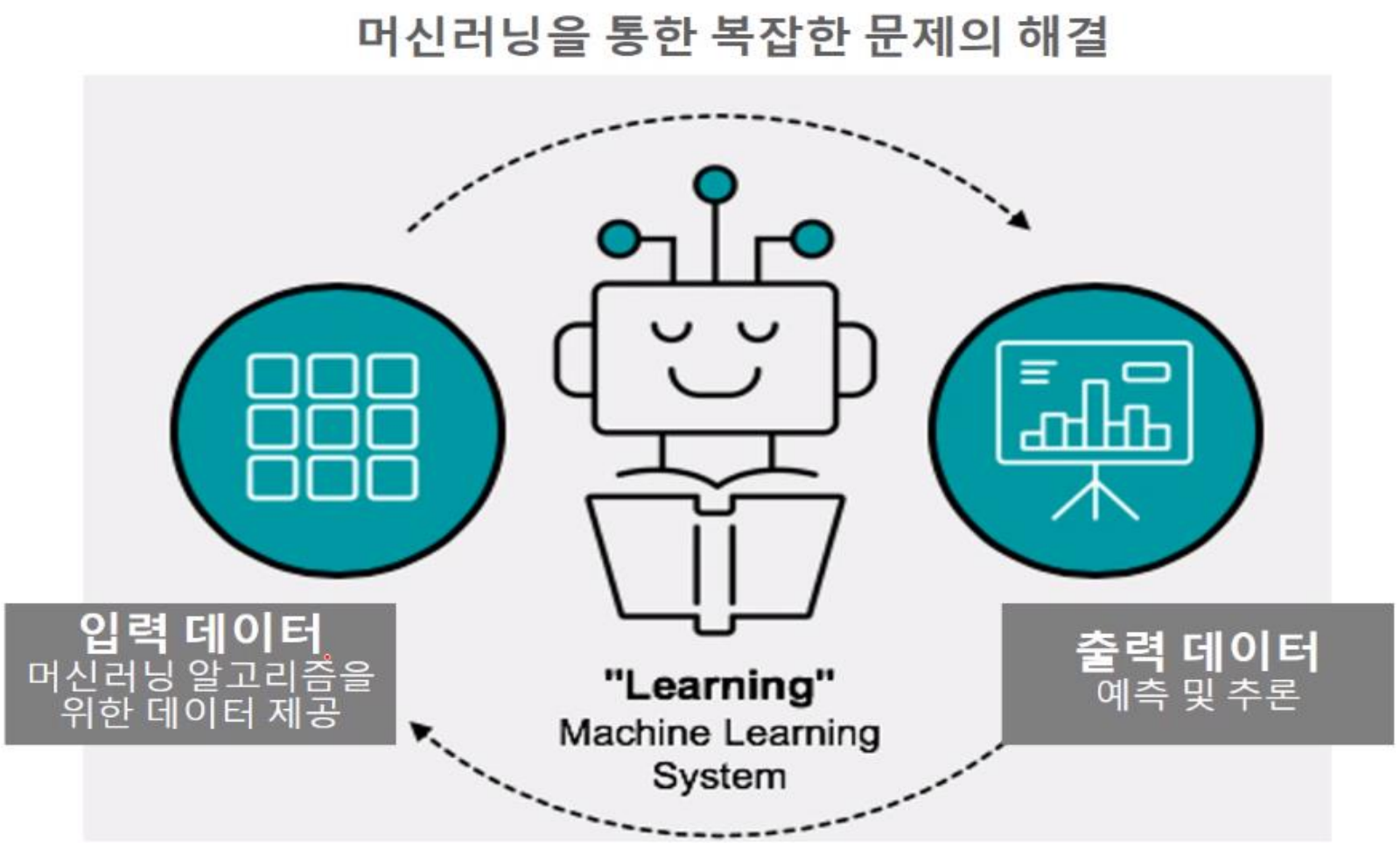
사이킷런으로 시작하는 머신러닝

머신러닝(Machine Learning) 개념



머신러닝(Machine Learning) 개념

- **머신 러닝(Machine Learning) 이란?**

"머신 러닝 또는 기계 학습은 인공 지능의 한 분야로, 컴퓨터가 **명시적 프로그래밍 없이 학습**할 수 있도록 하는 **알고리즘과 기술**을 개발하는 분야를 말한다." - 위키피디아

"Field of study that gives **computers** the ability to **learn** without being **explicitly programmed**." - Arthur Samuel, 1959

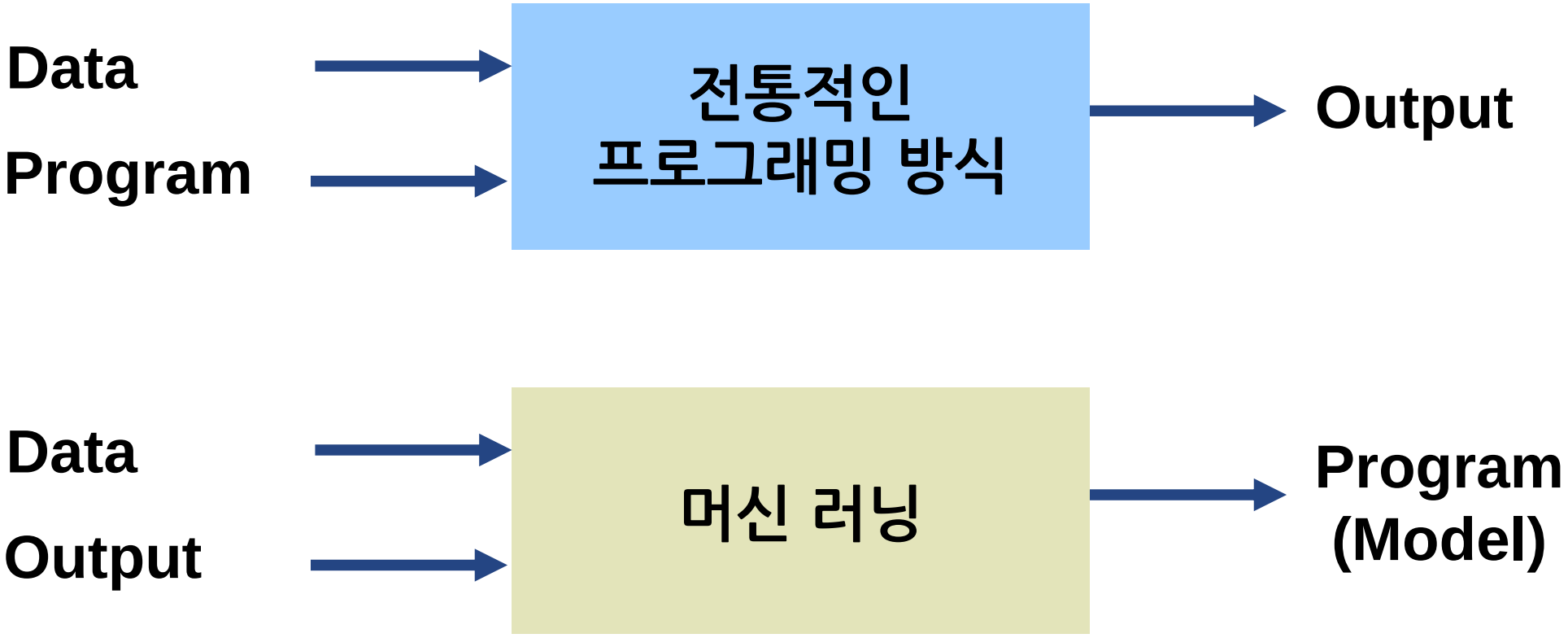
머신 러닝을 보다 **형식화**하여 정의하면 **"환경(Environment, E)"**과의 상호작용을 통해서 축적되는 경험적인 **"데이터(Data, D)"**를 바탕으로 **학습**하여, **"모델(Model, M)"**을 자동으로 구축하고 스스로 **"성능(Performance, P)"**을 향상하는 시스템 이다 (Mitchell, 1997)

Performance(P)

Machine Learning : Data(D) ⟶ Model(M)

머신러닝(Machine Learning) 개념

- 전통적인 프로그래밍 방식과 머신 러닝과의 차이점



머신러닝(Machine Learning) 개념

- 머신 러닝의 종류 - 지도학습, 비지도학습, 강화학습

지도학습

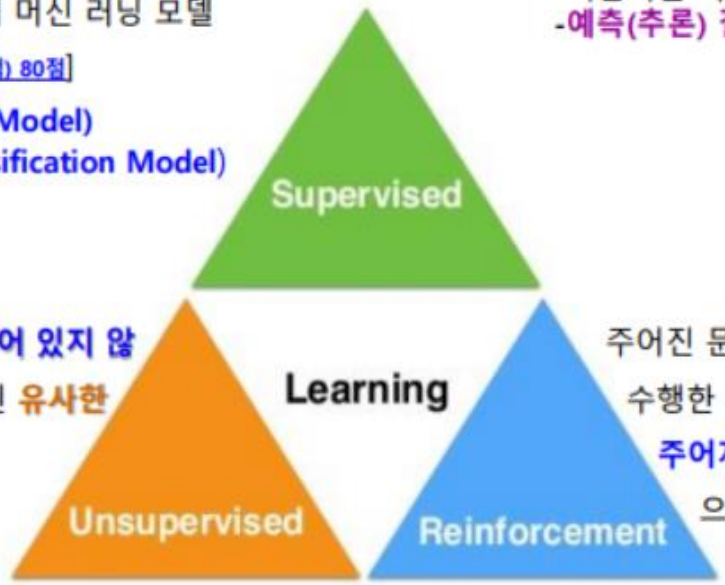
이미 **정답**을 알고(다른 의미로는 **레이블이 되어 있는 데이터**)
있는 **Training Data(학습 데이터)를 이용**하여 머신 러닝 모델
학습시키는 방식 [Xi(공부시간) 2시간 = Yi(시험성적) 80점]
ex : 시험 성적 예측, 주가 예측 **(Regression Model)**
Image Labeling, 스팸 메일 필터링 **(Classification Model)**

-**분류된 데이터(Labeled Data)**
-직접적인 피드백
-**예측(추론) 결과 (미래)**

비지도 학습

정답을 모르는 Training 데이터(즉 **레이블 되어 있지 않은 데이터**)를 이용, 학습하여 데이터에 내재된 **유사한 특성, 패턴, 구조** 등을 발견, 분석
ex : 구글 뉴스 그룹핑(비슷한 뉴스 그룹핑)
Word clustering (비슷한 단어끼리 Clustering Model),
Security

