 함량과 연관)
9. pH: 와인의 산도
10. 황산염 (Sulphates): 풍미를 강화하는 첨가물
11. 알코올 (Alcohol): 알코올 함량 (%)

### 타겟 변수(Target):

- 품질 (Quality): 와인의 품질을 0~10 사이의 점수로 평가한 값 (3명 이상의 블라인드 테스터의 평균 값으로 결정).

## ■ Red wine과 white wine의 데이터 개수, 특징수 파악

Red Wine Dataset Shape: (1599, 12)  
White Wine Dataset Shape: (4898, 12)

## ■ red wine 과 white wine의 통계치 출력

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149	10.422983	5.63602
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	1.065668	0.80756
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3.00000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000	5.00000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.200000	6.00000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.100000	6.00000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000	8.00000

## ■ 결측값 분석

```
print(df.isnull().sum())
```

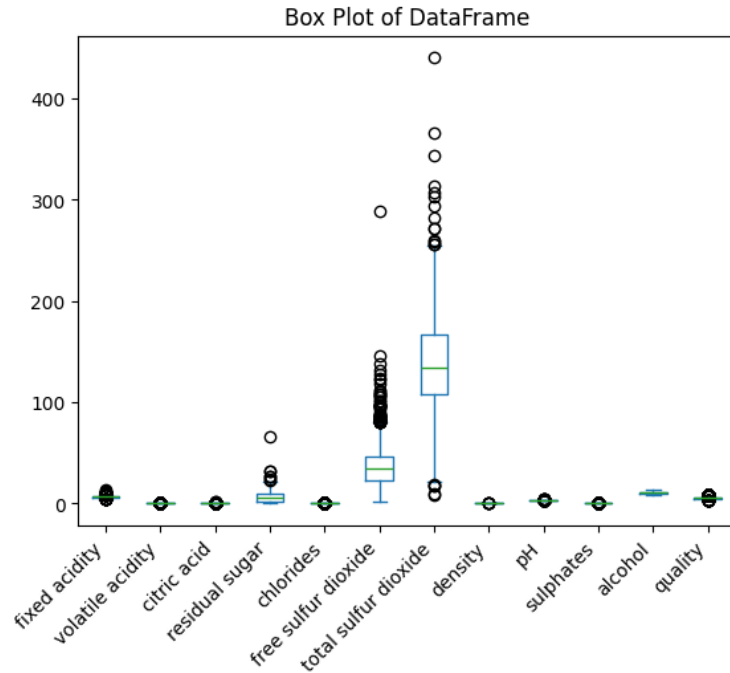
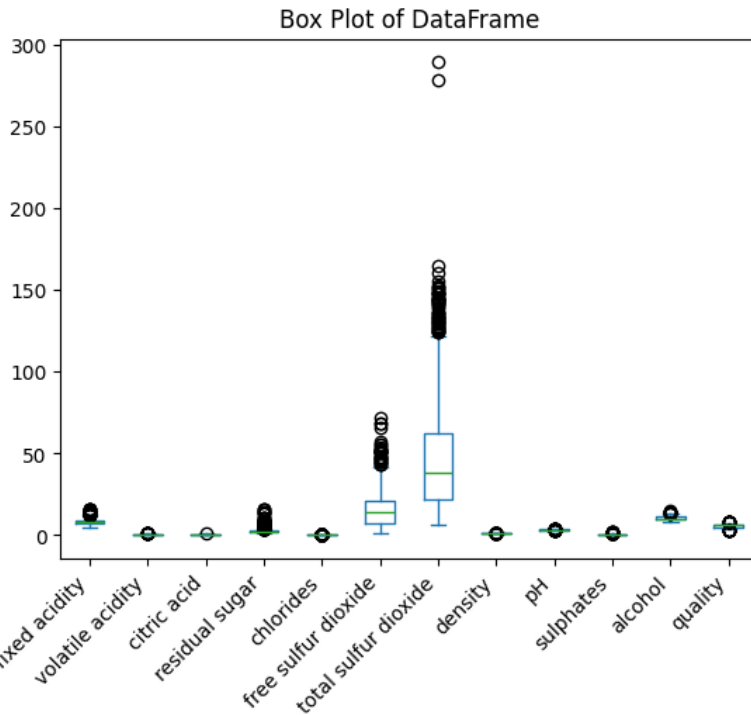
## ■ Real data는 결측값 존재 가능

```
print(red_wine.isnull().sum())
```

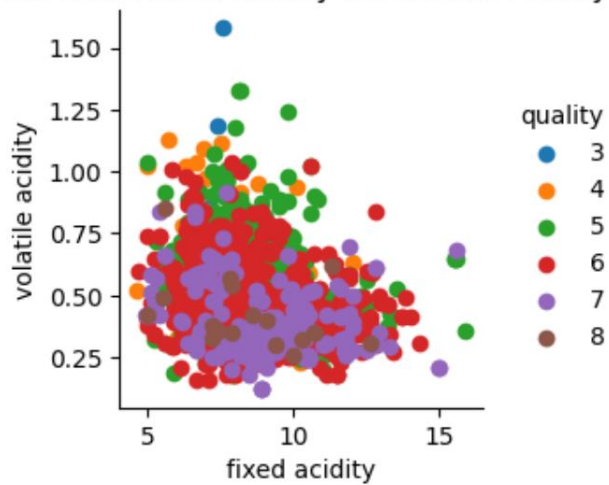
fixed acidity	0
volatile acidity	0
citric acid	0
residual sugar	0
chlorides	0
free sulfur dioxide	0
total sulfur dioxide	0
density	0
pH	0
sulphates	0
alcohol	0
quality	0
dtype: int64	

## ■ red wine 과 white wine의 box plot 그림

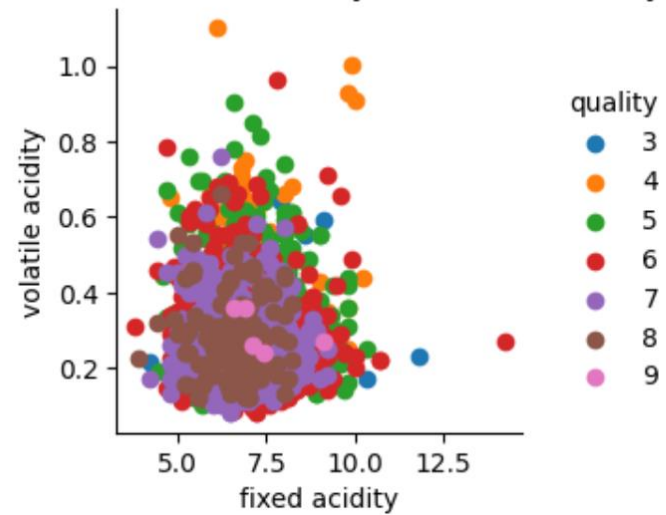
- `plt.xticks(rotation=45, ha='right')` # 45도 회전,
- 이상치가 있는 특징들 파악 가능. 이상치를 제외할 수도



