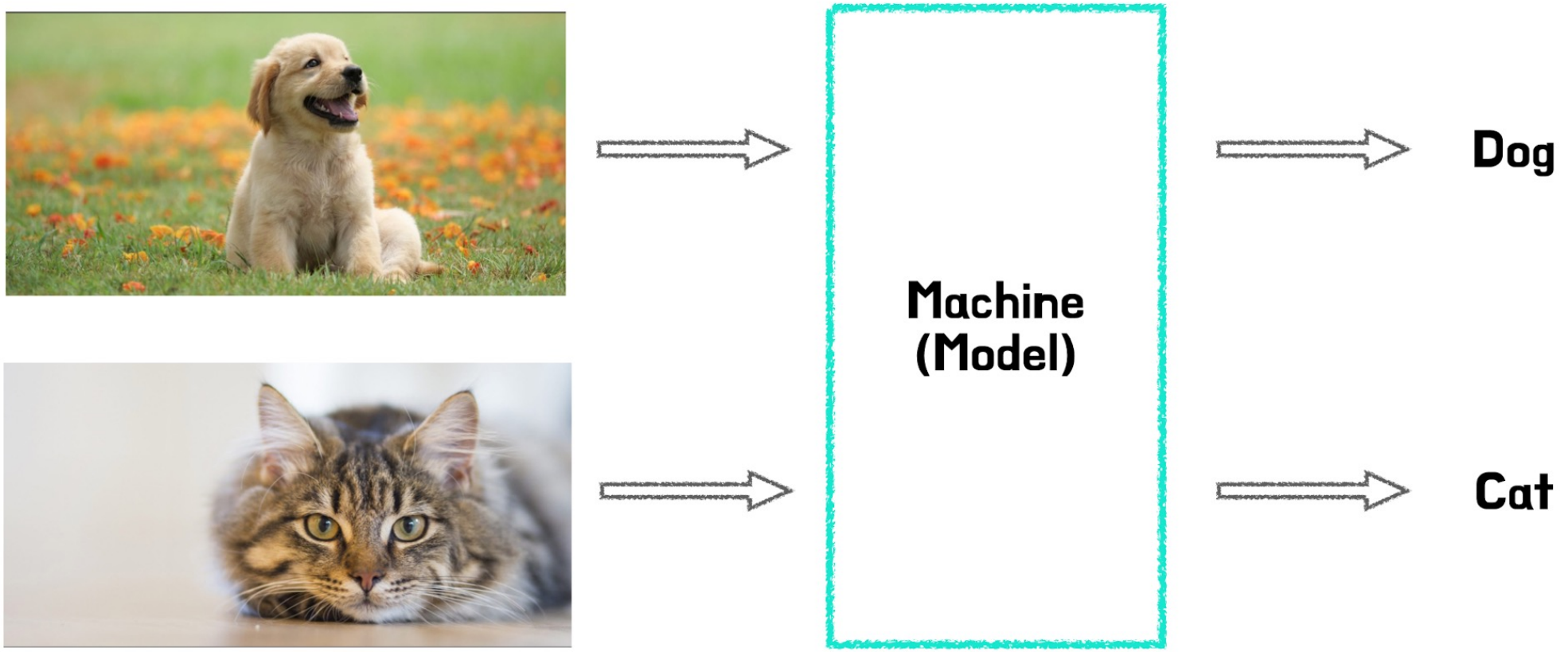
python™

머신러닝(Machine Learning)이란?



python™

Machine Learning Algorithm

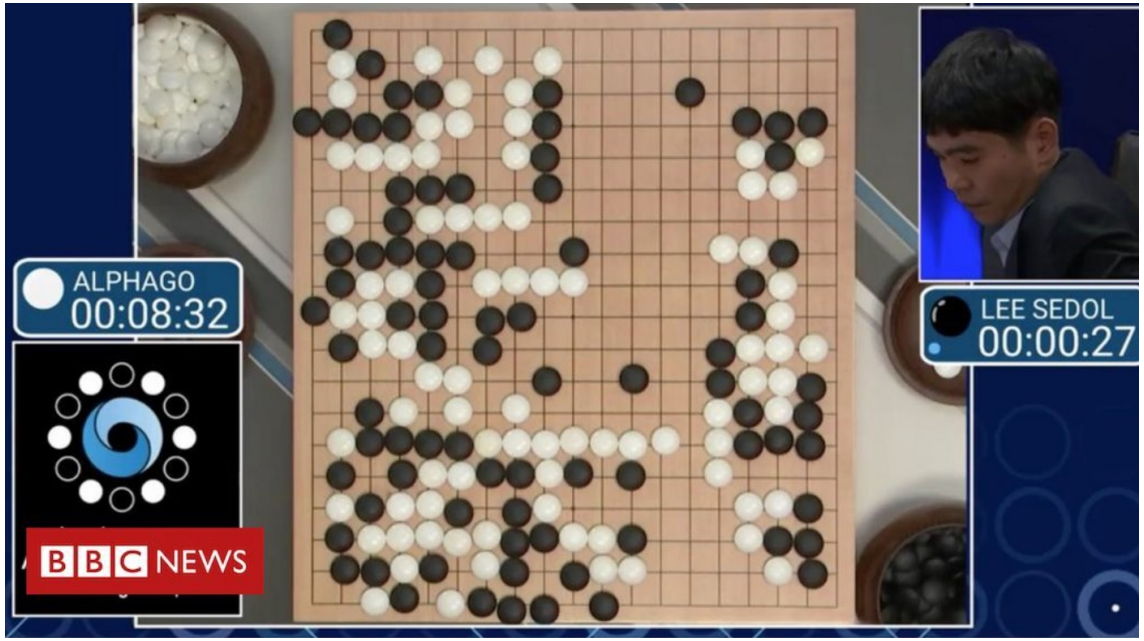
- 지도학습 (supervised learning)
  데이터에 대한 label(정답)이 주어진 상태에서 컴퓨터를 학습 시키는 것