-**분류가 안됨(No Labels)**
-피드백 없음
-**숨겨진 구조 찾기(특징, 패턴, 군집)**



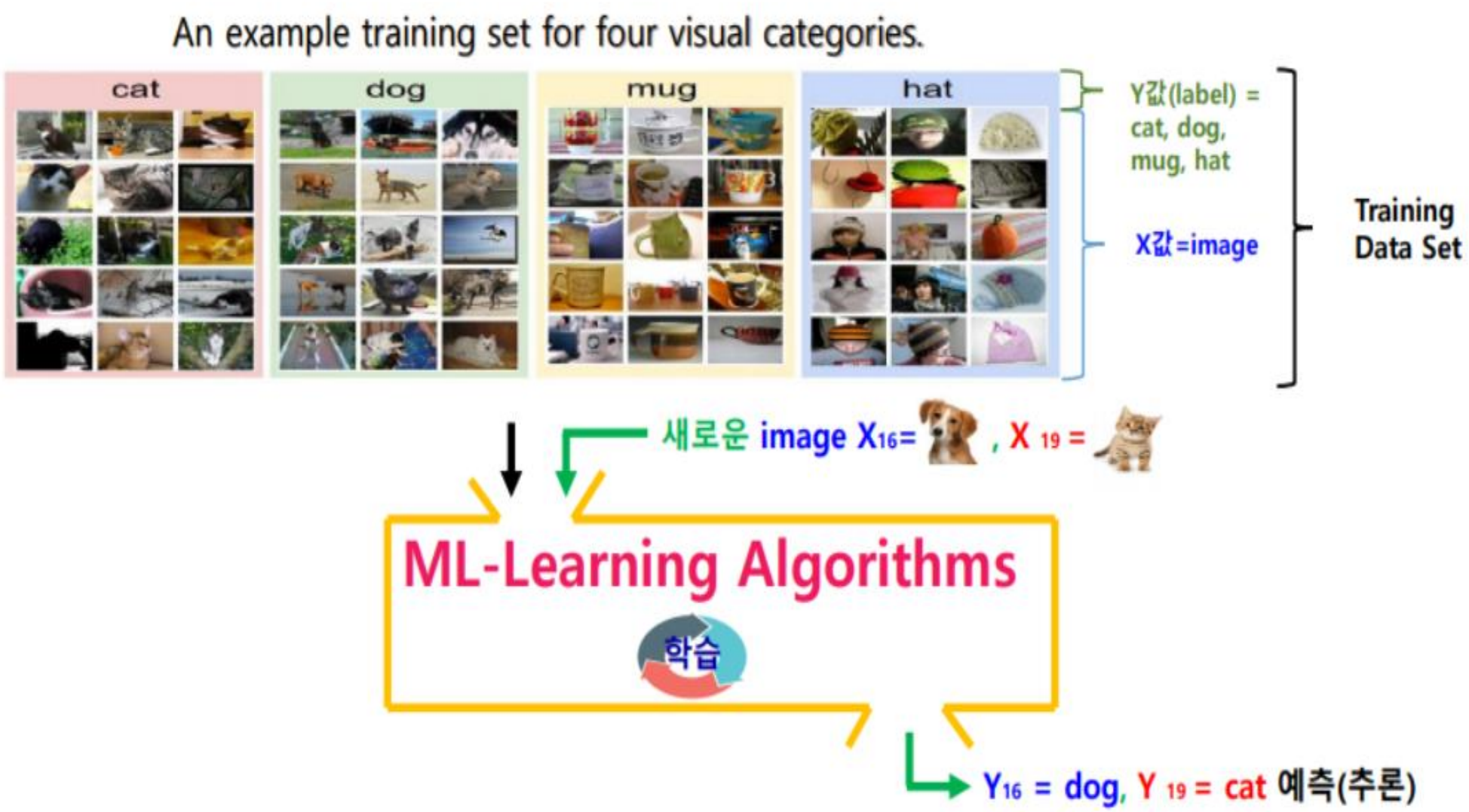
강화학습

주어진 문제의 답이 명확히 떨어지지 않지만,알고리즘이 수행한 결과에 따라서 **보상(Reward)과 손실(Penalty)이 주어져**, 이를 통해 **보상(Reward)을 최대화 하는 방향**으로 진행하는 학습 방법. **시행착오(trial and error)적인 방법**으로 적합한 행동을 탐색함
ex : **게임, 제어 전략**, 금융 시장의 매매 전략,
Robotics, 자율(Autonomous) 주행 차