Red Wine: Fixed Acidity vs. Volatile Acidity



White Wine: Fixed Acidity vs. Volatile Acidity

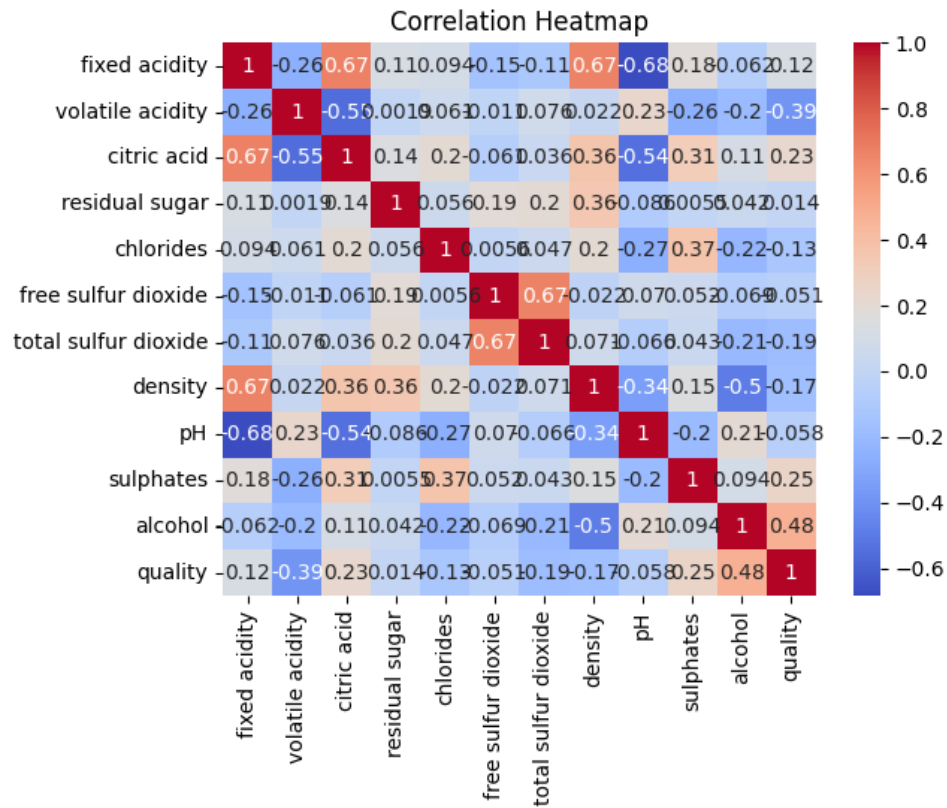




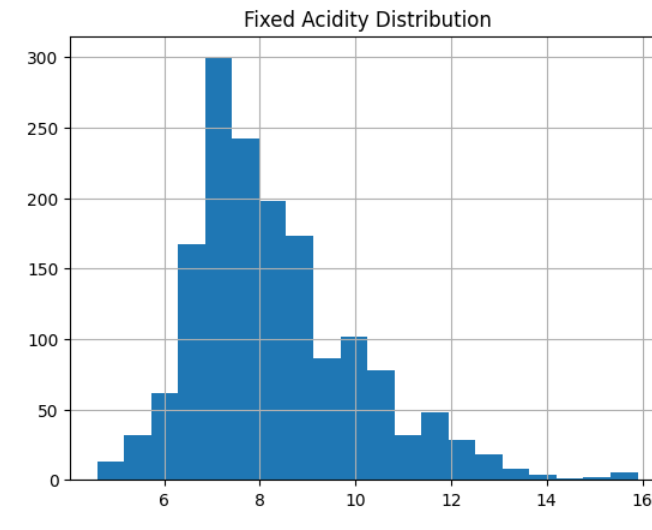
- 
- 각 변수 (특징)간의 상관관계 확인 가능
  - Pandas의 'corr' 는 특징간 상관관계를 보여준다. -1에서 1사이의 값을 갖는다

값의 범위:

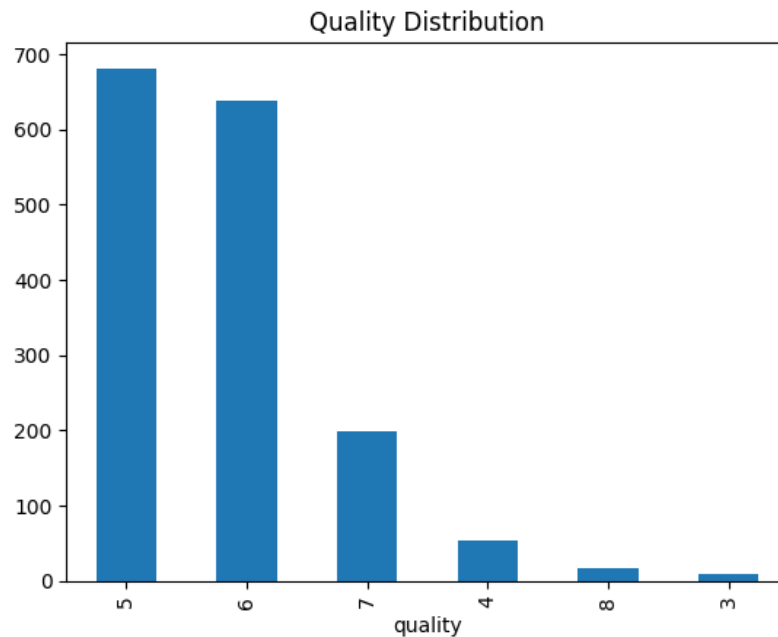
- 1: 완전한 양의 상관관계 (한 변수가 증가하면 다른 변수도 증가).
- -1: 완전한 음의 상관관계 (한 변수가 증가하면 다른 변수는 감소).
- 0: 선형 상관관계가 없음 (두 변수 간에 아무런 선형적 관계가 없음).



- 각각의 데이터가 정규분포를 따르는지, 왜곡되어 있는지 각 특징의 histogram을 그려서 파악 가능
- “fixed acidity” 특징의 경우 정규분포임을 확인 가능



- Class인 품질의 분포 파악
- Class 분포가 중간 quality에 몰려있는 것을 확인. 아주 높거나 낮은 quality의 데이터 개수는 매우 적음을 확인 가능



- 이러한 경우, **quality**를 고급/중급/저급 와인으로 범주화해서도 분석 가능

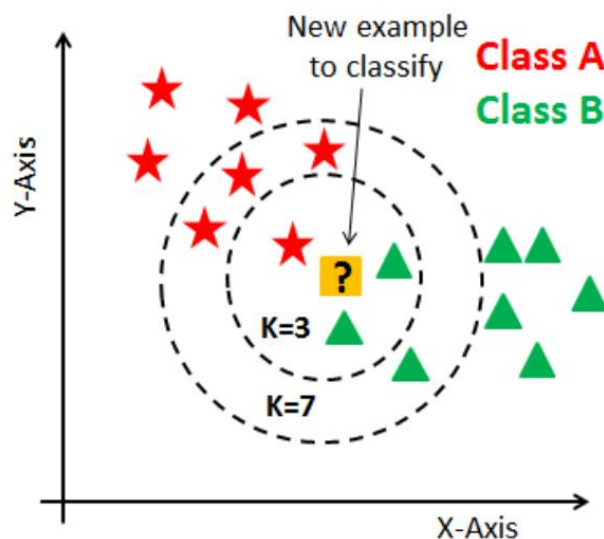
```
red_wine['quality_label'] = df['quality'].apply(lambda x:  
'High' if x >= 7 else ('Medium' if x == 6 else 'Low'))  
sns.countplot(x='quality_label', data=df)  
plt.title("Quality Label Distribution")  
plt.show()
```

---

특강 내용

# 거리기반 ML 모델: K-NN

- **K-Nearest Neighbor (K-NN): 지도 학습 방법의 하나**
- 학습 데이터와 새 데이터를 비교하여 가장 가까운 K 개의 이웃 (neighbors)를 기반으로 예측하거나 분류하는 간단하고 직관적인 알고리즘
- 아래 ?의 경우 K=3이면 **CLASS B**로 분류



## ■ K-NN의 작동 원리

1. 훈련 데이터 준비: 각 데이터는 feature와 label 포함
2. 거리 계산: 새 데이터가 주어지면 훈련 데이터와의 거리를 계산 (예: 유클리드 거리)
3. 가장 가까운 K개의 이웃 선택: 거리 기준으로 가장 가까운 K개의 데이터를 선택
4. 분류: K개의 이웃 중 가장 많이 등장하는 class를 다수결로 선택



- 
- **Training data (훈련 데이터):** 머신 러닝 모델을 학습시키는 데 사용되는 데이터
  - **Test data:** 학습된 머신 러닝 모델의 성능을 평가하는데 사용되는 데이터

---

## ■ Iris data 실습

- 
- 1. 데이터 로드
  - Iris 데이터셋을 로드하여 “feature data (x)”와 “label data (y)”를 추출

```
# 1. 데이터 로드  
iris = load_iris()  
X, y = iris.data, iris.target
```

- **2. 데이터 분할 (훈련 80%, 테스트 20%)**
- 전체 데이터의 80%를 학습용, 20%는 학습한 데이터를 평가하는 테스트용으로 쓴다
- 테스트 데이터를 통해 보지 않은 새로운 데이터에 얼마나 잘 동작하는지
- 일반적으로 훈련 데이터와 테스트 데이터를 8:2, 7:3정도로 나눈다
- 마지막의 “random\_state=42”는 재현 가능성을 위해 사용하는 옵션

# 2. 데이터 분할 (훈련 80%, 테스트 20%)

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

- 3. feature 정규화
- “StandardScaler” 데이터를 평균이 0이고 표준편차가 1인 분포로 변환하는 것. Why? 정규 분포 가정
- “Feature scaling through standarization” is an important preprocessing for many machine learning algorithms.
- Feature 값의 범위가 서로 다르더라도 비교할 수 있는 공통적인 기준 마련

```
# 3. 특성 정규화 (Standard Scaling)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

#### ■ 4. K-NN 알고리즘 정의 및 학습

- K=5, 알고리즘에서 사용할 이웃의 수. 가장 가까운 5개의 이웃을 찾고, 이 이웃들의 label을 기반으로 예측
- K-NN은 **비모수적(non-parametric) 분류 방법**입니다. 즉, 학습할 때 모델을 만들어 놓지 않고, 데이터를 기반으로 즉시 예측을 수행하는 방식

```
# 4. KNN 분류기 정의 및 학습
k = 5 # K값 설정
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, y_train)
```

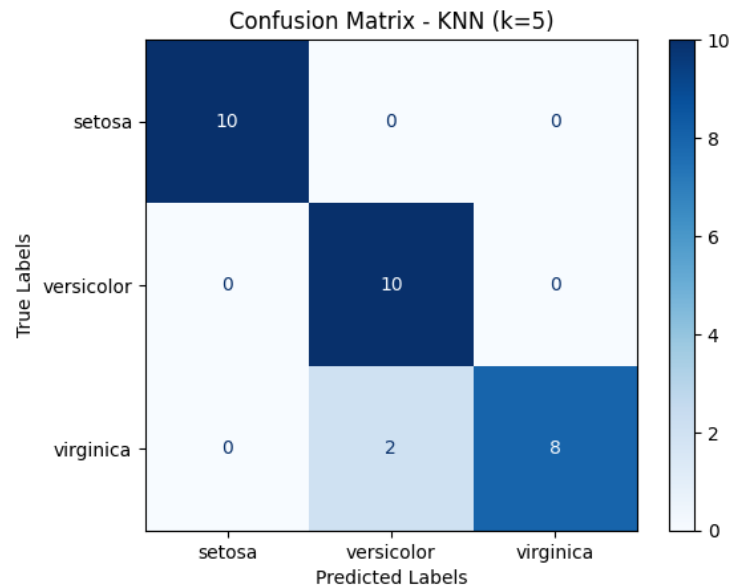
#### ■ 5. 모델 예측