- 비지도학습 (unsupervised learning)
  데이터에 대한 label(정답)이 주어지지 않은 상태에서 컴퓨터를 학습 시키는 것

- 강화학습(reinforcement learning)
  현재 상태에서 어떤 행동을 취하는 것이 최적인지를 컴퓨터에 학습시키는 것

강화학습 (Reinforcement Learning)

- 현재 상태에서 어떤 행동을 취하는 것이 가장 좋은지 컴퓨터가 학습
  행동을 취할 때 마다 외부에서 보상이 주어지는데 이는 긍정적일 수도 부정적일 수도 있음
  또한 보상은 즉각적으로 주어질 수도 있고, 시간이 흐른 후에 주어질 수도 있음(지연된 보상)
  가장 긍정적인 결과를 출력하기 위한 방법을 계속해서 학습함



강화학습의 가장 대표적인 예시인 알파고

python™

Machine Learning Models

- Regression (회귀분석)
  데이터를 바탕으로 숫자 값을 예측한다.
  예) 공부량 → 시험점수

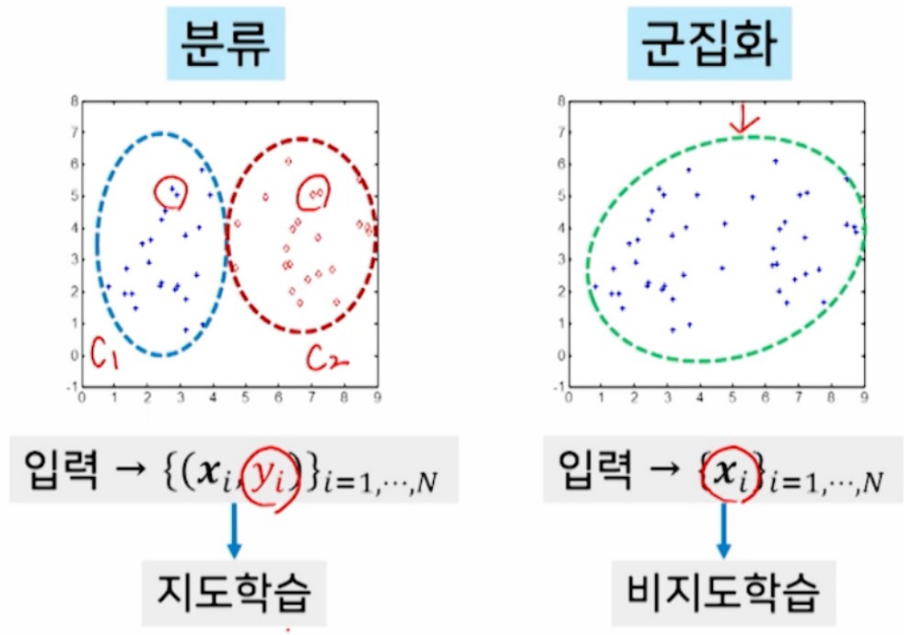
- Classification (분류)
  데이터를 바탕으로 A인지 B인지를 예측한다.
  예) 공부량 → 학점 (A / B / C ...)

