

## Chapter 02. 사이킷런으로 시작하는 머신러닝

### ✓ 01. 사이킷런 소개와 특징

사이킷런: 파이썬 머신러닝 라이브러리 중 가장 많이 사용되는 라이브러리, 파이썬을 위한 가장 쉽고 효율적인 개발 라이브러리를 제공

- 특징:
  - 파이썬 기반의 다른 머신러닝 패키지도 사이킷런 스타일의 API를 지향할 정도로 쉽고 가장 파이썬스러운 API를 제공
  - 머신러닝을 위한 매우 다양한 알고리즘과 개발을 위한 편리한 프레임워크와 API를 제공
  - 오랜 기간 실전 환경에서 검증되었으며, 매우 많은 환경에서 사용되는 성숙한 라이브러리
- Anaconda를 설치하면 기본으로 사이킷런까지 설치 완료, 별도로 다시 설치해야 한다면 pip와 Anaconda의 conda 명령어를 통해 가능

```
conda install scikit-learn
```

```
pip install scikit-learn
```

### ✓ 02. 첫번째 머신러닝 만들어보기 - 붓꽃 품종 예측하기


- 분류: 대표적 지도학습 방법의 하나
- 지도학습: 다양한 피쳐와 분류 결정값인 레이블 데이터로 모델을 학습한 뒤, 별도의 테스트 데이터 세트에서 미지의 레이블을 예측 (명확한 정답이 주어진 데이터를 먼저 학습한 뒤 미지의 정답을 예측하는 방식)
  - 학습 데이터 세트: 학습을 위해 주어진 데이터 세트
  - 테스트 데이터 세트: 머신러닝 모델의 예측 성능을 평가하기 위해 별도로 주어진 데이터 세트

- 사이킷런 패키지 내의 모듈명

- `sklearn.datasets`: 사이킷런에서 자체적으로 제공하는 데이터 세트를 생성하는 모듈의 모임
- `sklearn.tree`: 트리 기반 ML 알고리즘을 구현한 클래스의 모임
- `sklearn.model_selection`: 학습 데이터와 검증 데이터, 예측 데이터로 데이터를 분리하거나 최적의 파라미터로 평가하기 위한 다양한 모듈의 모임
  - 하이퍼 파라미터: 머신러닝 알고리즘별로 최적의 학습을 위해 직접 입력하는 파라미터들, 머신러닝 알고리즘의 성능 튜닝 가능

- 학습용 데이터와 테스트용 데이터의 분리

- 학습 데이터로 학습된 모델이 얼마나 뛰어난 성능을 가지는지 평가하려면 테스트 데이터 세트가 필요
- 사이킷런은 `train_test_split()` API를 제공, 학습 데이터와 테스트 데이터를 `test_size` 파라미터 입력 값의 비율로 쉽게 분할
  - ex) `test_size=0.2`면 전체 중 테스트 데이터가 20%, 학습 데이터가 80%
- `X_train, X_test, y_train, y_test = train_test_split(iris_data, iris_label, test_size=`

◀  ▶

- `iris_data`: 피쳐 데이터 세트, `iris_label`: 레이블 데이터 세트, `test_size=0.2`: 전체 데이터 세트 중 테스트 데이터 세트의 비율, `random_state`: 호출 시 같은 학습/테스트 용 데이터 세트를 생성하기 위해 주어지는 난수 발생 값