```
# 5. 모델 예측
y_pred = knn.predict(X_test)
```

## ■ 6. 평가 지표: 혼동 행렬 계산 및 시각화

```
# 6. 혼동 행렬 계산 및 시각화
conf_matrix = confusion_matrix(y_test, y_pred)
disp = ConfusionMatrixDisplay(confusion_matrix=conf_matrix, display_labels=iris.target_names)
```

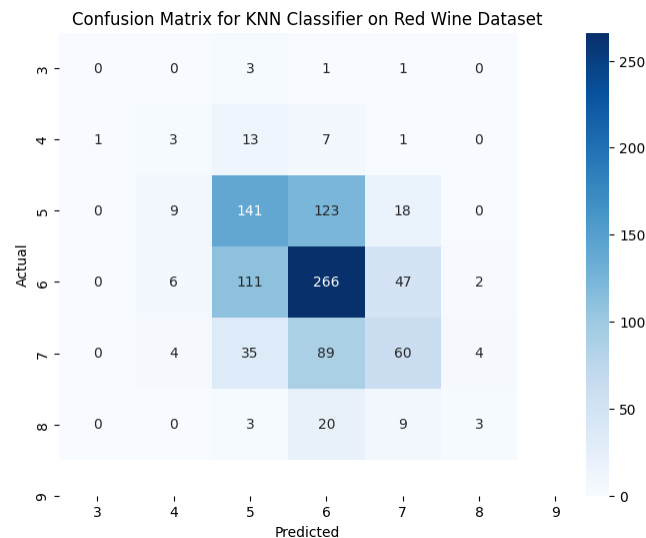
- **성능 평가 지표 예: Confusion matrix**
- 각 행: 실제 class, 각 열: 예측 class
- Test sample 30개 중에 28개는 올바르게 예측 (대각선 부분)
- 정확도:  $\frac{28}{30} * 100 = 93.333 \dots$





### ■ 실습문제 3

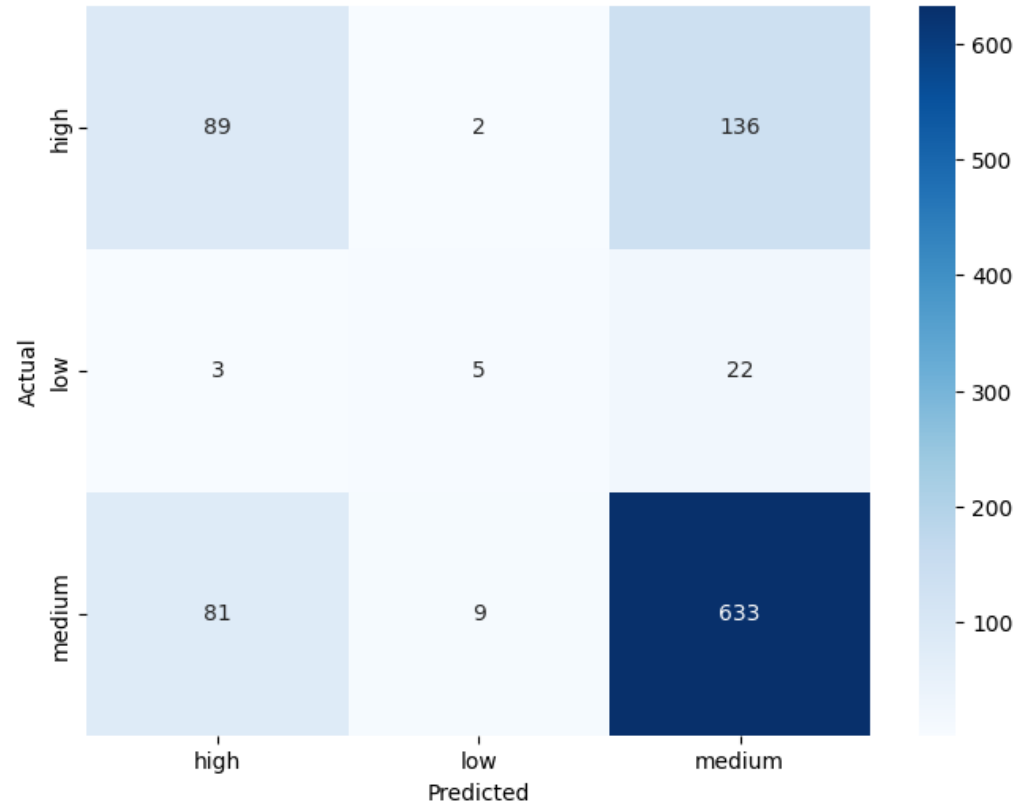
- 앞에서 사용한 wine quality dataset 중에 white wine에 대하여 KNN 모델 학습 후에 예측하고 결과를 confusion matrix로 출력해보자



## ■ 이번에는 QUALITY를 3단계로 나누어서 다시 학습 및 예측해보자