- Clustering (군집화)

- Others

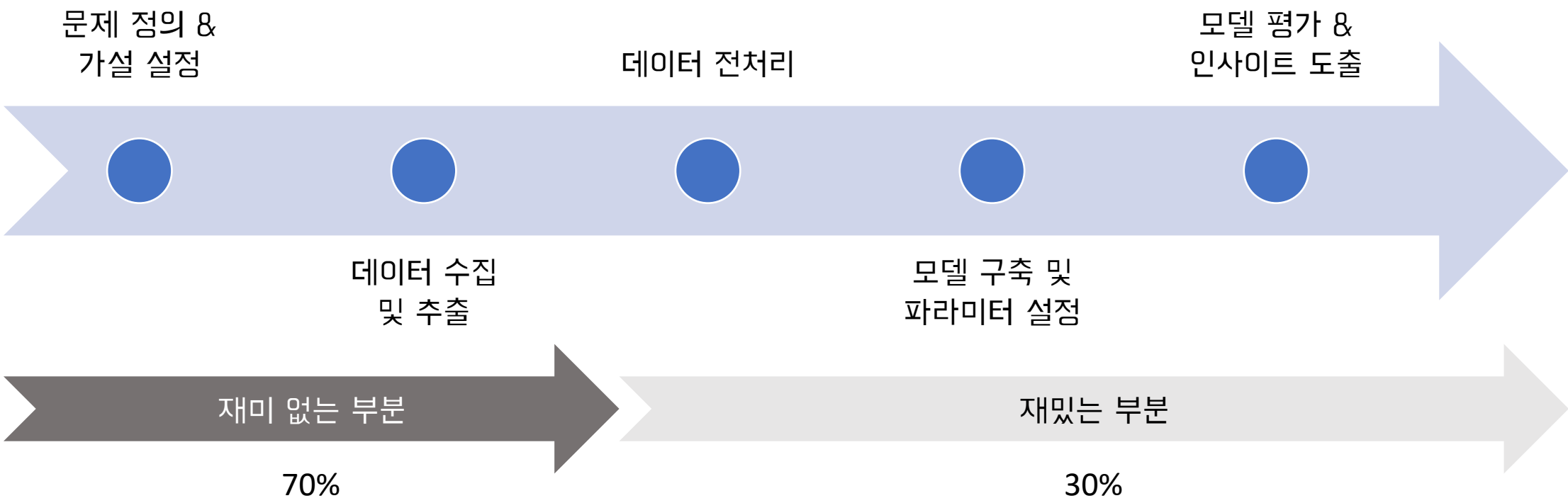
Classification vs Clustering

- Classification (분류) : supervised learning (지도학습)
- Clustering (군집화) : unsupervised learning (비지도학습)



python™

데이터 분석 순서



python™

데이터 전처리

- 데이터 전처리
  데이터 분석을 하기 위해 분석하기 좋은 형태로 데이터를 가공하는 일

  모델을 만들기 전에 하는 매우매우매우매우매우매우 중요한 부분
  전처리에 따라 같은 데이터를 같은 모델을 넣어도 결과가 아예 달라질 수 있다!

- 결측치 제거
- 데이터 타입 변환
- 누락 feature 추가
- 부적합 feature 삭제 등등...

Training / Validation / Test Set

- Training Set
  모델 학습에 이용되는 데이터 셋

- Validation Set
  Training Set으로 만든 모델의 성능을
  측정하는 데이터 셋

- Test Set
  모델을 적용시킬 실제 데이터 셋

  * 주의사항 *
  Test Set는 **절대로** 모델 학습에
  이용 되어서는 안됨!

python™

Cross Validation

- Train Set를 Train Set와 Validation Set로 분류

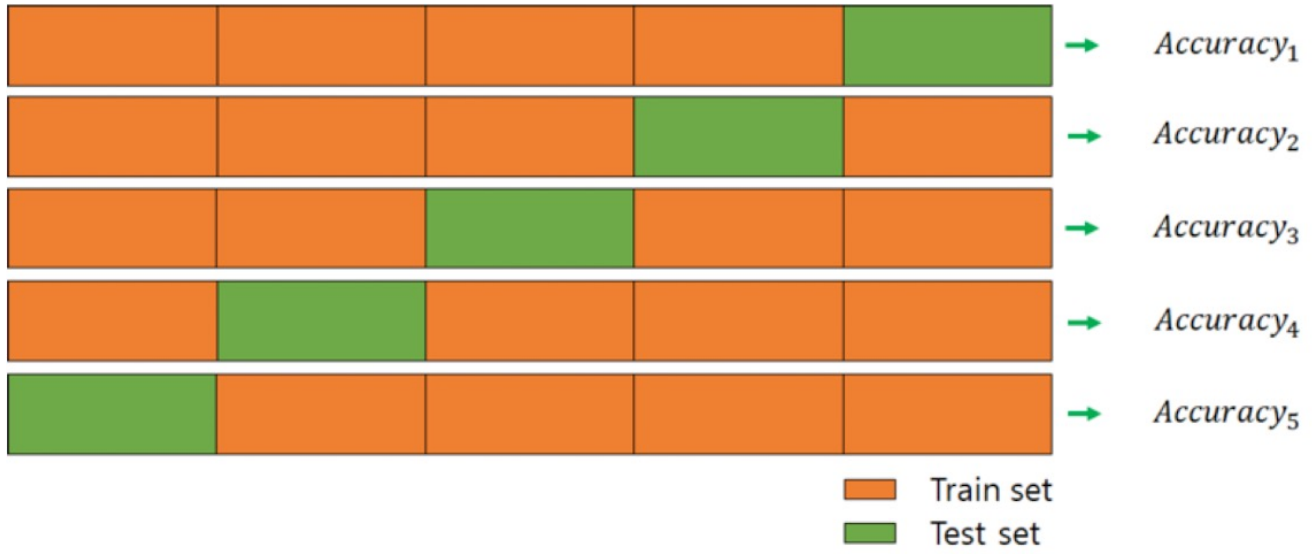
- Validation Set를 기준으로 hyper parameter tuning을 진행하면,
  결국 특정 Validation Set에만 좋은 성능을 보이는 편향된 model을 만들게 됨
  → Overfitting 발생