-의사결정 프로세스
-**보상 제도**
-**행동(Action)시리즈 배우기**

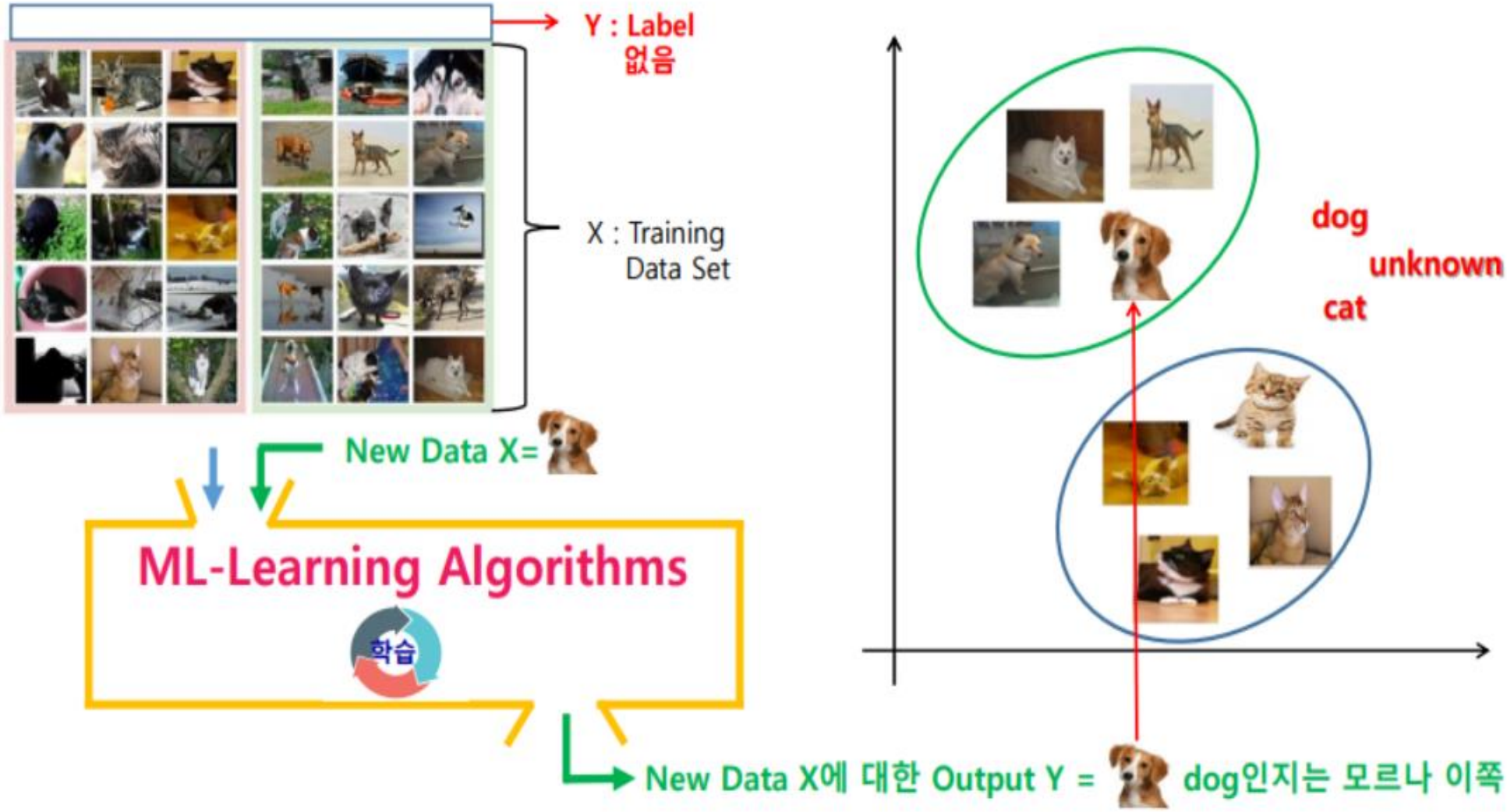
머신러닝(Machine Learning) 개념

- **Supervised Learning(예시 : image label)**



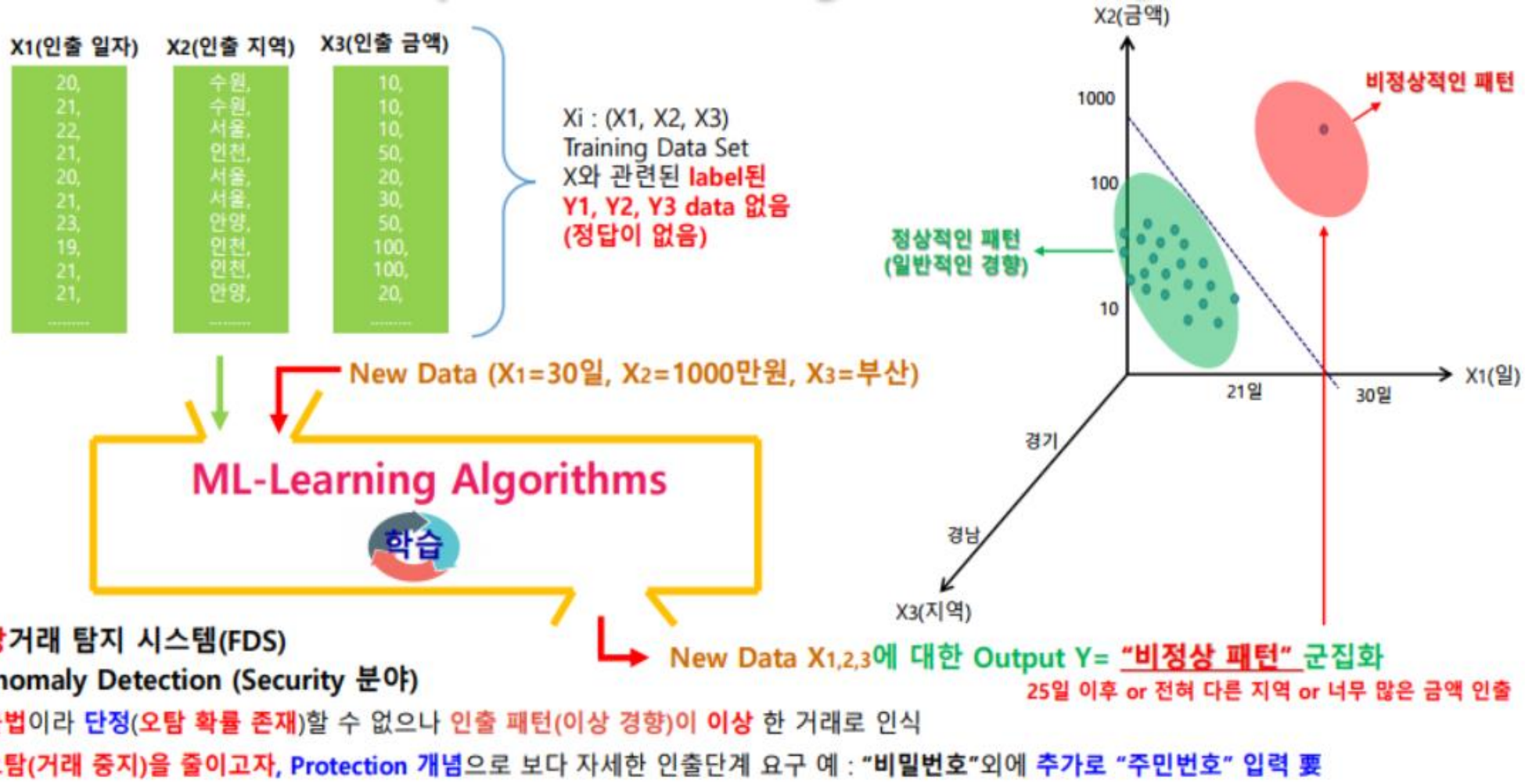
머신러닝(Machine Learning) 개념

- **Unsupervised Learning(예시 : 군집화(Clustering))**



머신러닝(Machine Learning) 개념

- **Unsupervised Learning(예시 : Clustering)**



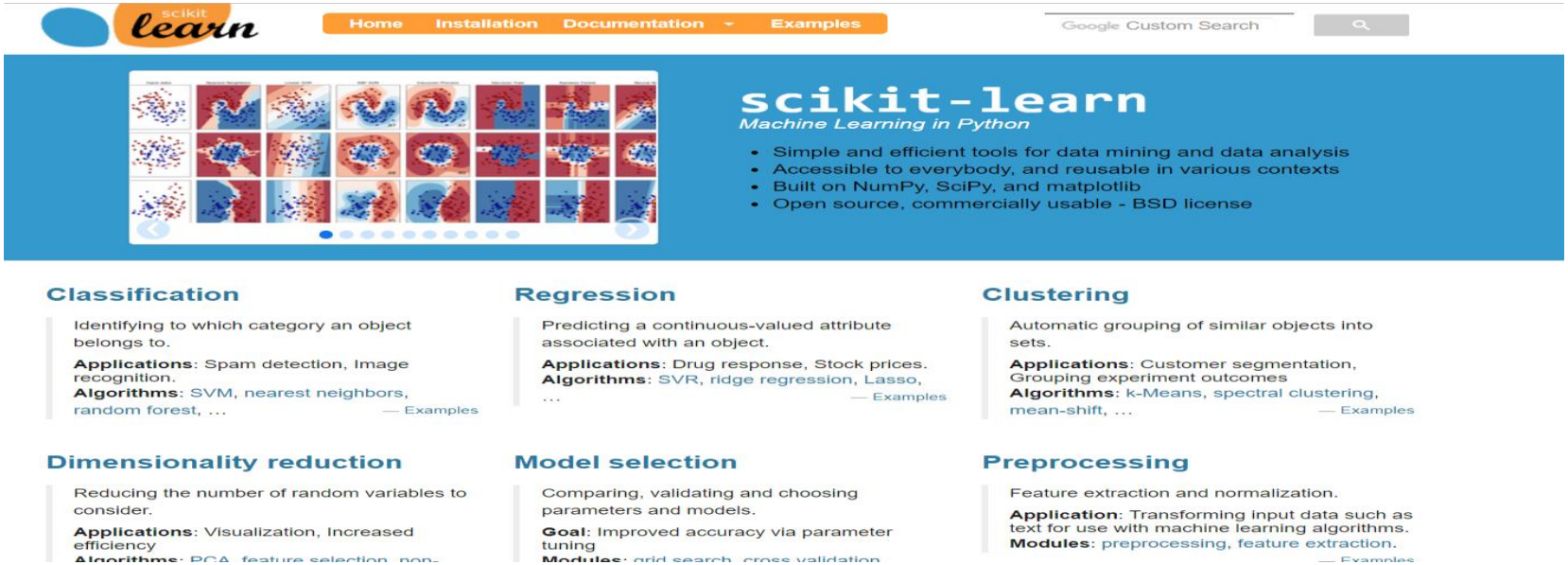
머신러닝 용어 정의

- **피처(Feature), 속성**
  - 피처는 데이터 세트의 일반 속성을 일컫는 말
  - 머신러닝은 2차원 이상의 다차원 데이터에서도 많이 사용되므로 타겟 값을 제외한 나머지 속성을 모두 피쳐로 지칭

- **레이블, 클래스, 타겟(값), 결정(값)**
  - 타겟 값 또는 결정 값은 지도 학습 시 데이터의 학습을 위해 주어지는 정답 데이터
  - 지도 학습 중 분류의 경우에는 이 결정 값을 레이블 또는 클래스로 지칭

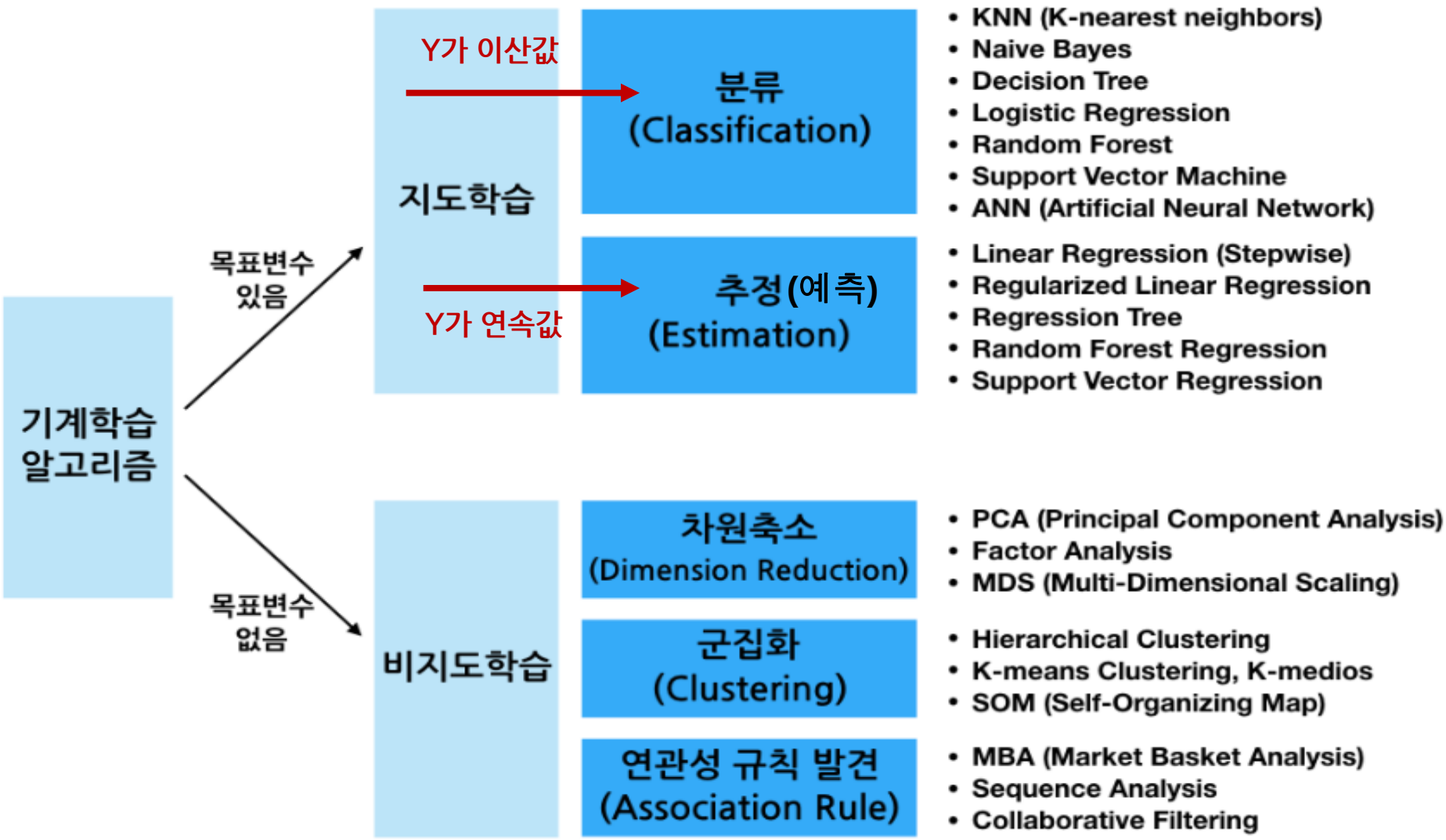
사이킷런 소개와 특징

- Scikit-learn 소개 : https://scikit-learn.org/stable/
  - 파이썬 머신러닝 라이브러리 중 가장 많이 사용되는 라이브러리



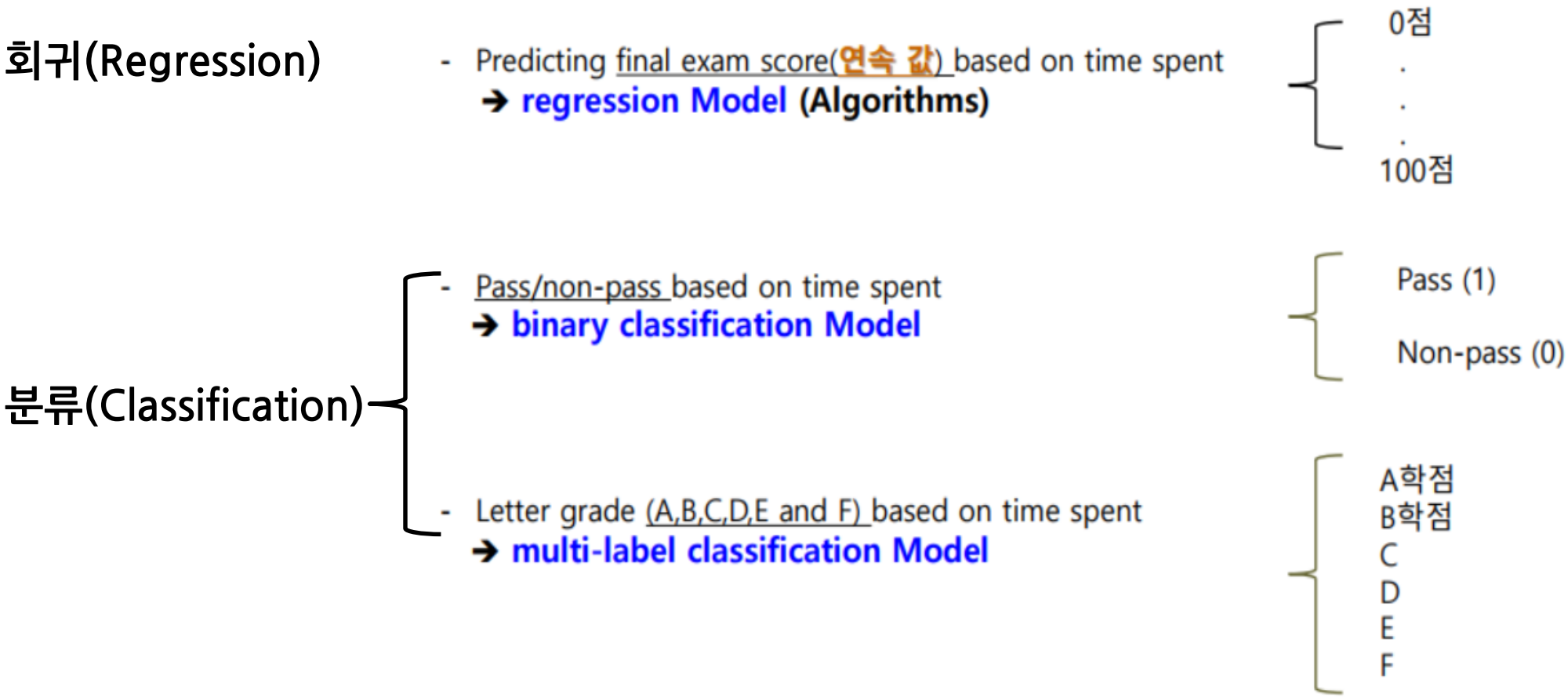
- 사이킷런의 특징
  - 파이썬 기반의 다른 머신러닝 패키지도 사이킷런 스타일의 API를 지향할 정도로 가장 파이썬스러운 API 제공
  - 머신러닝을 위한 매우 다양한 알고리즘과 개발을 위한 편리한 프레임워크와 API 제공
  - 오랜 기간 실전 환경에서 검증되었으며, 매우 많은 환경에서 성숙한 라이브러리
  - 주로 Numpy와 Scipy 기반 위에서 구축된 라이브러리

