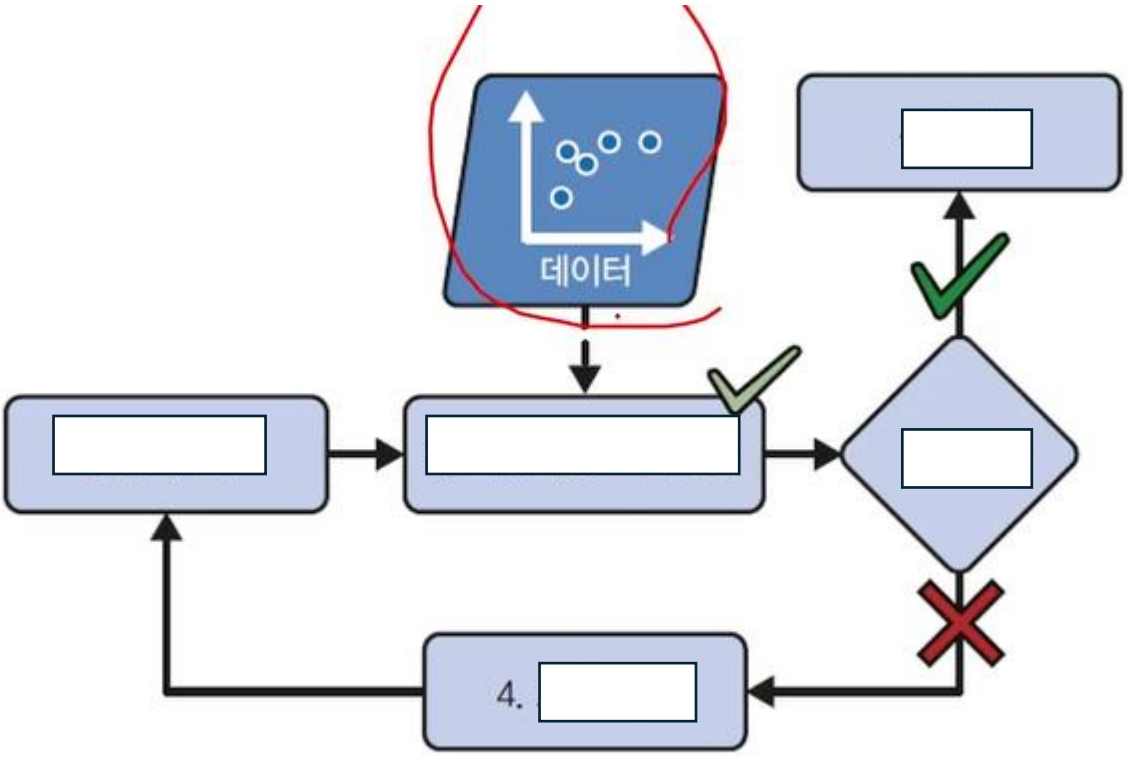
데이터에서 학습하도록 컴퓨터를 프로그래밍 하는 과학의 명칭:

- 대표 예시 1개:

머신러닝 기본 용어 3개: , ,



머신러닝 접근 방법

머신러닝은 자동으로 ( ) 적응한다

대용량의 데이터를 분석하여 숨겨진 패턴을 발견하는 기법: 데이터 마이닝

머신러닝 훈련 지도방식: / / / /

레이블이 있는 지도방식:

레이블이 없는 학습방식:

데이터 없이 행동으로 습득:

시스템이 학습하는 데 사용하는 샘플:

각 훈련 데이터: /

머신러닝 작업 전처리: , ,

탐지 등 3개: , ,

한번에 모아서:

실시간으로 업데이트:

나누어서 하는 학습:

데이터에 따라 적응하는 속도:

시스템이 훈련 샘플을 기억함으로써 학습:

샘플들의 모델을 만들어 예측:

라이브러리

모델이 훈련 데이터에만 잘 맞지만 일반성이 떨어지는 경우:

반대로:

과대적합 방지를 위하여:

검증을 하기 위하여 ( )와 검증 ( )

프로젝트 진행
-> 데이터 구하기 -> 데이터 / -> 데이터 -> ->
-> 솔루션 제시 ->

데이터에 대한 간단한 설명하는 메소드:

테스트 학습 분할:

상관관계 메소드:

누락된 데이터 관리: , ,

중간값으로 대체하는 클래스:

문자로 되어있는 카테고리형 변수를 숫자로 변경하는 클래스: 클래스

입력값이 고정 포인트에서 멀어질수록 출력 값이 지수적으로 감소하는 함수:

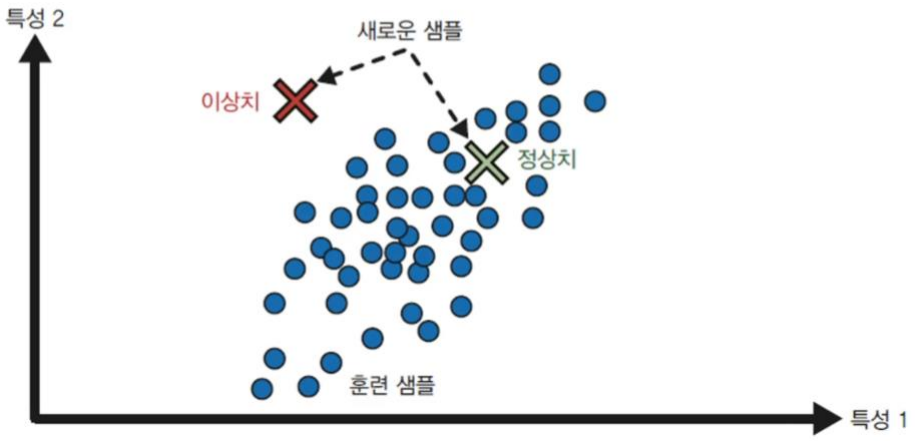
성능 향상 위하여 값 미세조정:

가장 좋은 하이퍼파라미터 값 구하기: .

