

* Naïve Bayes Classifier

$P(B|A)$ 를 알 때, $P(A|B)$ 를 아는 방법

↳ A일 때 B인 경우

$$P(B|A) = \frac{P(A \cap B)}{P(A)}, P(A) > 0$$

$$P(A \cap B) = P(B|A) \times P(A)$$

$P(H)$ 는 어떤 사건이

발생했단 주장에 대한 진위로

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A) \times P(A)}{P(B)} \quad P(H|E) \text{ 새로운 정보를 받을 때} \\ \text{증거로 신뢰도 } P(H|E) \text{를 } P(H) \text{로 갱신}$$

$$\text{사후 확률} = \frac{\text{조건부 확률}}{\text{evidence}} \times \text{사전 확률}$$

신뢰도 갱신

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)} \quad : \text{새로운 증거 추가되면} \\ \text{기존 예측 확률 개신팀}$$

Laplace smoothing (Add-one Smoothing)

+1 을 하는 이유 : zero count problem 해결하고자.

TPR 진양성

FPR 위양성

F1 SCORE 재현율과 정밀도의 조화평균

└ 모델이 TRUE 라고 분류한 것 중에서 실제로 TRUE 인 비율

object → 문자형

delays_df.sample(n=10) 랜덤으로 10개 나옴

<data와 target을 분리>

X = delays_df.loc[:, ['DAY_WEEK', 'CRS_DEP_TIME', ...]]
 모든 데이터에 대해 ↗ 01 feature 보여줘.

X.head(10)

y = delays_df.loc[:, ['flight_status']]
 모든 데이터에 대해 ↗ 01 feature 보여줘.

y.sample(10)

자료 < 수치형 - quantitative data : 연속, 이산
 범주형 - qualitative data : 순위형, 명목형
 A+B+ A, B, O, FMS

문자형 자료는 범주형으로 인코딩

X['CRS_DEP_TIME'] = round(X['CRS_DEP_TIME']/100)

↳ 시간을 변환하는 것임 (수를 작게 만들기)

X['CRS_DEP_TIME'] = X['CRS_DEP_TIME'].astype('category')

↳ 출발예정시간을 범주형 변수로 변환

수치형 → 숫자. 값 사이에 다른 값 가능 키, 체온, 혈압...

범주형 → '기준', 동시에 여러곳에 1번/2번/3번, A/B/O/AB형, 수/우/이/양/가
 속하지 X

숫자를 범주형으로 바꾸는 것 category 사용
(int를 범주형으로 바꾸기)

drop_first = "True" → A, B, O 아니면 AB 모두 ~ 초기 해제

<target 레이블 인코딩>

연착을 중요 변수로 설정 : 정상(0), 연착(1)

y = y.replace(['on_time', 'delayed'], [0, 1])

<파이썬>

DATE

01

- 넘파이

import numpy as np

ndarray 데이터가 생성, 숫자, 값, 불 등 모두 가능

↳ 같은 데이터 타입만 가능

array2 = array1.reshape(1,5) 차원조정

↳ np.sort() 정렬형태로 반환

ndarray.sort() 원본 행렬을 변환

- 판다스

에서 데이터를 CSV 형식

import pandas as pd → 캐글 다운로드

- 사이킷런 . 파이썬 ML 라이브러리

pip install scikit-learn = 1.0.2

import sklearn

- 지도학습

명확한 정답이 주어진 데이터를 먼저 학습한 뒤, 미지의 정답을 예측함

별도로 데이터 세트를 테스트 세트로 지정해줌

X-train, X-test, y-train, y-test

= train-test-split (iris_data, iris_label, test_size=0.2, random_state

피처 데이터 세트 ↓ 테스트 세트의 비율 ↓ = 11)

레이블 데이터 세트 ↓ 난수 발생값.

학습데이터 확보 이후

① 데이터 세트 분리 : 학습데이터와 테스트데이터로 분리

DecisionTreeClassifier 객체 생성

② 모델학습 : 학습데이터 기반으로 ML 알고리즘

* 학습수행 dt_clf.fit(X_train, y_train) ③ 예측수행 : 레이터의 분류 예측

* 학습 완료된 객체에서 테스트 데이터 세트로 예측 수행 ④ 평가 : 실제 결과값과 비교하여

pred = dt_clf.predict(X-test)

평가

↳ 테스트 데이터 사용

- 사이킷런의 메서드

`fit()` → ML 모델 학습

`predict()` → 학습된 모델 예측

- 사이킷런의 주요 모듈

`sklearn.datasets` 예제 데이터 세트

`sklearn.preprocessing` 전처리 기능 제공

`sklearn.feature_selection` 큰 영향 미치는 피처를 우선순위로 작업 수행

`sklearn.feature_extraction` 텍스트, 이미지 등 벡터화된 피처를 측정

`sklearn.decomposition` 피처 처리, 차원축소

`sklearn.naive_bayes` 나이브 베이즈 알고리즘 제공

- 피처 처리 → ML 알고리즘 학습 / 예측 수행 → 평가

- 분류와 클러스터링을 위한 표본 데이터 생성기

`datasets.make_classification()` : 무작위 데이터 생성

`datasets.make_blobs()`

- target

분류 : 레이블값

회귀 : 숫자 결과값

- 데이터의 key는 피처들의 데이터 값

- 학습 / 테스트 데이터 세트를 분리 안하면 예측 정확도 100%

↳ 왜? 이미 학습한 데이터 세트를 기반으로 예측했기에

`test_size` 테스트 데이터 세트의 크기

↳ 해결책 K폴드 교차검증

- 데이터 인코딩

↳ 카테고리 피처를 특정 숫자로 변환하는 것 TV: 1 방송고 2 ...

↳ LabelEncoder 객체로 생성한 후 `fit()` 과 `transform()`

호출하여 레이블 인코딩 수행

- 판다스

One-Hot Encoding

고유값 해당 \rightarrow 1 표시, 이외 0 표시

get_dummies() 이용시 더 편리