머신러닝(Machine Learning)의 종류



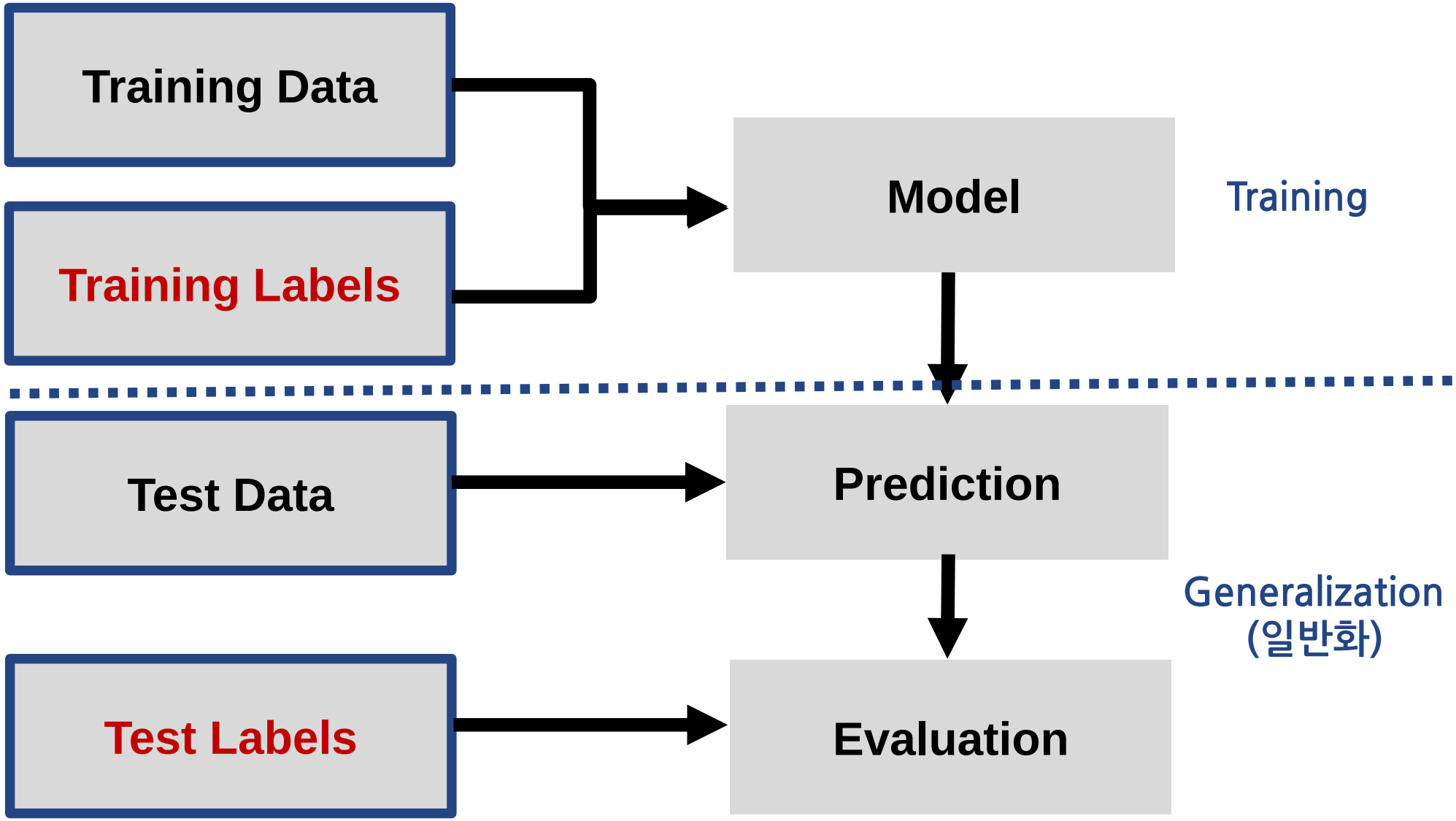
머신러닝의 종류

- **Supervised Learning(Model Type)**



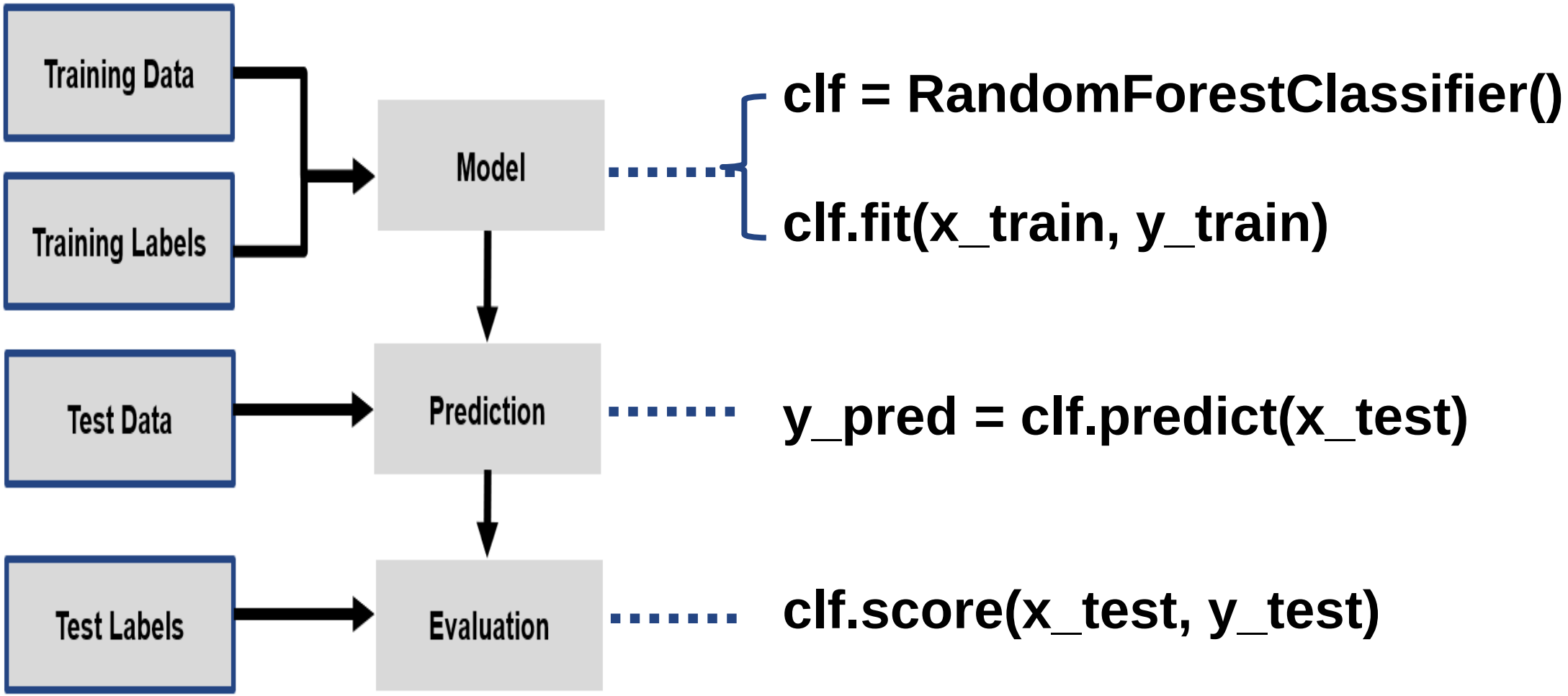
머신러닝의 종류

- 지도 학습(Supervised Machine Learning)



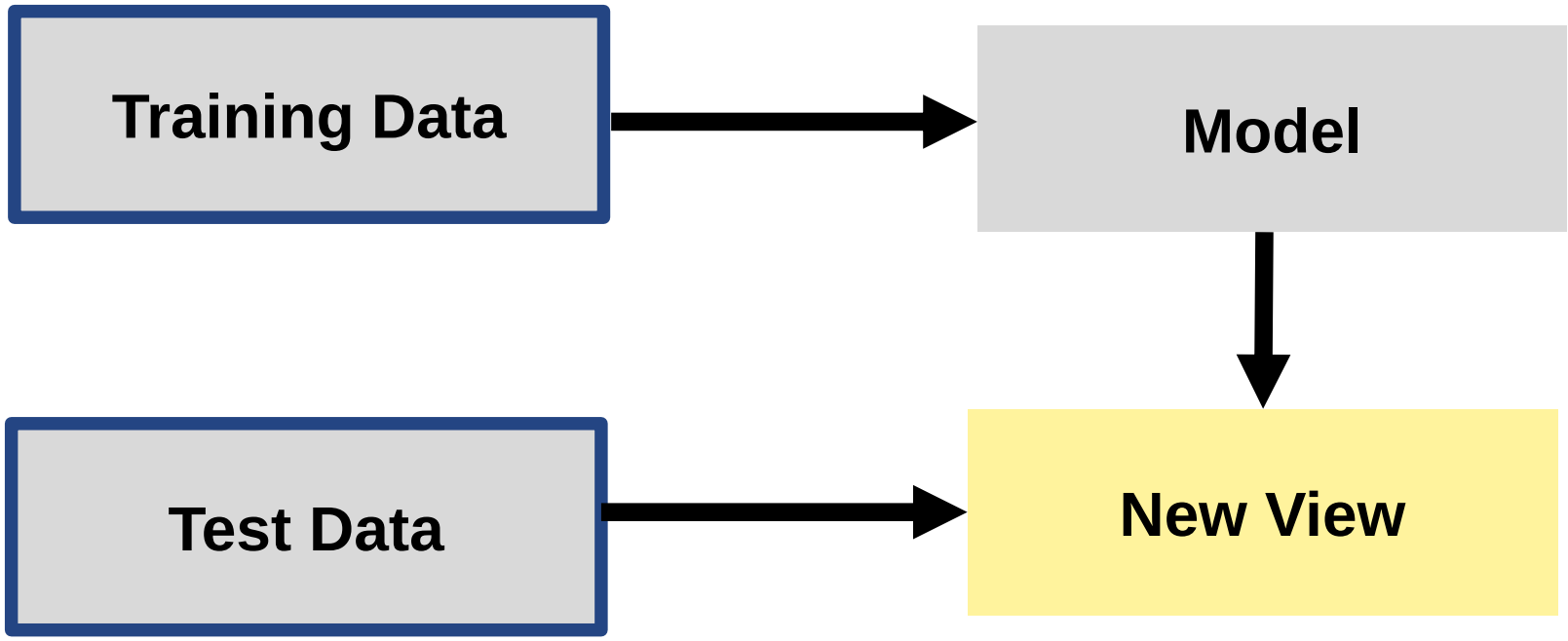
머신러닝의 종류

- 사이킷런의 지도 학습(Supervised Machine Learning)