- 이를 해결하기 위한 방법이 Cross Validation

Cross Validation

- 특정 Validation만이 아닌 전반적으로 좋은 성능을 내기 위해서는
  Train Set의 모든 데이터들을 활용하여 모델을 만들어야 하는 필요성이 있다.

- Cross Validation은 이를 데이터의 모든 부분을 활용하여 모델을 생성하며,
  모든 부분을 활용하여 나온 k개의 평가지표를 평균내서 최종 모델 성능을 판단한다.



python™

Cross Validation 의 장단점

- 장점

  - 모든 Training Set의 데이터를 학습에 활용할 수 있다. (Training Set)
  - 모든 Training Set의 데이터를 평가에 활용할 수 있다. (Validation Set)

- 단점

  시간이 오래 걸린다

Grid Search

- 모델의 하이퍼 파라미터를 넣을 수 있는 값들을 지정한 뒤, 그 중 가장 높은 성능을 보이는 하이퍼 파라미터를 찾아내는 방법

  Ex) 김예원 학생은 A과목을 듣는 당일날 반드시 예습(1시간)과 복습(2시간)을한다.
  '예습'과 '복습'은 모델이 되고, '1시간'과 '2시간'은 하이퍼 파라미터에 해당한다.

- Grid Search는 이 하이퍼 파라미터를 하나하나 탐색해가며, 가장 고성능의 하이퍼 파라미터를 찾아내는 방법이다.

Hyper Parameter

- Hyper Parameter (초매개변수)
  : 모델을 생성할 때 사용자가 직접 설정하는 변수
  - Decision Tree 모델을 이용한다고 할 때, Tree의 개수와 깊이 등을 미리 설정한다.

- Hyper Parameter vs Parameter
  Parameter(매개변수)는 학습 과정에서 생성되는 변수들이다.

  ex) polynomial regression에서 3차원 함수로 regression 한다고 할 때,

  3차원 → Hyper Parameter // 3차원 함수의 계수 → Parameter

Grid Search 코드의 예제

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV

# 데이터를 로딩하고 학습데이타와 테스트 데이터 분리
iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris_data.data, iris_data.target,
                                                    test_size=0.2, random_state=121)
dtree = DecisionTreeClassifier()

### parameter 들을 dictionary 형태로 설정
parameters = {'max_depth':[1,2,3], 'min_samples_split':[2,3]}


# param_grid의 하이퍼 파라미터들을 3개의 train, test set fold 로 나누어서 테스트 수행 설정.
### refit=True 가 default 임. True이면 가장 좋은 파라미터 설정으로 재 학습 시킴.
grid_dtree = GridSearchCV(dtree, param_grid=parameters, cv=3, refit=True)

# 붓꽃 Train 데이터로 param_grid의 하이퍼 파라미터들을 순차적으로 학습/평가 .
grid_dtree.fit(X_train, y_train)

# GridSearchCV 결과 추출하여 DataFrame으로 변환
scores_df = pd.DataFrame(grid_dtree.cv_results_)
scores_df[['params', 'mean_test_score', 'rank_test_score', \
           'split0_test_score', 'split1_test_score', 'split2_test_score']]
```

Grid Search 의 장단점

- 장점

  - 높은 성능의 모델을 만들어낼 수 있다 (Grid Search의 목적)

- 단점

  - 시간이 겁나게 오래걸린다

Confusion Matrix

- Classification에서 사용되는 성능 평가 표

	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

- Accuracy : 정확히 예측한 비율
  = (TP + TN) / (TP + TN + FP + FN)
- Precision : 1로 예측한 것 중 실제로 1인 비율
  = TP / (TP+FP)
- Recall : 1중에서 실제로 1로 예측한 비율
  = TP / (TP + FN)
- False Alarm : 0을 1로 잘못 예측한 비율 (제 1종 오류)
  = FP / (FP + TN)

python™