시간 소요 문제 -> 랜덤화:

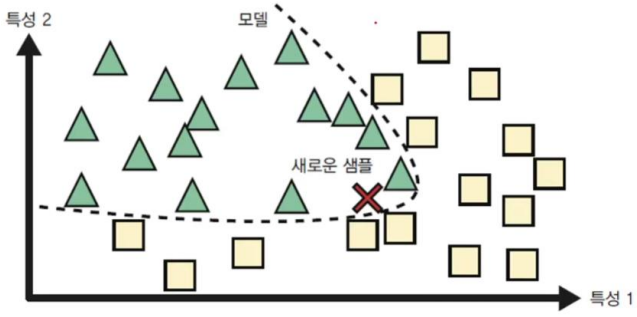
강화 학습은 ( )을 관찰해서 ( ) 을 실행하고 그 결과로 ( ) 또는 ( ) 부과

데이터를 나누어서 하는 학습:



강화학습의 학습하는 시스템을 칭하는 명칭:

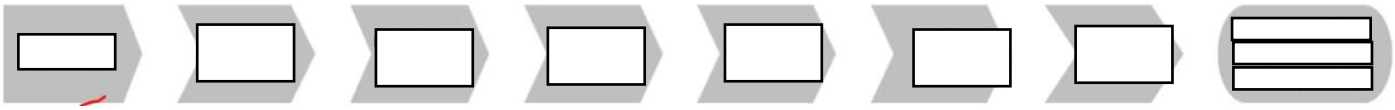
( )



( )

```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.linear_model import LinearRegression

# 데이터를 다운로드하고 준비합니다.
data_root = "https://github.com/ageron/data/raw/main/"
lifesat = pd.read_csv(data_root + "lifesat/lifesat.csv")
X = lifesat[["GDP per capita (USD)"]].values
y = lifesat[["Life satisfaction"]].values

# 데이터를 그래프로 나타냅니다.
lifesat.plot(kind='scatter', grid=True,
             x="GDP per capita (USD)", y="Life satisfaction")
plt.axis([23_500, 62_500, 4, 9])
plt.show()

# 선형 모델을 선택합니다.
model =

# 모델을 훈련합니다.
model.

# 키프로스에 대해 예측을 만듭니다.
X_new = [[37_655.2]]       # 2020년 키프로스 1인당 GDP
print(model(X_new)) # 출력: [[6.30165767]]
```

최근접 이웃 모델 구성: model = (= 3)



성능 측정 지표 대표 2종류: ,

```
>>> housing.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

가장 먼저 수정이 필요한 컬럼은?

데이터에 관한 간략한 설명을 보여주는 메소드:

각 카테고리마다 몇 개가 있는지 보여주는 메소드:

숫자형 특성의 요약 정보를 보여주는 메소드:

특성간 표준 상관관계를 보여주는 메소드:

n개의 중복되지 않은 서브셋을 랜덤으로 분할하는 메소드:

모든 A/B 쌍에 대해 클래스 A의 샘플이 클래스 B로 분류된 횟수를 세는 것:

- 만드는 함수:

```
ray([[53892,  687],
     [ 1891, 3530]])
```


오차 행렬로 만들 수 있는 것: ,

양성 예측의 정확도:

분류기가 정확하게 감지한 양성 샘플의 비율:

거짓 양성 비율에 대한 진짜 양성 비율의 곡선:

완벽한 분류기는 ROC의 AUC가 ( ) 이 됨

매우 강력하고 선형 비선형 회귀 이상치 탐색에도 사용할 수 있는 다목적 머신러닝 모델:

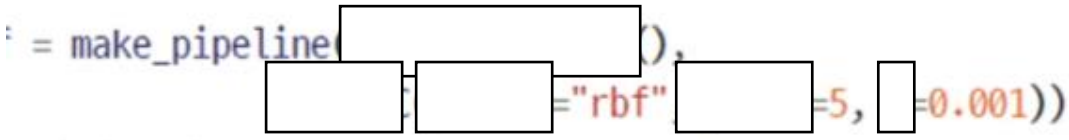
모든 데이터가 확실하게 분류되어야 하는 분류:

어느정도 오류는 유연:

소프트함을 정하기 위한 하이퍼파라미터 약어:

비선형 SVM 분류:

비선형 데이터를 처리할 수 있는 커널 함수:



서포트 벡터로 분류하는 약어:

서포트 벡터로 수치 예측:

객관식 15개

단답/주관 3개

서술형 2개