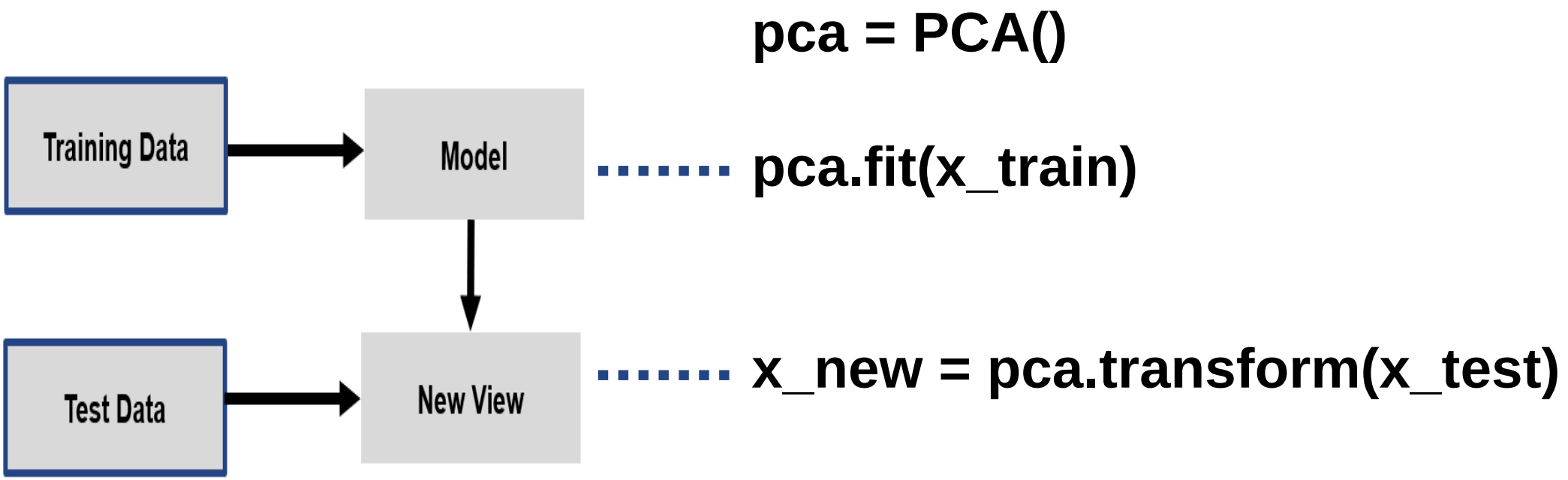
머신러닝의 종류

- 비지도 학습(Unsupervised Machine Learning)



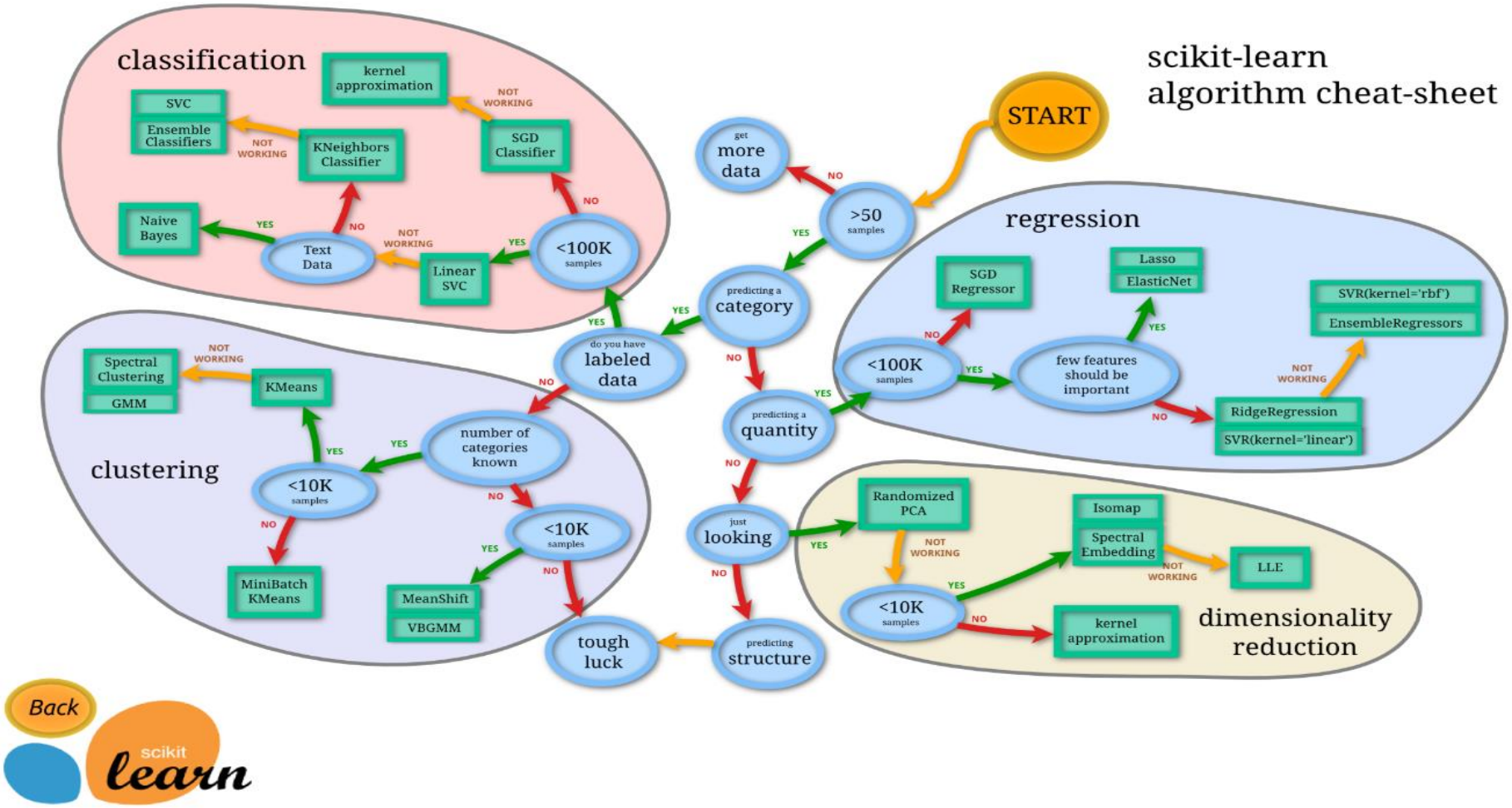
머신러닝의 종류

- 사이킷런의 비지도 학습(Unsupervised Machine Learning)



사이킷런 소개와 특징

- **Scikit-learn algorithm cheat-sheet**
  - https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html



사이킷런 소개와 특징

- **사이킷런 모듈 및 기능**
  - Sample dataset, Data preprocessing 기능 , Supervised learning, Unsupervised learning, 모델 평가 기능
  - 사이킷런은 다양한 머신러닝 알고리즘을 하나의 패키지 안에서 모두 제공해 준다는 점

- **사이킷런 패키지에서 제공하는 머신러닝 알고리즘(http://scikit-learn.org)**

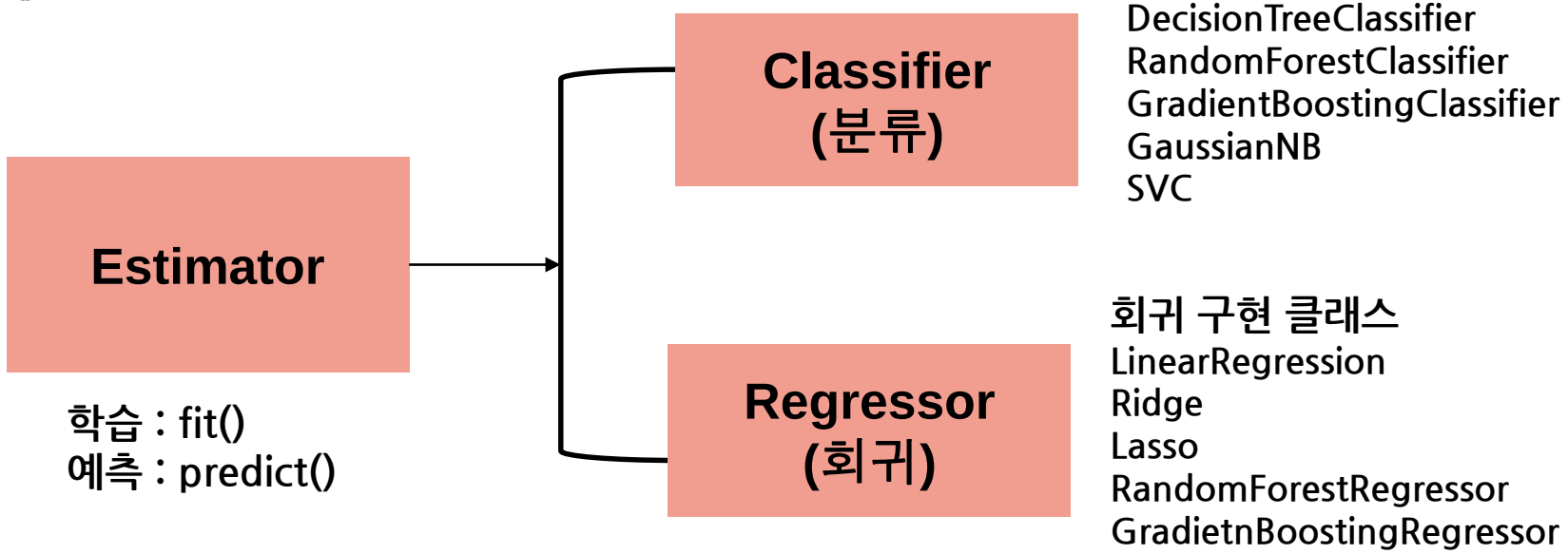
Supervised Learning	Unsupervised Learning
GeneralizedLinearModels	Gaussianmixturemodels
LinearandQuadraticDiscriminantAnalysis	Manifoldlearning
Kernelridgeregression	Clustering
SupportVectorMachines	Biclustering
StochasticGradientDescent	Decomposingsignalsincomponents(matrixfactorizationproblems)
NearestNeighbors	Covarianceestimation
GaussianProcesses	NoveltyandOutlierDetection
Crossdecomposition	DensityEstimation
NaiveBayes	Neuralnetworkmodels(unsupervised)
DecisionTrees	
Ensemblemethods	

사이킷런 소개와 특징

- 사이킷런의 기반 프레임워크 익히기 - Estimator 클래스 및 fit(), predict() 메서드

  - Estimator **클래스**
    - **지도학습**의 모든 알고리즘을 구현한 클래스의 통칭
    - 분류 알고리즘을 구현한 클래스 Classifier, 회귀 알고리즘을 구현한 클래스 Regressor 제공
    - fit()과 predict()로 내부에서 구현

  - fit() - **사이킷런의 ML 모델 학습**
  - predict() - **학습된 모델의 예측**



사이킷런 소개와 특징

- **사이킷런의 기반 프레임워크 익히기 - Estimator 클래스 및 fit(), predict() 메서드**

- **사이킷런의 비지도 학습**
  - 차원 축소, 클러스터링, 피처 추출(Feature Extraction)
  - **fit()과 transform()을 적용**
  - **비지도 학습**에서 fit() - 입력 데이터의 형태에 맞춰 데이터를 변환하기 위한 사전 구조를 맞추는 작업
  - transform() - 입력 데이터의 차원 변환, 클러스터링, 피처 추출 등의 실제 작업

사이킷런 소개와 특징

- 사이킷런 주요 모듈

분류	모듈명	설명
예제 데이타	sklearn.datasets	사이킷런에 내장되어 예제로 제공하는 데이타세트
데이터 분리, 검증 & 파라미터 튜닝	sklearn.model_selection	교차 검증을 위한 학습용/테스트용 분리, 그리드서치(Grid Search)로 최적 파라미터 추출 등의 API 제공
피처 처리	sklearn.preprocessing	데이터 전처리에 필요한 다양한 가공 기능 제공 (문자열을 숫자형 코드 값으로 인코딩, 정규화, 스케일링 등)
	sklearn.feature_selection	알고리즘에 큰 영향을 미치는 피처를 우선순위 대로 셀렉션 작업을 수행하는 다양한 기능 제공
	sklearn.feature_extraction	텍스트 데이터나 이미지 데이터의 벡터화된 피쳐를 추출하는데 사용됨 예를 들어, 텍스트 데이터에서 Count Vectorizer나 tf-idf Vectorizer 등을 생성하는 기능 제공 텍스트 데이터 피처 추출은 sklearn.feature_extraction.text 모듈에 이미지 데이터의 피처 추출은 sklearn.feature_extraction.img 모듈에 지원 API가 있음
피처 처리 & 차원 축소	sklearn.decomposition	차원 축소와 관련된 알고리즘을 지원하는 모듈 PCA, NMF, Truncated SVD 등을 통해 차원 축소 기능 수행