```
# wine quality 한번 더 3단계로 target class 나눔  
# 품질을 기준으로 범주화 (예: 3~5: 낮음, 6: 중간, 7~8: 높음)  
y = y.apply(lambda q: 'low' if q <= 4 else 'high' if q >= 7 else 'medium')
```

Confusion Matrix for KNN Classifier on white Wine Dataset

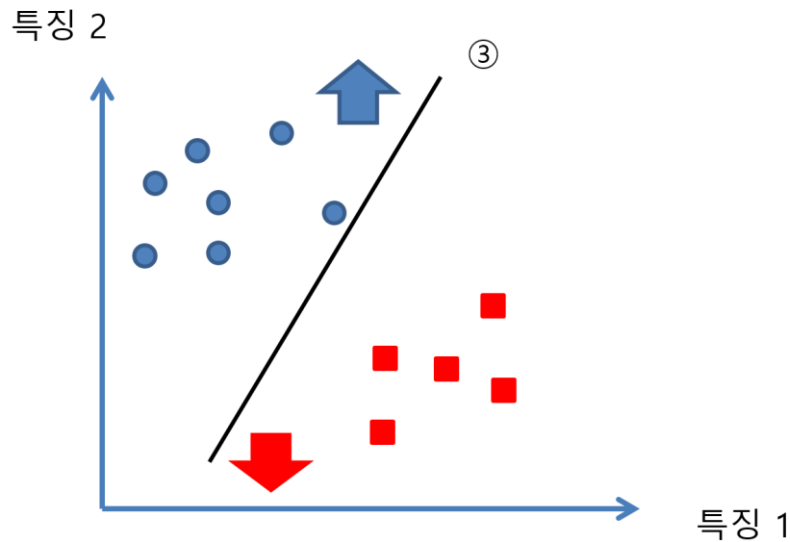


---

## 특강 내용

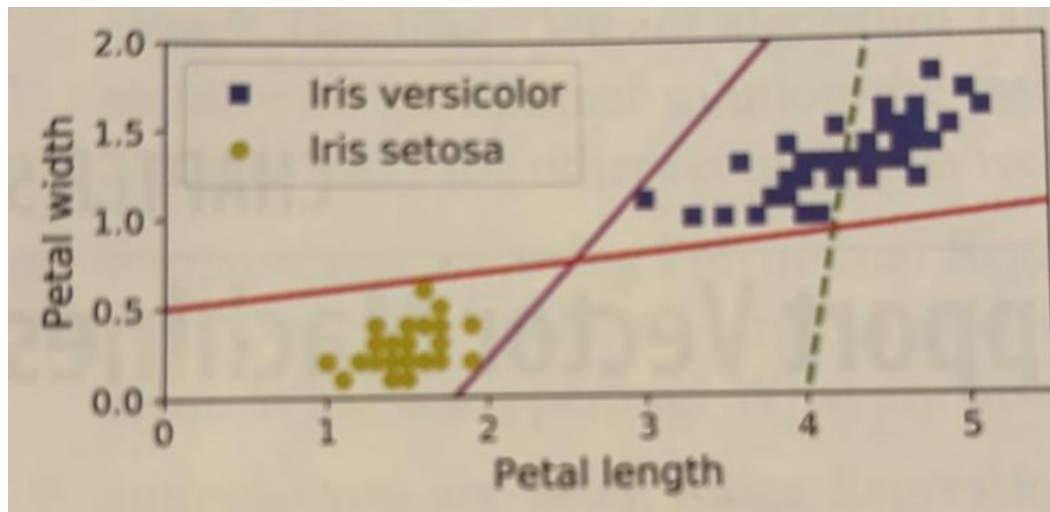
# 거리기반 ML 모델: SVM (서포트 벡터 머신)

- **Support vector machine (SVM)**
- 각 class간의 “마진 (margin)”을 최대화하는 결정 경계를 찾는 지도 학습 방법
- 기본적인 방법은 직선 (선형)형태의 결정 경계를 사용

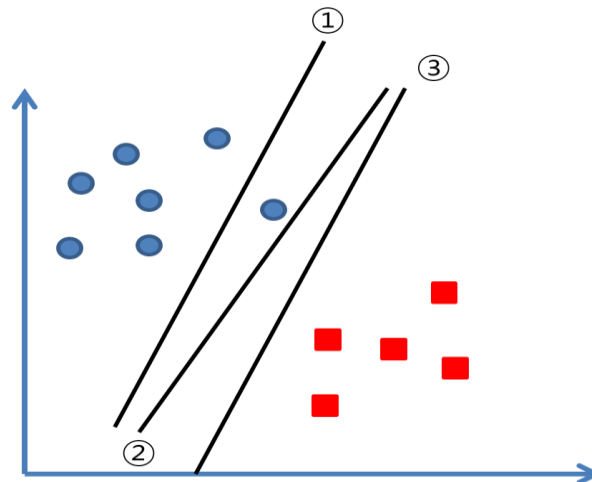


특징 1

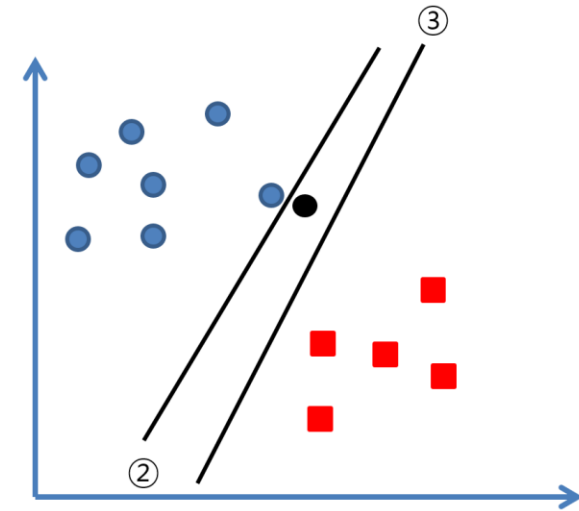
- 직선 형태의 선형 SVM 분류기 예
- Iris dataset에서 2개의 특징 “petal length”와 “petal width”로 2개의 class를 선형 분리한 예
- 2개의 서로 다른 선형 분류기 예 (빨간색)



- 아래 세가지 선형분리기의 성능에 대해 살펴보자
- 직선의 위쪽은 파란색원, 아래쪽은 빨간색 사각형으로 분류하고자 한다
- 여기서 샘플 데이터 들은 학습 데이터 이다
- 선형 분류기 1: 샘플 하나를 틀리게 분류하므로 좋지 않은 분류기이다
- 선형 분류기 2: 모든 샘플을 제대로 분리하였고 오류가 없다
- 선형 분류기 3: 모든 샘플을 제대로 분리하였고 오류가 없다
- 즉, 선형 분류기 2와3은 training 데이터에 대해서는 둘 다 오류가 없다



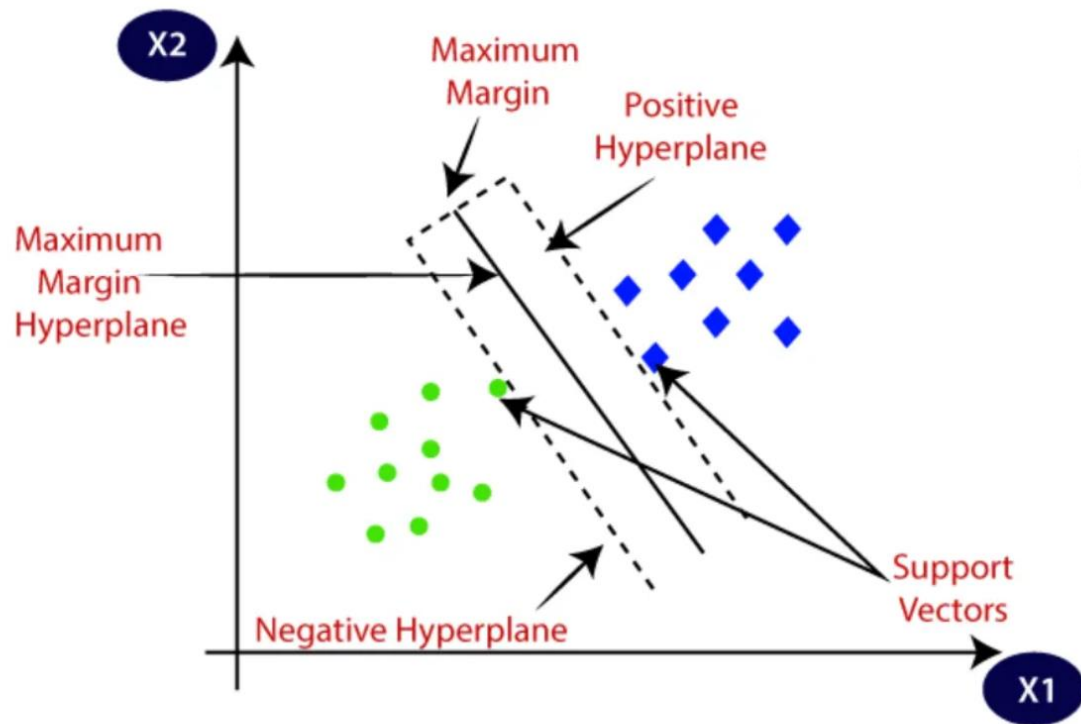
- 하지만, 선형분류기 2와 3은 성능이 같지 않다.
- machine learning에서 분류기의 목적은 **미래의 데이터가 들어왔을 때 이 데이터가 어떤 분류에 속하는지 예측**하는 것이기 때문이다
- 예를 들어 검정 데이터가 생성되었을때 선형분류기 2는 빨간색으로 분류하지만 선형분류기 3은 파란색으로 분류 한다
- Q) 그렇다면 선형 분류기 2와 3중에 어떤것이 분류기로써 더 성능이 좋을까?
- 이 경우 검정색 데이터는 파란색에 가깝기 때문에 파란색으로 분류하는것이 더 타당하다고 추측할 수 있다 (빨간색 점들과는 멀다)
- 그러므로 분류기 2와 분류기 3중에 분류기 3이 더 좋다고 생각할 수 있다.



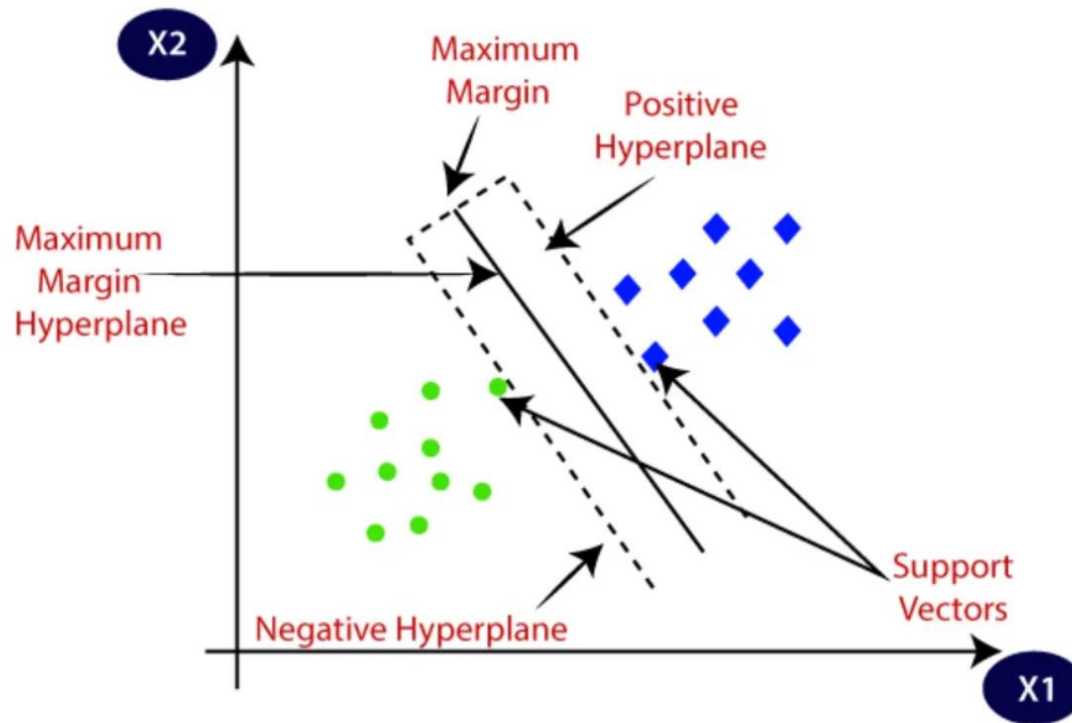


- SVM 분류기의 주요 개념
- 서포트 벡터(Support Vectors):