사이킷런 소개와 특징

- 사이킷런 주요 모듈

분류	모듈명	설명
머신러닝 알고리즘	sklearn.ensemble	앙상블 알고리즘 제공 랜덤 포레스트, 에이타 부스터, 그래디언트 부스팅 등을 제공
	sklearn.linear_model	주로 선형 회귀, 릿지(Ridge), 라쏘(Lasso) 및 로지스틱 회귀 등 회귀 관련 알고리즘 지원
	sklearn.naïve_bayes	나이브베이즈 알고리즘 제공, 가우시안 NB, 다항 분포 NB 등
	sklearn.neighbors	최근접 이웃 알고리즘 제공, K-NN등
	sklearn.svm	서포트 벡터 머신 알고리즘 제공
	sklearn.tree	의사 결정 트리 알고리즘 제공
평가	sklearn.metrics	분류, 회귀, 클러스터링, 페어와이즈(Pairwise)에 대한 다양한 성능 측정 방법 제공 Accuracy, Precision, Recall, ROC-AUC, RMSE 등 제공
유틸리티	sklearn.pipeline	피처 처리 등의 변환과 ML 알고리즘 학습, 예측 등을 함께 묶어서 실행할 수 있는 유틸리티 제공

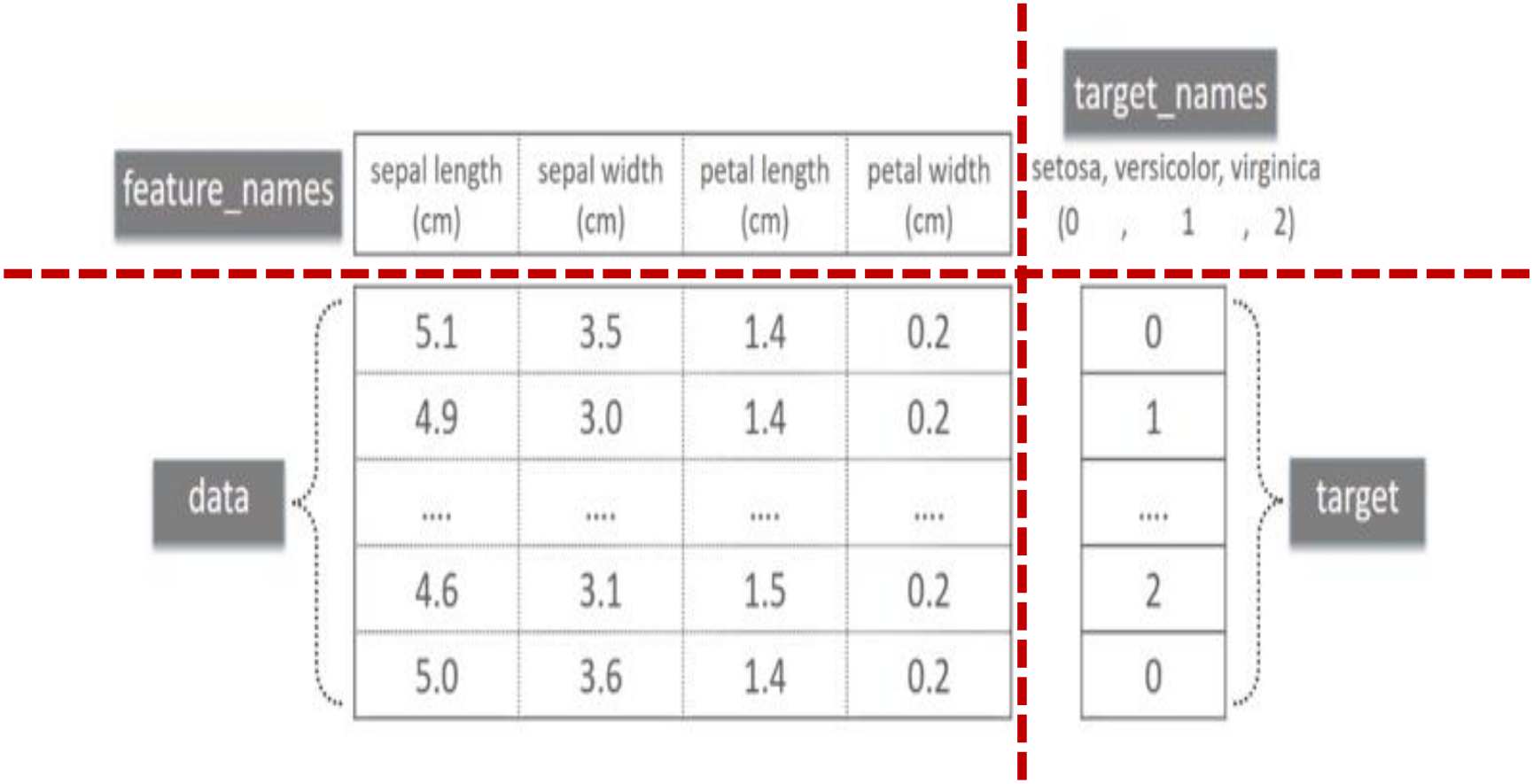
사이킷런 소개와 특징

- 사이킷런 내장 예제 데이터 셋 - 분류 및 회귀용

API명	설명
datasets.load_boston()	회귀 용도, 미국 보스턴의 집 피처들과 가격에 대한 데이터 세트
datasets.load_breast_cancer()	분류 용도, 위스콘신 유방암 피처들과 악성/음성 레이블 데이터 세트
datasets.load_diabetes()	회귀 용도, 당뇨 데이터 세트
datasets.load_digits()	분류 용도, 0에서 9까지 숫자의 이미지 픽셀 데이터 세트
datasets.load_iris()	분류 용도, 붓꽃에 대한 피처를 가진 데이터 세트

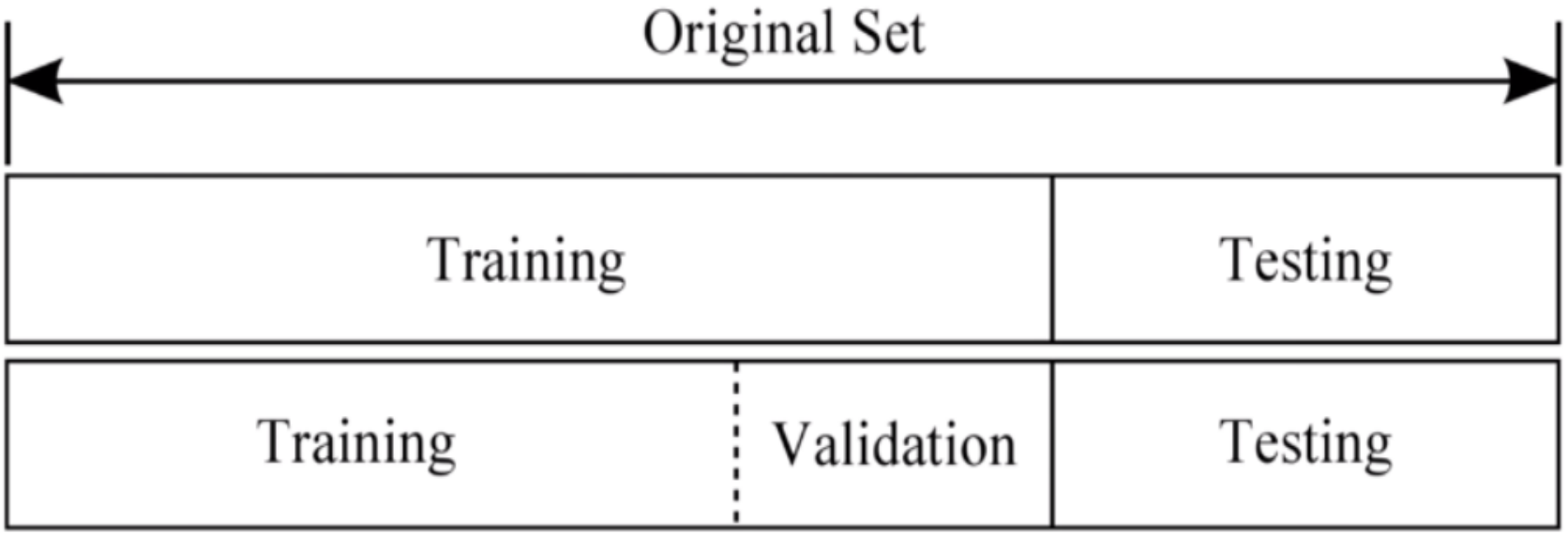
사이킷런 소개와 특징

- 내장 예제 데이터 셋 구성
  - data, target, feature_names, target_names



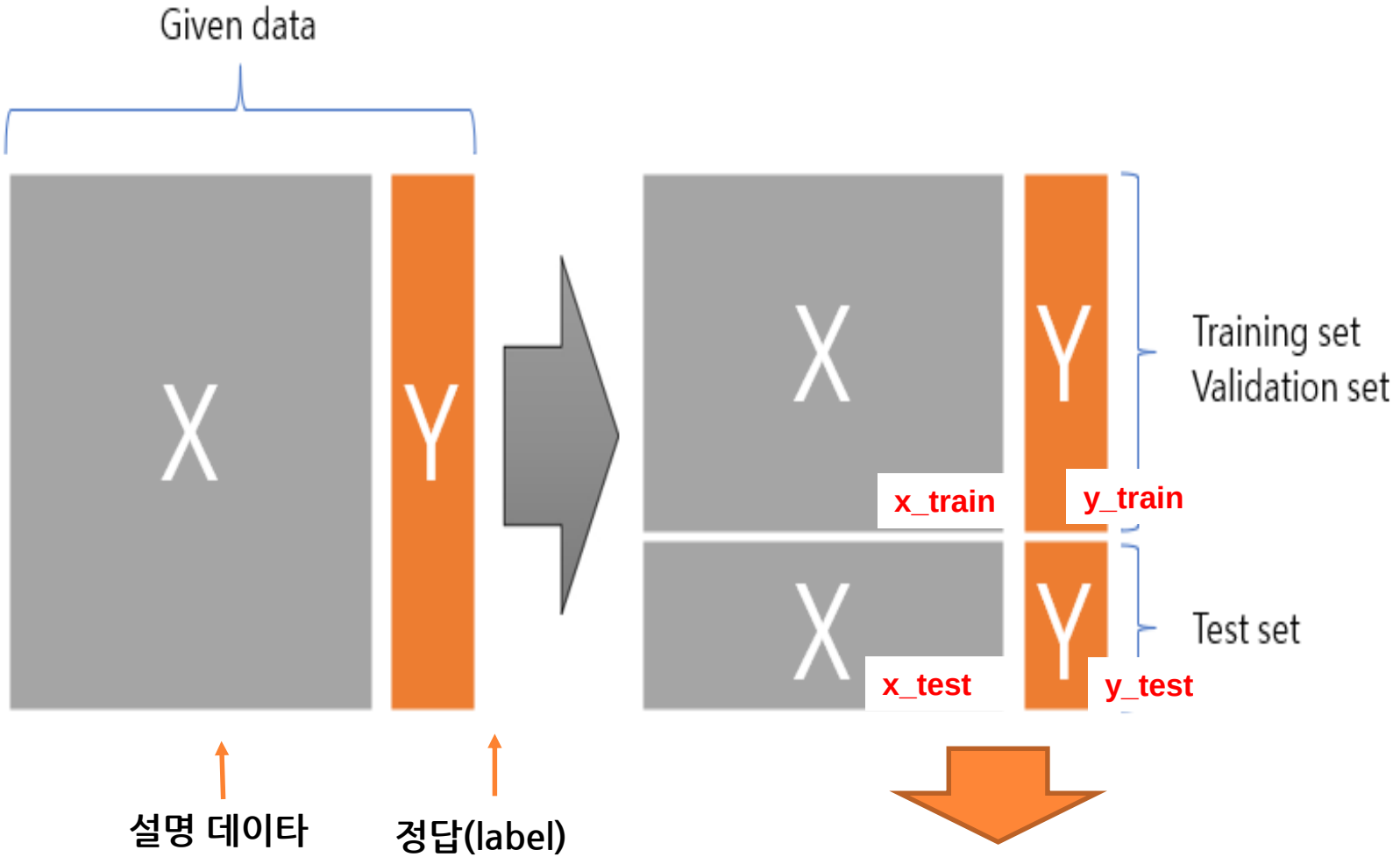
홀드아웃(Hold Out)