  - 서포트 벡터는 결정 경계를 결정하는 중요한 데이터 포인트들
- 결정 경계(Decision Boundary):
  - SVM의 결정 경계는 데이터를 분리하는 선(2차원: 직선, 3차원: 평면)
  - 이 경계를 통해 두 클래스를 분리할 수 있습니다.
- 마진(Margin):
  - 마진은 결정 경계와 각 클래스의 서포트 벡터 사이의 거리입니다.
  - SVM은 이 마진을 최대화하는 결정을 내리려고 합니다.
  - 마진이 넓을수록 분류의 일반화 능력이 좋아집니다.

## ■ Support vector, 결정 경계, Margin

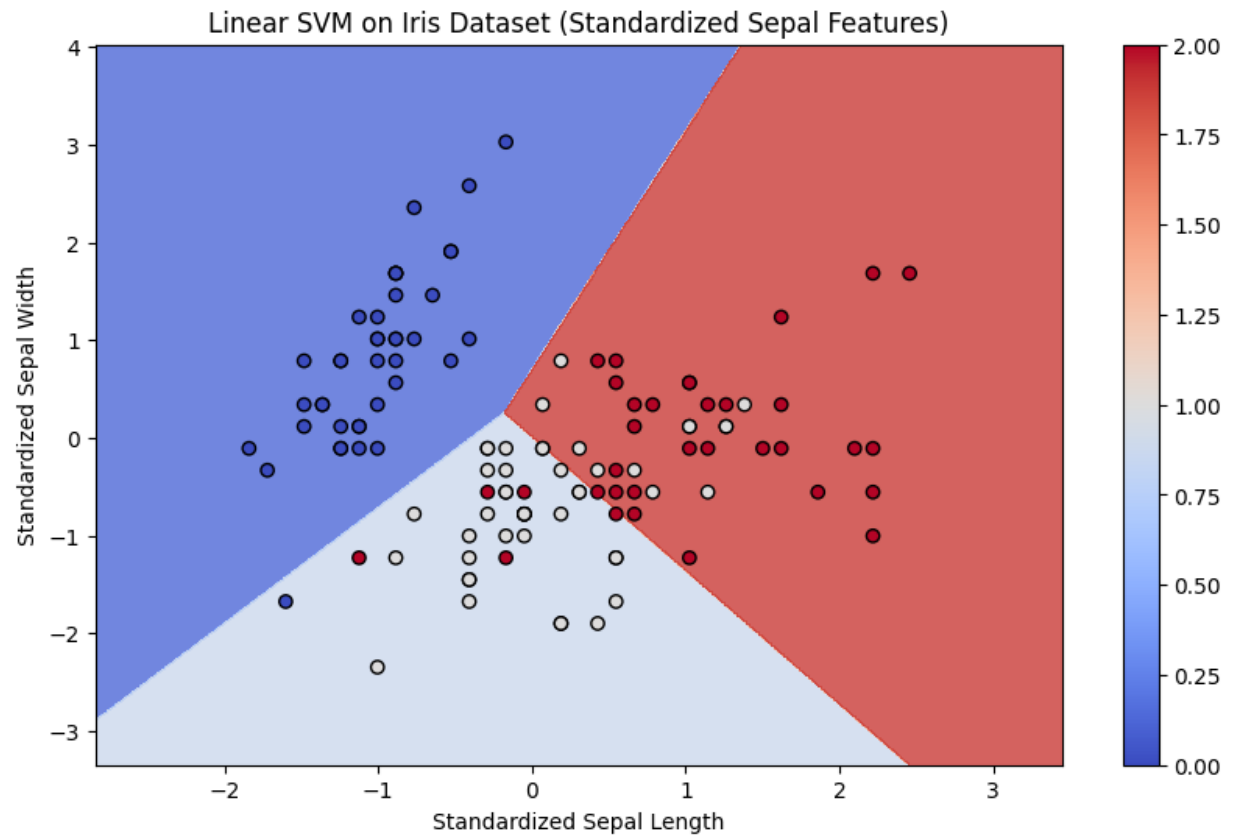


- SVM 분류기의 목적: 결정 경계와 각 클래스의 서포트 벡터 사이의 거리 (margin)을 최대화 하도록 결정 경계를 정함



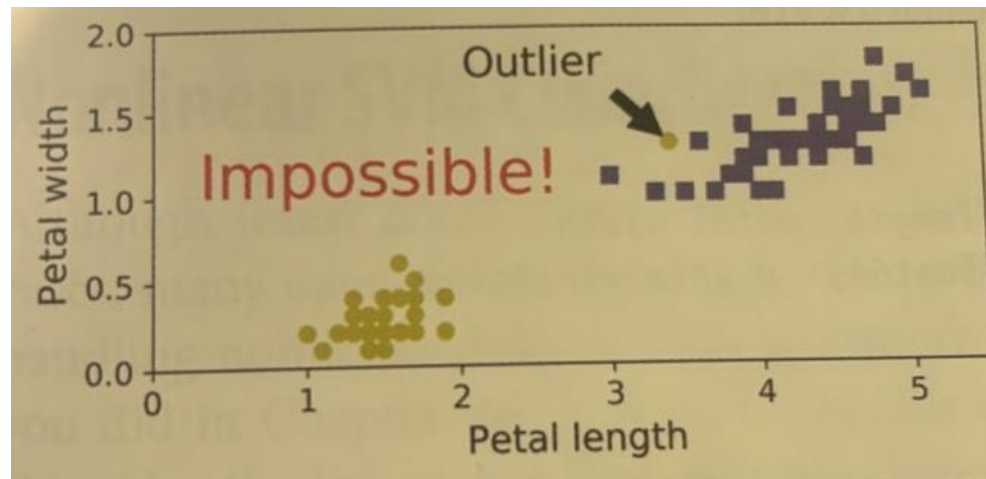
- 
- 선형 SVM의 작동원리 (2개의 class 분류)
  - 1. 데이터 준비
  - 2. 데이터를 두 class로 나누는 경계 정의
  - 3. Margin (경계와 가장 가까운 데이터와의 거리) 계산
  - 목표: margin을 최대화 하는 경계 찾기
  - 4. 최적화 문제 풀 (margin 최대화, 조건, 데이터들이 올바르게 분류)
  - 5. 학습 완료 및 예측
  - 새로운 데이터에 대해서 경계 기준으로 어떤 class에 속하는지 분류

- 실습: 선형 SVM 분류기로 분류 및 시각화
- SVM도 거리 기반이므로 각 feature에서의 거리에 기반한 ML 방법으로 특징에 대한 standardization 전처리 수행
- 주의: 거리기반 ML 방법인 SVM은 특징의 scale에 민감하다. 따라서 특징에 대한 전처리가 필수적이다
- Scikit learn에서는 선형 SVM 분류기를 위한 **‘LinearSVC’함수** (linear support vector classification) 존재



Test Accuracy: 0.73

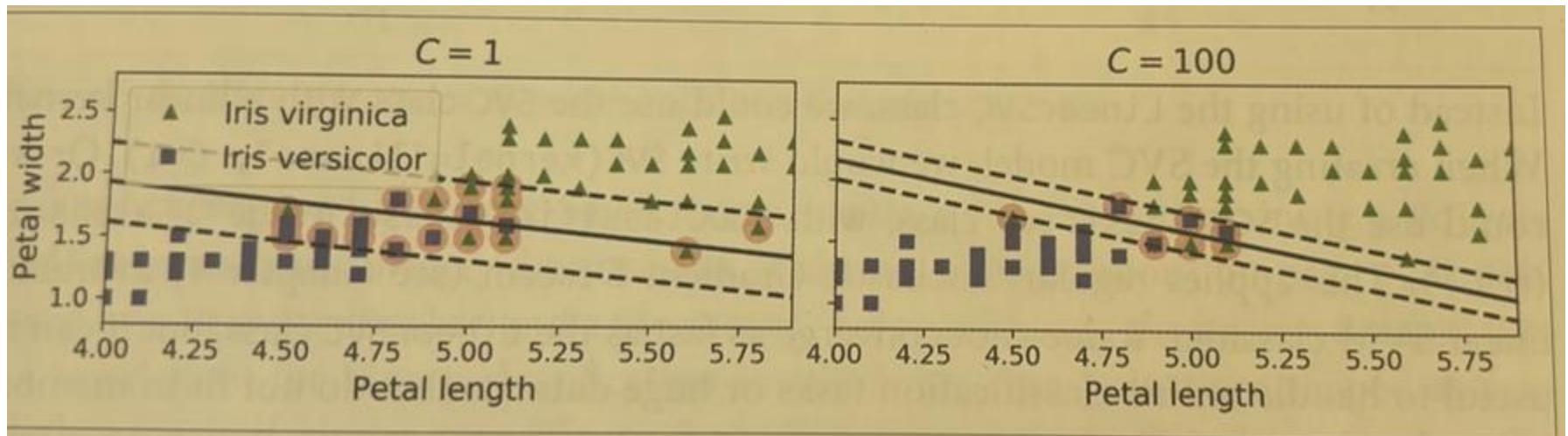
- **Soft margin classification**
- 모든 데이터에 대해서 선형 분류기로 완벽하게 분류하기는 어렵다
- 특히, 아래 예처럼 **outlier**라고 불리는 데이터의 이상치가 존재하는 경우에 그렇다



- 이러한 경우에는 어느 정도의 **margin 위반**이라고 부르는 예외를 허용할 수 밖에 없다
- 이를 **soft margin classification**이라고 한다
- Scikit-learn에서도 SVM 모델을 사용시에 이를 조절할 수 있는 하이퍼파라미터가 있는데 그 중에 대표적인 것이 **C 값**이다
- C값이 크다: margin 위반을 최소화하도록 노력 (엄격)
- C값이 작다: margin 위반을 더 많이 허용 (더 유연)



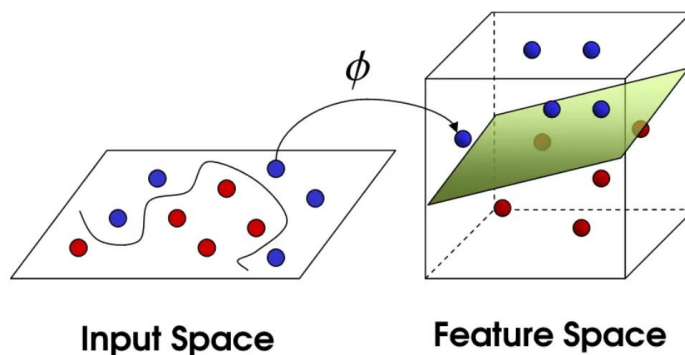
- (왼쪽) 작은 C값: 많은 수의 margin 위반 허용
- (오른쪽) 큰 C값: 적은 수의 margin 위반 허용



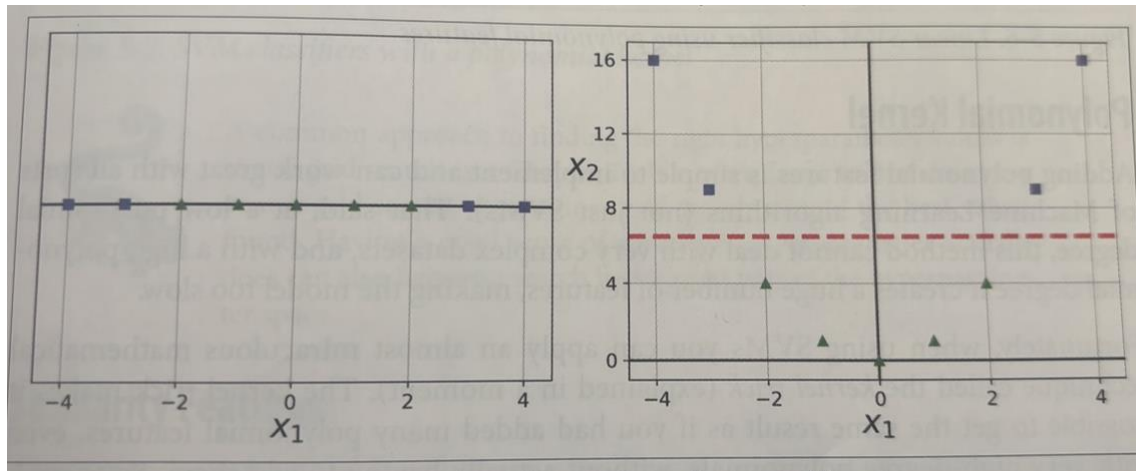
- Margin을 위반은 기본적으로는 좋은 것은 아니다.
- 하지만, ML 모델의 일반화 입장에서는 outlier가 있는 경우에는 이들을 일부 허용할 수 밖에 없다
- 어차피 미래의 test 데이터를 예측하는 것이 목표!
- 즉, 하이퍼 파라미터인 C값을 조절하여 **선형적으로 완벽히 분리되지 않는 데이터에도 적용 가능하면서 noise가 있는 데이터에도 안정적으로 동작하게 할 수 있다**
- 이를 ML에서는 **regularization (규제)**라고 부른다.
- 많은 ML 모델에서는 그래서 규제 파라미터가 있다

## ■ 비선형 SVM 분류기

- 앞에서 배운 선형 SVM 분류기는 많은 경우에 높은 분류기 정확도를 보인다. 하지만, 데이터 분포가 모두 직선 (선형) 분류기로 분류가 가능하지는 않다
- 이러한 경우 데이터를 **고차원 공간으로 mapping** 하여 그 후에 직선 (선형)으로 분류하도록할 수 있다



- 간단한 idea
- 1D 데이터에 특징을 추가.  $x_2 = (x_1)^2$
- 2D 데이터로 바뀐뒤 완벽히 선형(직선) 분리 가능

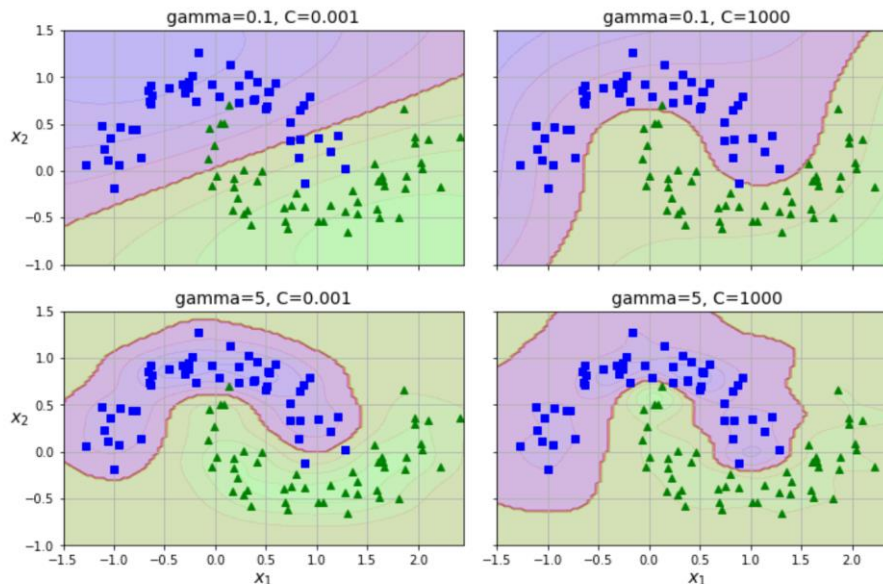


- **Kernel trick**
- 원래 데이터를 더 고차원의 특징 공간으로 mapping한 뒤에 그 공간에서 선형 결정 경계를 찾는 방법으로 방금 전 예에서 **간단한 수식으로 고차원 mapping** 이를 **kernel trick** 이라고 한다
- 이외에도 비선형 SVM 분류기를 위한 다양한 kernel trick이 존재 한다. 가장 많이 쓰이는 것은 **‘rbf’ kernel**

	Scikit-learn 함수
선형 SVM 분류기	LinearSVC
비선형 SVM 분류기	SVC

- Kernel trick에서의  $\gamma$  (gamma) 파라미터
- 앞에서의 C 파라미터와 함께 중요한 파라미터
- $\gamma \uparrow$  : 보다 복잡한 결정 경계 생성 (overfitting 가능성 상승)
- $\gamma \downarrow$  : 더 단순한 결정 경계 생선 (underfitting 가능성 상승)
- Tuning 이유: 데이터의 분포에 따라 적절한  $\gamma$  찾아야

$\gamma, C$ 값이 모두 작음



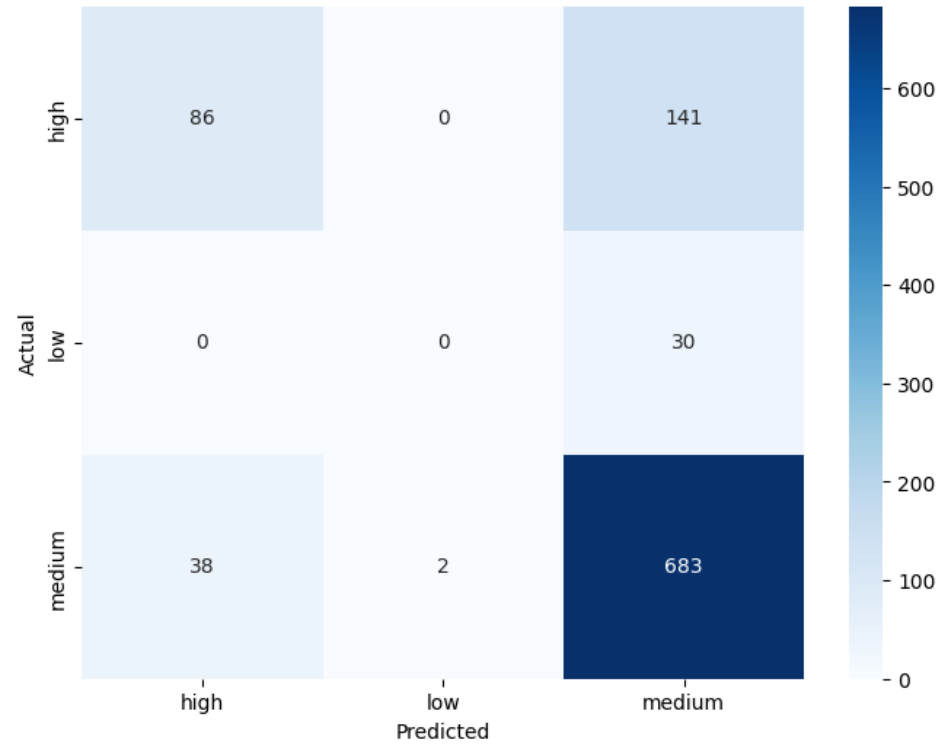
$\gamma, C$ 값이 모두 큼

## ■ 실습문제 4

- 앞에서 사용한 wine quality dataset 중에 white wine에 대하여 standarization으로 정규화후에 선형 SVM 혹은 비선형 SVM 모델 학습 후에 예측하고 결과를 confusion matrix로 출력해보자