- 홀드아웃(Hold Out)
  - 데이터를 훈련 데이터와 테스트 데이터로 나눔
  - 모형의 최종 성능을 객관적으로 측정하기 위한 방법으로 트레이닝에 사용되지 않은 새로운 데이터(테스트 데이터)를 사용해서 예측한 결과를 기반으로 성능을 계산
  - 일정한 비율로 Train/Test의 비율로 나누어 사용(7:3, 8:2, 6:4)



홀드아웃(Hold Out)

- 홀드아웃(Hold Out)



사이킷런의 학습/테스트 데이터 분류 : x_train, x_test, y_train, y_test

사이킷런 모듈

- Model Selection 모듈 - 학습 데이터와 테스트 데이터 분리하는 모듈
- 학습 데이터 세트와 테스트 데이터 세트

학습 데이터 세트	테스트 데이터 세트
• 머신러닝 알고리즘의 학습을 위해 사용 • 데이터의 속성들과 결정값(레이블) 모두를 가지고 있음 • 학습 데이터를 기반으로 머신러닝 알고리즘이 데이터 속성과 결정값의 패턴을 인지하고 학습	• 테스트 데이터 세트에서 학습된 머신러닝 알고리즘을 테스트 • 테스트 데이터는 속성 데이터만 머신러닝 알고리즘에 제공하며, 머신러닝 알고리즘은 제공된 데이터를 기반으로 결정값을 예측 • 테스트 데이터는 학습 데이터와 별도의 데이터 세트로 제공되어야 함

사이킷런 모듈

- 홀드아웃 : 학습 데이터와 테스트 데이터 분리 - train_test_split()
  - sklearn.model_selection의 train_test_split() 함수

```
X_train, X_test, y_train, y_test = train_test_split( iris_data.data,
                                                     iris_data.target,
                                                     test_size=0.3,
                                                     random_state=0)
```

- test_size : 전체 데이터에서 테스트에서 테스트 데이터 세트 크기를 얼마로 샘플링 할 것인가를 결정, 디폴트는 0.25, 즉 25% 입니다.

- shuffle : 데이터를 분리하기 전에 데이터를 미리 섞을지를 결정, 디폴트는 True, 데이터를 분산시켜서 좀 더 효율적인 학습 및 테스트 데이터 세트를 만드는 데 사용

- random_state : 호출할 때마다 동일한 학습/데스트용 데이터 세트를 생성하기 위해 주어지는 난수 값, train_test_split()는 호출 시 무작위로 데이터를 분리하므로 random_state 를 지정하지 않으면 수행할 때마다 다른 학습/테스트용 데이터를 생산한다. 재현율을 보장받기 위해서는 옵션을 설정해야 한다.

사이킷런 모듈

- 홀드 아웃 : 학습 데이터와 테스트 데이터 분리 - train_test_split()
  - 1) 데이터 로드

```
1  import pandas as pd
2  import numpy as np
3
4  # 데이터 로드
5  path = "../data/DataPreprocess.csv"
6  df1 = pd.read_csv(path)
7  df1.head()
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes

```
1  df1.shape  # 10개의 관측치, 4개 변수
```

```
(10, 4)
```

사이킷런 모듈

- 홀드 아웃 : 학습 데이터와 테스트 데이터 분리 - train_test_split()
  - 2) 데이터와 레이블 나누기

```
1  # 데이터와 레이블 나누기 - 종속변수(반응변수)와 독립변수(설명변수) 나누기
2  x = df1.values[:, :-1]   # 데이터
3  y = df1.values[:, -1]    # 레이블(정답)
4  x, y
```

```
(array([['France', 44.0, 72000.0],
        ['Spain', 27.0, 48000.0],
        ['Germany', 30.0, 54000.0],
        ['Spain', 38.0, 61000.0],
        ['Germany', 40.0, nan],
        ['France', 35.0, 58000.0],
        ['Spain', nan, 52000.0],
        ['France', 48.0, 79000.0],
        ['Germany', 50.0, 83000.0],
        ['France', 37.0, 67000.0]], dtype=object),
 array(['No', 'Yes', 'No', 'No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes'],
       dtype=object))
```

사이킷런 모듈

- 홀드 아웃 : 학습 데이터와 테스트 데이터 분리 - train_test_split()
  - 3) 학습 데이터와 테스트 데이터 분리

```
1  from sklearn.model_selection import train_test_split
2
3  x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
4  x_train.shape, x_test.shape
```

```
((8, 3), (2, 3))
```

```
1  x_train, x_test
```

```
(array([['Germany', 40.0, nan],
        ['France', 37.0, 67000.0],
        ['Spain', 27.0, 48000.0],
        ['Spain', nan, 52000.0],
        ['France', 48.0, 79000.0],
        ['Spain', 38.0, 61000.0],
        ['France', 44.0, 72000.0],
        ['France', 35.0, 58000.0]], dtype=object),
 array([['Germany', 30.0, 54000.0],
        ['Germany', 50.0, 83000.0]], dtype=object))
```

사이킷런 모듈

- 실습 문제) iris 데이타셋으로 학습 데이터와 테스트 데이터 분리

1) 사이킷런의 train_test_split 이용하여 구현

```
1  from sklearn.model_selection import train_test_split
2  from sklearn.datasets import load_iris
3
4  # iris 데이타셋 로드
5  iris_data = load_iris()
6
7  # 학습/테스트 데이타 세트 분리
8  x_train, x_test,y_train, y_test= train_test_split(iris_data.data, iris_data.target,
9                                                    test_size=0.3, random_state=0)
```

```
1  x_train.shape, x_test.shape
```

```
((105, 4), (45, 4))
```

```
1  y_train.shape, y_test.shape
```

```
((105,), (45,))
```

사이킷런 모듈

- 실습 문제) iris 데이타셋으로 학습 데이터와 테스트 데이터 분리

2) 판다스 DataFrame의 슬라이싱을 이용하여 구현

```
1  import pandas as pd
2
3  iris_df = pd.DataFrame(iris_data.data, columns=iris_data.feature_names)
4  iris_df['target']=iris_data.target
5  iris_df.head(2)
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0

```
1  train = iris_df.iloc[:, :-1]
2  target = iris_df.iloc[:, -1]
3  x_train, x_test, y_train, y_test = train_test_split(train, target,
4                                                      test_size=0.3, random_state=0)
```

```
1  print(type(x_train), type(x_test))
2  print(type(y_train), type(y_test))
```

```
<class 'pandas.core.frame.DataFrame'> <class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.series.Series'> <class 'pandas.core.series.Series'>
```

사이킷런 모듈

- 실습 문제) iris 데이타셋으로 학습 데이터와 테스트 데이터 분리

3) 결정트리를 이용한 분류 및 정확도

```
1 from sklearn.tree import DecisionTreeClassifier
2 from sklearn.metrics import accuracy_score
3
4 dt_clf = DecisionTreeClassifier( ) # 모델 선택
5 dt_clf.fit(x_train, y_train)       # 학습
6 pred = dt_clf.predict(x_test)      # 예측
7
8 print('예측 정확도: {0:.4f}'.format(accuracy_score(y_test,pred))) # 모델평가 - 정확도
```

```
예측 정확도: 0.9778
```

```
1 pred
```

```
array([2, 1, 0, 2, 0, 2, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0, 0, 2, 1,
       0, 0, 2, 0, 0, 1, 1, 0, 2, 1, 0, 2, 2, 1, 0, 2, 1, 1, 2, 0, 2, 0,
       0])
```

```
1 pred == y_test
```

```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
        True, False,  True,  True,  True,  True,  True,  True,  True])
```

사이킷런 모듈

■ 교차 검증(K-Fold Cross Validation)

- K개의 Fold를 만들어서 진행, **총 데이터 개수가 적은 데이터 셋에 대하여 정확도를 향상시킬 수 있음**
- 기존의 Train/Validation/Test 세개의 집단으로 분류하는 것보다 Train/Test 셋만으로 분류된 학습 데이터 셋이 많기 때문
- 데이터 수가 적은데 검증과 테스트에 데이터를 뺏기면 underfitting 되는 모델이 학습됨

사이킷런 모듈

- **교차 검증의 종류**
  - **1) K-Fold 교차 검증**
    - 가장 보편적으로 사용되는 교차 검증 기법
    - 먼저 K개의 데이터 폴드 세트를 만들어서 K번만큼 각 폴트 세트에 학습과 검증 평가를 반복적으로 수행하는 방법
  - **2) Stratified K-Fold**
    - 불균형(imbalanced) 분포도를 가진 레이블(결정 클래스) 데이터 집합을 위한 K-폴드 방식
    - 불균형한 분포를 가진 레이블 데이터 집합은 특정 레이블 값이 특이하게 많거나 매우 적어서 값의 분포가 한쪽으로 치우치는 것
    - K-Fold가 레이블 데이터 집합이 원본 데이터 집합의 레이블 분포를 학습 및 테스트 세트에 제대로 분배하지 못하는 경우의 문제를 해결해 줌
    - Stratified K-Fold는 원본 데이터의 레이블 분포를 고려한 뒤 이 분포와 동일하게 학습과 검증 데이터 세트를 분배
    - (예) 대출 사기 데이터 - 데이터 1억건, 사기 대출 1,000건 전체의 0.0001%사기 대출

사이킷런 모듈

- 교차 검증의 종류
  - 3) cross_val_score() 함수 이용하여 교차검증을 보다 간편하게
    - cross_val_score() 함수 : 폴드 세트 추출, 학습/예측, 평가를 한번에 수행

- cross_val_score(estimator, X, y=None, groups=None, scoring=None, cv=None, n_jobs=None, verbose=0, fit_params=None, pre_dispatch='2*n_jobs', error_score=nan,)

- 중요 파라메터 - estimator, X, y, cv
- 1) estimator
  - 사이킷런의 분류 알고리즘 클래스인 Classifier또는 회귀 알고리즘 클래스인 Regressor를 의미
  - classifier(분류)가 입력되면 Stratified K 폴드 방식으로 레이블 분포에 따라 학습/테스트 세트 분할
  - regressor(회귀)로 입력되면 Stratified K 폴드 방식으로 분할할 수 없으므로 K 폴드 방식으로 분할
- 2) X - 피처 데이터 세트
- 3) y - 레이블 데이터 세트
- 4) cv - 교차 검증 폴드 수
- cross_val_score() 수행 후 반환 값 - scoring 파라미터로 지정된 성능 지표 측정값을 배열형태로 반환