```
# wine quality 한번 더 3단계로 target class 나눔  
# 품질을 기준으로 범주화 (예: 3~4: 낮음, 6: 중간, 7~9: 높음)  
y = y.apply(lambda q: 'low' if q <= 4 else 'high' if q >= 7 else 'medium')
```

Confusion Matrix for KNN Classifier on white Wine Dataset



0.7846938775510204



---

## 특강 내용

# Pre-processing: standardization

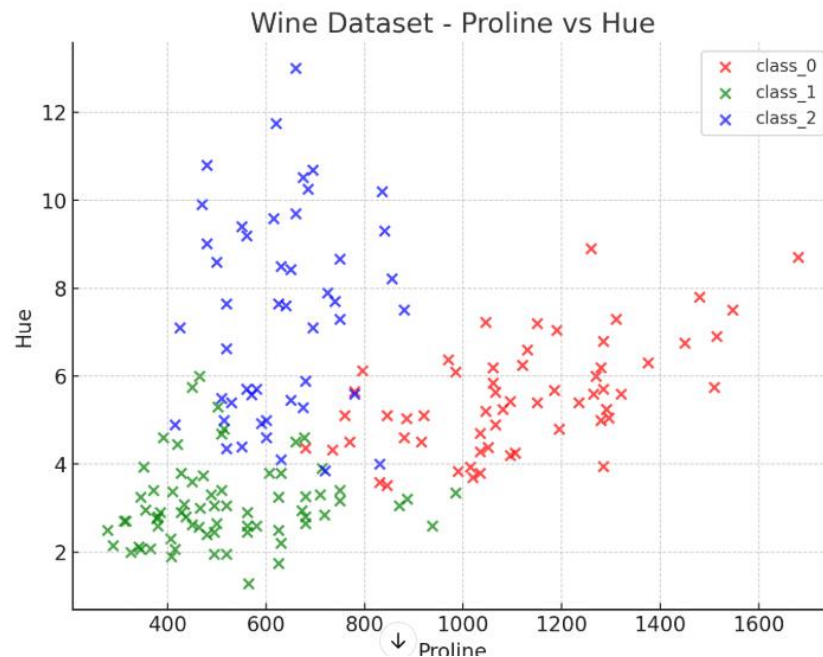
- 많은 머신 러닝 알고리즘은 앞에서 배운 **"feature scaling through standarization"**이 매우 중요한 전처리 방법이다
- 앞의 EDA에서 보았지만 많은 데이터는 정규 분포를 따르고 있다고 가정한다
- 다음과 같은 **"wine recognition dataset"**을 통하여 **"feature scaling through standarization"**의 중요성을 한번 더 살펴보자

- UCI machine learning repository에서 다운로드 가능
- [UCI Machine Learning Repository](#)
- 총 178개의 샘플
- 3개의 와인 품종 (클래스 0, 1, 2)

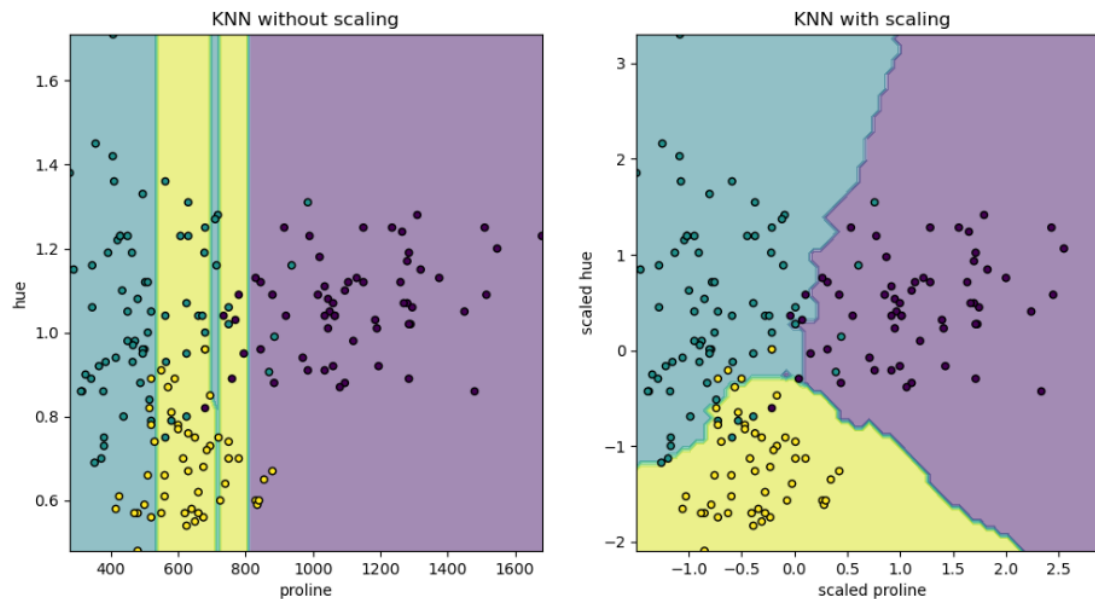
특징 (Features):

- 총 13개의 연속형 변수로 구성된 화학적 측정값.
- 주요 특징:
  - Alcohol: 알코올 함량.
  - Malic acid: 말산 함량.
  - Ash: 회분 함량.
  - Alkalinity of ash: 회분의 알칼리도.
  - Magnesium: 마그네슘 함량.
  - Total phenols: 총 페놀 함량.
  - Flavanoids: 플라보노이드 함량.
  - Nonflavanoid phenols: 비플라보노이드 페놀 함량.
  - Proanthocyanins: 프로안토시아닌 함량.
  - Color intensity: 색 강도.
  - Hue: 색조.
  - OD280/OD315 of diluted wines: 희석된 와인의 OD280/OD315 비율.
  - Proline: 프롤린 함량.

- “proline”:0~1000 사이, “hue”:1에서 10 사이
- KNN 같은 거리 기반의 ML 모델에서는 **standardization**과 같은 방법의 전처리가 필수
- 그렇지 않으면 특정 feature에 영향을 더 받게 된다



- (실습) KNN without standardization, with standardization 비교
- 어떤 것이 맞는걸까?

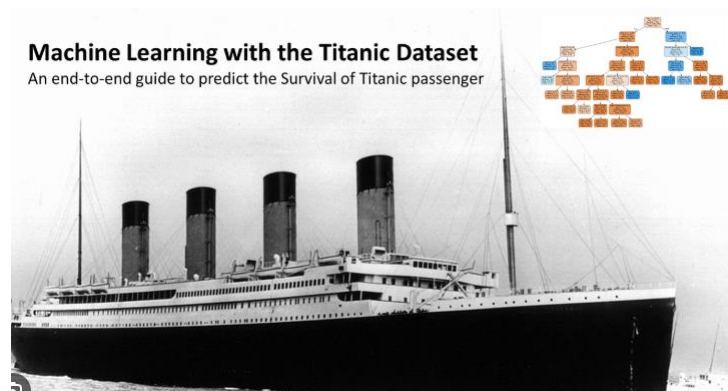


---

## 특강 내용

# Pre-processing: 범주형 데이터 처리

- **Titanic dataset**
- 1912년 타이타닉 호가 침몰된 사고와 관련된 데이터를 담고 있고 주로 “생존자 예측“에 관련된 머신 러닝 학습에 사용
- 각 승객에 대한 여러 특징을 담고 있고, 이를 기반으로 승객이 살았는지 사망했는지 (class)를 예측



---

## 주요 컬럼 (특징)

타이타닉 데이터셋에 포함된 주요 변수들은 다음과 같습니다:

1. **Survived:** 승객의 생존 여부 (0 = 사망, 1 = 생존)
2. **Pclass:** 승객의 클래스 (1, 2, 3) - 1등급, 2등급, 3등급
3. **Name:** 승객의 이름
4. **Sex:** 성별 (male, female)
5. **Age:** 나이
6. **SibSp:** 승객과 함께 탑승한 형제/배우자 수
7. **Parch:** 승객과 함께 탑승한 부모/자녀 수
8. **Ticket:** 티켓 번호
9. **Fare:** 운임 요금
10. **Cabin:** 객실 번호
11. **Embarked:** 탑승한 항구 (C = Cherbourg, Q = Queenstown, S = Southampton)



- 실제 dataset
- 다운로드: [Titanic - Machine Learning from Disaster | Kaggle](#)

lua 코드 복사

PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Fare
1	1	Braund, Mr. Owen Harris	male	22	1	0	7.25
2	1	Cumings, Mrs. John Bradley	female	38	1	0	71.0
3	3	Heikkinen, Miss. Laina	female	26	0	0	7.92

---

- **Categorical features**

- 범주형 데이터를 의미하며, 이는 숫자가 아닌 어떤 고유한 값으로 구성된 데이터
- 앞의 titanic dataset 참고. 예: 성별, “male”, “female”
- 이러한 데이터는 encoding이라는 작업을 통하여 수치형 데이터로 변환. 그래야 ML 모델의 학습이 가능
- 가장 많이 쓰이는 2가지 방법 소개

- **1. Label encoding**
- 각 범주형 값을 고유한 숫자 값으로 변환
- Titanic dataset을 예로 들면 ‘male’은 0으로 ‘female’은 1으로
- 단, 숫자 간의 크기를 ML 모델이 학습할 수 있음

```

PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7.25,,S
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thayer)",female,38,1,0,PC 17599,71.2833,C85,C
3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3101282,7.925,,S
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",female,35,1,0,113803,53.1,C123,S
5,0,3,"Allen, Mr. William Henry",male,35,0,0,373450,8.05,,S
6,0,3,"Moran, Mr. James",male,,0,0,330877,8.4583,,Q
7,0,1,"McCarthy, Mr. Timothy J",male,54,0,0,17463,51.8625,E46,S
8,0,3,"Palsson, Master. Gosta Leonard",male,2,3,1,349909,21.075,,S
9,1,3,"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)",female,27,0,2,347742,11.1333,,S
10,1,2,"Nasser, Mrs. Nicholas (Adele Achem)",female,14,1,0,237736,30.0708,,C
11,1,3,"Sandstrom, Miss. Marguerite Rut",female,4,1,1,PP 9549,16.7,G6,S
12,1,1,"Bonnell, Miss. Elizabeth",female,58,0,0,113783,26.55,C103,S
13,0,3,"Saunderscock, Mr. William Henry",male,20,0,0,A/5. 2151,8.05,,S
14,0,3,"Andersson, Mr. Anders Johan",male,39,1,5,347082,31.275,,S
15,0,3,"Vestrom, Miss. Hulda Amanda Adolfina",female,14,0,0,350406,7.8542,,S
16,1,2,"Hewlett, Mrs. (Mary D Kingcome) ",female,55,0,0,248706,16,,S
17,0,3,"Rice, Master. Eugene",male,2,4,1,382652,29.125,,Q

```

- 
- Scikit-learn에는 'labelencoder' 함수 존재
  - Encode target labels with value between 0 and n\_classes-1.

```
from sklearn.preprocessing import LabelEncoder  
le = LabelEncoder()  
data['Sex'] = le.fit_transform(data['Sex'])
```

- 실습
- 아래는 encoding 결과

	PassengerId	Name	Sex	Age	Fare
0	1	Braund, Mr. Owen Harris	1	22	7.2500
1	2	Cumings, Mrs. John Bradley	0	38	71.2833
2	3	Heikkinen, Miss. Laina	0	26	7.9250
3	4	Futrelle, Mrs. Jacques Heath	0	35	53.1000

## ■ 2. one-hot encoding

- 각 범주를 이진 벡터로 변환. 각 범주를 새로운 열로 만들어 해당 범주에 해당하면 1, 아니면 0을 할당

id	color
1	red
2	blue
3	green
4	blue



id	color_red	color_blue	color_green
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0

- 실습: “embarked” 특징에 대해서 one-hot encoding 수행해봄
- “embarked”: 승객이 탑승한 항구

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7.25,,S
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thayer)",female,38,1,0,PC 17599,71.2833,C85,C
3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3101282,7.925,,S
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",female,35,1,0,113803,53.1,C123,S
5,0,3,"Allen, Mr. William Henry",male,35,0,0,373450,8.05,,S
6,0,3,"Moran, Mr. James",male,,0,0,330877,8.4583,,Q
7,0,1,"McCarthy, Mr. Timothy J",male,54,0,0,17463,51.8625,E46,S
8,0,3,"Palsson, Master. Gosta Leonard",male,2,3,1,349909,21.075,,S
9,1,3,"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)",female,27,0,2,347742,11.1333,,S
10,1,2,"Nasser, Mrs. Nicholas (Adele Achem)",female,14,1,0,237736,30.0708,,C
11,1,3,"Sandstrom, Miss. Marguerite Rut",female,4,1,1,PP 9549,16.7,G6,S
12,1,1,"Bonnell, Miss. Elizabeth",female,58,0,0,113783,26.55,C103,S
13,0,3,"Saunderscock, Mr. William Henry",male,20,0,0,A/5. 2151,8.05,,S
14,0,3,"Andersson, Mr. Anders Johan",male,39,1,5,347082,31.275,,S
15,0,3,"Vestrom, Miss. Hulda Amanda Adolfina",female,14,0,0,350406,7.8542,,S
16,1,2,"Hewlett, Mrs. (Mary D Kingcome) ",female,55,0,0,248706,16,,S
17,0,3,"Rice, Master. Eugene",male,2,4,1,382652,29.125,,Q
```

## ■ 실습

	PassengerId	Name	Sex	Age	Fare	₩
0	1	Braund, Mr. Owen Harris	male	22	7.2500	
1	2	Cumings, Mrs. John Bradley	female	38	71.2833	
2	3	Heikkinen, Miss. Laina	female	26	7.9250	
3	4	Futrelle, Mrs. Jacques Heath	female	35	53.1000	

	Embarked_C	Embarked_Q	Embarked_S
0	False	False	True
1	True	False	False
2	False	False	True
3	False	True	False



- 
- **주의:** one hot encoding은 범주형 데이터의 값들 간에 순서나 관계가 없을때 많이 쓰인다
  - 하지만, 범주형 데이터의 범주가 많을 경우 " 차원의 저주 " 에 빠질 수 있음
  - 이러한 경우 대부분 0으로 채워진 값으로 효율성에 문제 생김

---

특강 내용

# Pre-processing: 결측값 채우기

- **Missing value (결측값) 대처**
- 여러 이유로, 많은 실제 데이터셋은 missing value들을 갖고 있다
- 이러한 경우 값이 들어가야 하는 자리에 blanks, NaN등과 같은 표시로 되어 있을 수 있다

Row/ Col	1	2	3	4	5
1	0.24	-0.1		0.18	0.42
2	0.19	-0.22	-0.2	0.12	0.21
3	0.21	0.09	0.57	-0.14	0.29
4	0.76	0.07	0.04	-0.06	0.3
5	0.46	0.12	0.49	-0.42	0.28
6	0.43	-0.23	-0.3	-0.24	0.23
7	0.44	-0.32	0.26	-0.77	0.31
8	0.11	0.03		-0.24	0.36

- Real dataset인 titanic dataset의 예
- 결측치인 빈칸 (blank)이 많음

```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7.25,,S
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thayer)",female,38,1,0,PC 17599,71.2833,C85,C
3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3101282,7.925,,S
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",female,35,1,0,113803,53.1,C123,S
5,0,3,"Allen, Mr. William Henry",male,35,0,0,373450,8.05,,S
6,0,3,"Moran, Mr. James",male,,0,0,330877,8.4583,,Q
7,0,1,"McCarthy, Mr. Timothy J",male,54,0,0,17463,51.8625,E46,S
8,0,3,"Palsson, Master. Gosta Leonard",male,2,3,1,349909,21.075,,S
9,1,3,"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)",female,27,0,2,347742,11.1333,,S
...
```

## ■ 데이터셋에 결측치가 있는지 파악하는 법?

```
print(df.isnull().sum())
```

- 
- **Missing value 대처**
  - 1. Missing value를 포함하고 있는 전체 행이나 열을 버림
  - 문제점: 가치있는 데이터가 상실될 수 있다
  - 2. Impute the missing values by inferring them from the known part

---

- 소개할 두 가지 imputation 전략

- 1. univariate feature imputation

단일 특성을 기반으로 missing value를 채움

1) mean: missing value를 특징의 평균값으로 대체

2) median: missing value를 특징의 중앙값으로 대체

3) most\_frequent: missing value를 특징의 최빈값으로 대체

## ■ 예:

```
# 샘플 데이터 생성 (결측값 포함)
data = {
    'Age': [25, np.nan, 30, np.nan, 35],
    'Salary': [50000, 60000, np.nan, 80000, 90000],
    'Gender': ['Male', 'Female', np.nan, 'Female', 'Male']
}
df = pd.DataFrame(data)
```

	Age	Salary	Gender
0	25.0	50000.0	Male
1	30.0	60000.0	Female
2	30.0	70000.0	Female
3	30.0	80000.0	Female
4	35.0	90000.0	Male



## ■ 2. KNN imputation

- 결측값이 있는 샘플을 주변의 K개의 이웃 값 (거리 기반)으로 대체

```
data = {  
    'Age': [25, np.nan, 30, np.nan, 35],  
    'Salary': [50000, 60000, np.nan, 80000, 90000],  
    'Gender': ['Male', 'Female', 'Female', 'Female', 'Male']  
}
```

	Age	Salary	Gender
0	25.0	50000.0	Male
1	30.0	60000.0	Female
2	30.0	70000.0	Female
3	30.0	80000.0	Female
4	35.0	90000.0	Male

---

## ■ Strategy

- 간단한 데이터이거나 연속형 데이터는 평균이나 중앙값으로 결측값 대체
- 또한, 결측값이 많을 수록 평균/중앙값 대체가 적합
- 범주형 데이터는 최빈값이나 상수 (예: 0)으로 대체
- 복잡한 데이터는 KNN imputation으로 대체하는 경우가 많음