사이킷런 모듈

■ 사이킷런의 교차검증(K-Fold Cross Validation)의 방법

- K-Fold 클래스를 이용한 교차 검증 방법
  - (1) 폴트 세트 설정
  - (2) for 루프에서 반복적으로 학습/검증 데이터 추출 및 학습과 예측 수행
  - (3) 폴트 세트별로 예측 성능을 평균하여 최종 성능 평가

```
1  from sklearn.model_selection import KFold
2
3  cv = KFold(n_splits=5, shuffle=True, random_state=0)
4  for train_index, test_index in cv.split(x) :
5      print('train_index : ', train_index)
6      print("." * 80)
7      print("test_index : ", test_index)
8      print("=" * 80)
```

```
train_index :  [0 1 3 4 5 6 7 9]
................................................................................
test_index :  [2 8]
================================================================================
train_index :  [0 1 2 3 5 6 7 8]
................................................................................
test_index :  [4 9]
================================================================================
train_index :  [0 2 3 4 5 7 8 9]
................................................................................
test_index :  [1 6]
================================================================================
train_index :  [0 1 2 4 5 6 8 9]
................................................................................
test_index :  [3 7]
================================================================================
train_index :  [1 2 3 4 6 7 8 9]
................................................................................
test_index :  [0 5]
================================================================================
```

사이킷런 모듈

- 실습) iris 데이타셋으로 교차검증(K-Fold Cross Validation)

```
1  from sklearn.tree import DecisionTreeClassifier
2  from sklearn.metrics import accuracy_score
3  from sklearn.model_selection import KFold
4  import numpy as np
5
6  iris = load_iris()
7  features = iris.data
8  label = iris.target
9
10 dt_clf = DecisionTreeClassifier(random_state=156)
11
12 # 5개의 폴드 세트로 분리하는 KFold 객체와 폴드 세트별 정확도를 담을 리스트 객체 생성.
13 kfold = KFold(n_splits=5)
14 cv_accuracy = []
15 print('붓꽃 데이터 세트 크기:',features.shape[0])
16
```

```
붓꽃 데이터 세트 크기: 150
```

사이킷런 모듈

- 실습) iris 데이타셋으로 교차검증(K-Fold Cross Validation)

```
n_iter = 0

# KFold객체의 split( ) 호출하면 폴드 별 학습용, 검증용 테스트의 로우 인덱스를 array로 반환
for train_index, test_index  in kfold.split(features):
    # kfold.split( )으로 반환된 인덱스를 이용하여 학습용, 검증용 테스트 데이터 추출
    x_train, x_test = features[train_index], features[test_index]
    y_train, y_test = label[train_index], label[test_index]

    #학습 및 예측
    dt_clf.fit(x_train , y_train)
    pred = dt_clf.predict(x_test)
    n_iter += 1

    # 반복 시 마다 정확도 측정
    accuracy = np.round(accuracy_score(y_test,pred), 4)
    train_size = x_train.shape[0]
    test_size = x_test.shape[0]
    print('₩n#{0} 교차 검증 정확도 :{1}, 학습 데이터 크기: {2}, 검증 데이터 크기: {3}'
          .format(n_iter, accuracy, train_size, test_size))
    print('#{0} 검증 세트 인덱스:{1}'.format(n_iter,test_index))

    cv_accuracy.append(accuracy)

# 개별 iteration별 정확도를 합하여 평균 정확도 계산
print('₩n## 평균 검증 정확도:', np.mean(cv_accuracy))
```

사이킷런 모듈

- 실습) iris 데이타셋으로 교차검증(K-Fold Cross Validation)

```
#1 교차 검증 정확도 :1.0, 학습 데이터 크기: 120, 검증 데이터 크기: 30
#1 검증 세트 인덱스:[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29]

#2 교차 검증 정확도 :0.9667, 학습 데이터 크기: 120, 검증 데이터 크기: 30
#2 검증 세트 인덱스:[30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
 54 55 56 57 58 59]

#3 교차 검증 정확도 :0.8667, 학습 데이터 크기: 120, 검증 데이터 크기: 30
#3 검증 세트 인덱스:[60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83
 84 85 86 87 88 89]

#4 교차 검증 정확도 :0.9333, 학습 데이터 크기: 120, 검증 데이터 크기: 30
#4 검증 세트 인덱스:[ 90  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107
 108 109 110 111 112 113 114 115 116 117 118 119]

#5 교차 검증 정확도 :0.7333, 학습 데이터 크기: 120, 검증 데이터 크기: 30
#5 검증 세트 인덱스:[120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137
 138 139 140 141 142 143 144 145 146 147 148 149]

## 평균 검증 정확도: 0.9
```

데이터 분석의 실수

- 과소적합(Under Fitting), 과대적합(Overfitting)

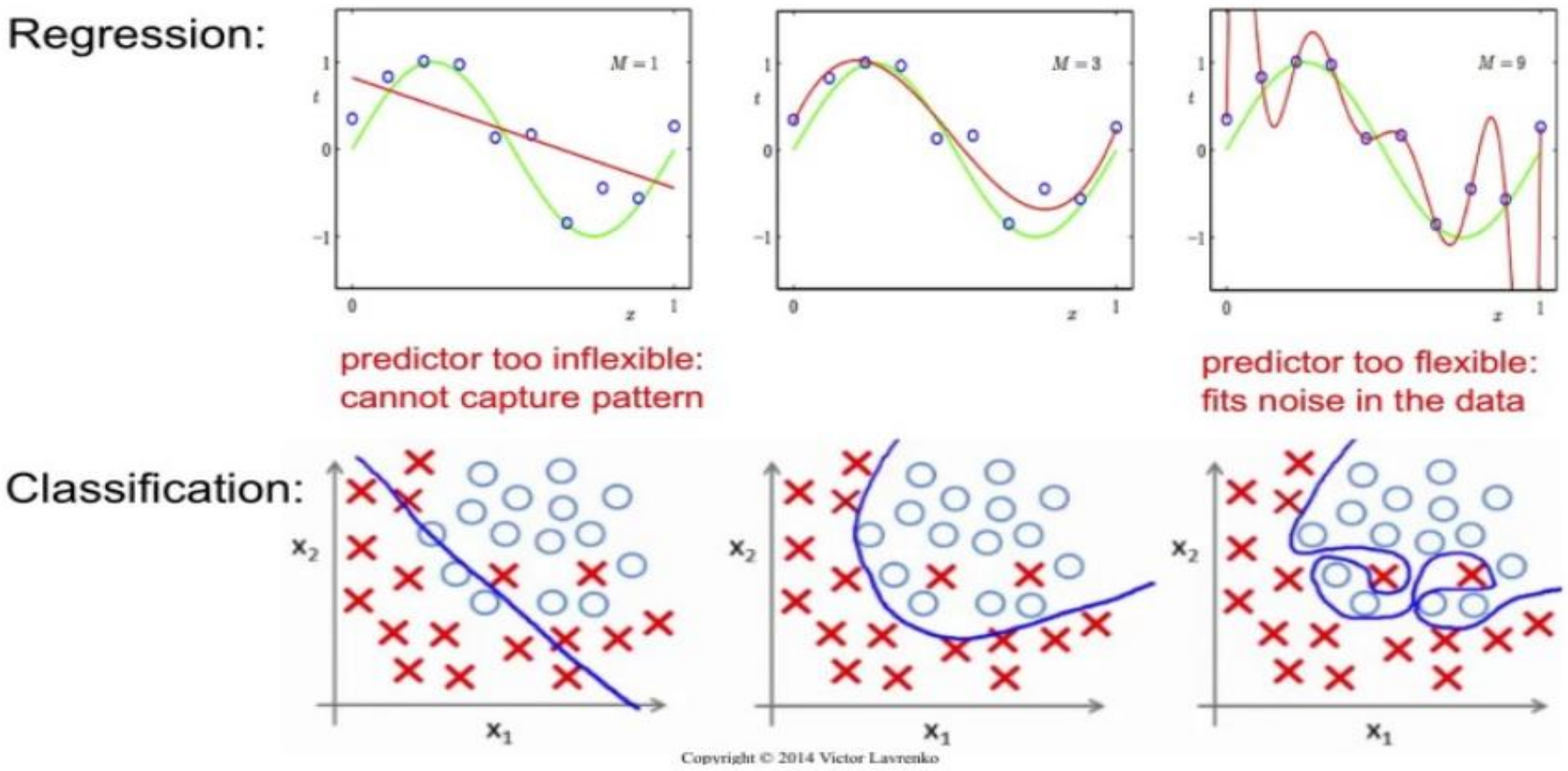


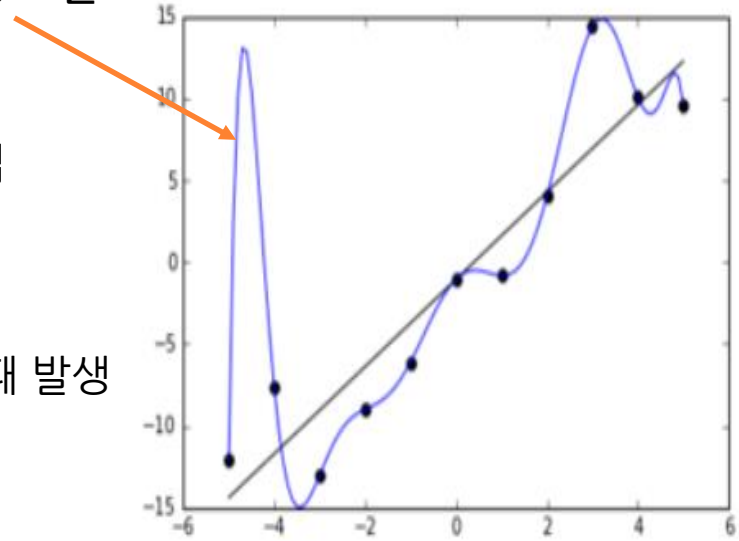
사진 출처: https://www.youtube.com/watch?v=dBLZg-RqoLg

데이터 분석의 실수

- **과대적합(Overfitting)**
  - 모델이 훈련 데이터에 너무 잘 맞지만 일반성이 떨어진다는 의미
  - 훈련 데이터에 잘 맞으면 좋은 것이라고 생각할 수도 있지만, "너무 잘 맞는 것"이 문제가 되는 것
  - 훈련 데이터에 너무 맞추어져 있기 때문에 훈련 데이터 이외의 다양한 변수에는 대응하기 힘들어짐
    또한 모델의 복잡도가 필요 이상으로 높아짐

  - **과대적합(오버피팅) 해결 방법**
    1) 훈련 데이터를 더 많이 모음
    2) 정규화(Regularization)
       - 규제(제약 조건), 드롭-아웃 등 다양한 방법을 이용해서
         적당한 복잡도를 가지는 모델을 자동적으로 찾아주는 기법
    3) 훈련 데이터 잡음을 줄임(오류 수정과 이상치 제거)

오버피팅 모델



- **과소적합(Underfitting)**
  - 모델이 너무 단순해서 데이터의 내재된 구조를 학습하지 못할 때 발생

  - **과소적합(언더피팅) 해결 방법**
    1) 파라미터가 더 많은 복잡한 모델을 선택
    2) 모델의 제약을 줄이기(규제 하이퍼 파라미터 값 줄이기)
    3) 조기종료 시점(overfitting이 되기 전의 시점)까지 충분히 학습