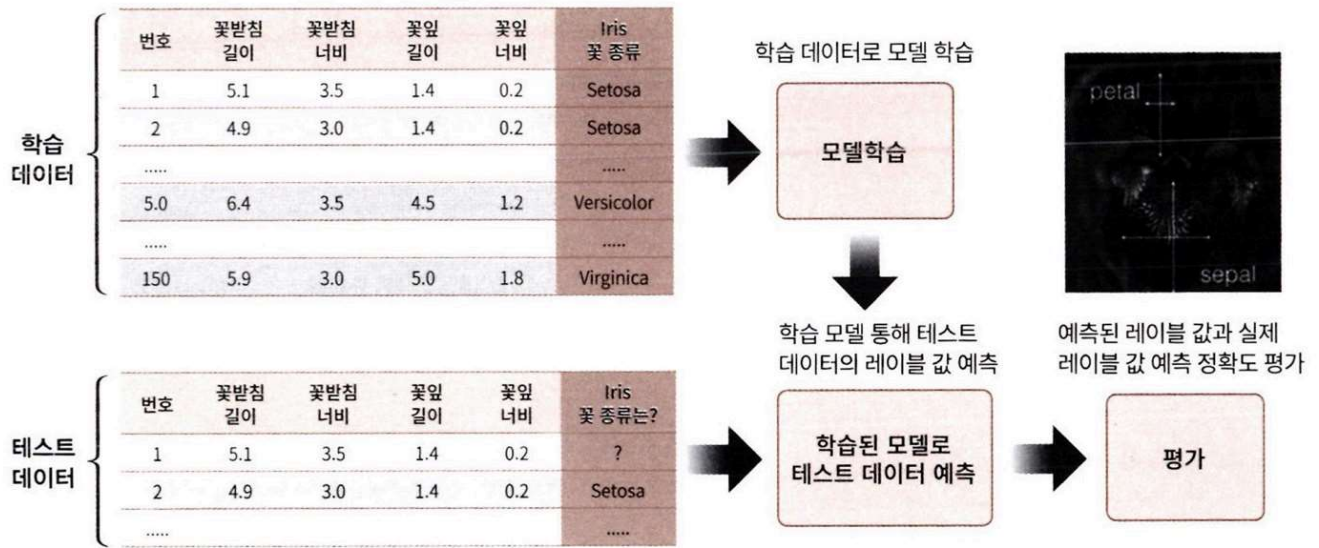
- 의사 결정 트리를 이용해 학습과 예측 수행

- 학습:
  - 의사 결정 트리 클래스 `DecisionTreeClassifier`를 객체로 생성
  - 생성된 객체의 `fit()` 메서드에 학습용 피쳐 데이터 속성과 결정 값 데이터 세트를 입력해 호출하면 학습을 수행
- 예측:
  - 예측은 반드시 학습 데이터가 아닌 다른 데이터 이용, 일반적으로 테스트 데이터 세트
  - 객체의 `predict()` 메서드에 테스트용 피쳐 데이터 세트를 입력해 호출하면 학습된 모델 기반에서 테스트 데이터 세트에 대한 예측값을 반환

- 예측 성능 평가

- 정확도 측정, 정확도는 예측 결과가 실제 레이블 값과 얼마나 정확하게 맞는지를 평가

- `accuracy_score()` 함수를 제공, 첫 번째 파라미터로 실제 레이블 데이터 세트, 두 번째 파라미터로 예측 레이블 데이터 세트 입력



〈붓꽃 데이터 세트 기반의 ML 분류 예측 수행 프로세스〉

### ✓ 03. 사이킷런의 기반 프레임워크 익히기

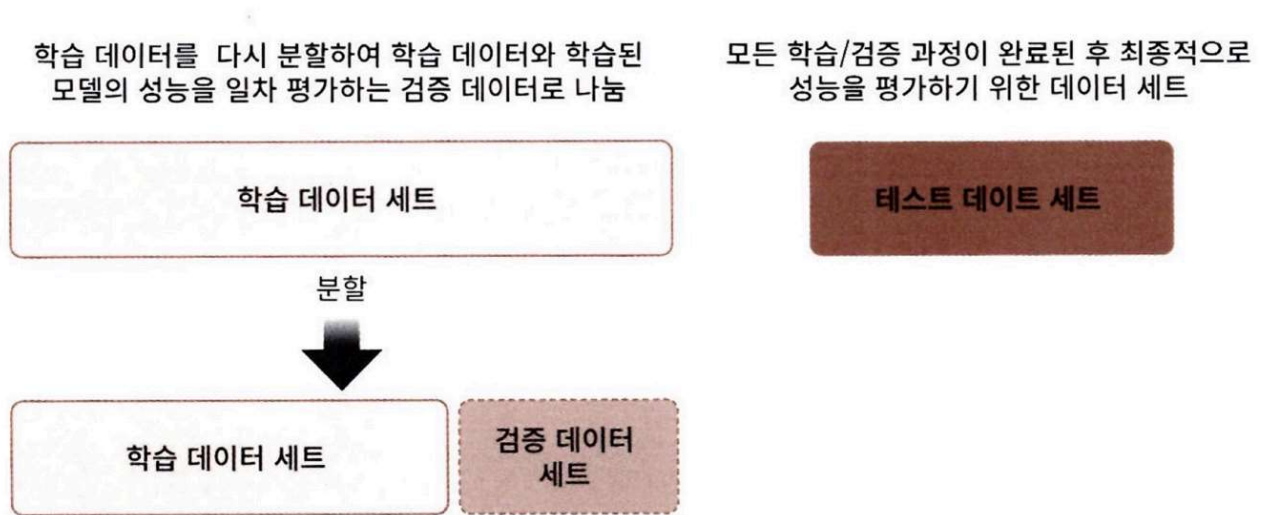
- Estimator 클래스 이해
  - Classifier와 Regressor을 합친 것, 지도학습의 모든 알고리즘을 구현한 클래스를 통칭
    - Classifier(분류): 분류 알고리즘을 구현한 클래스
    - Regressor(회귀): 회귀 알고리즘을 구현한 클래스
  - `fit()`와 `predict()`를 내부에서 구현
    - `fit()` 메서드: ML 모델 학습을 위함
    - `predict()` 메서드: 학습된 모델의 예측을 위함
- 비지도학습인 차원 축소, 클러스터링, 피쳐 추출 등을 구현한 클래스
  - `fit()`와 `transform()` 적용
    - 비지도학습, 피쳐 추출에서의 `fit()`은 입력 데이터의 형태에 맞춰 데이터를 변환하기 위한 사전 구조를 맞추는 작업
    - 입력 데이터의 차원 변환, 클러스터링, 피쳐 추출 등의 실제 작업은 `transform()`으로 수행
    - 사이킷런은 둘을 하나로 결합한 `fit_transform()`도 함께 제공
- 사이킷런의 주요 모듈

- 예제 데이터
  - `sklearn.datasets`: 사이킷런에 내장되어 예제로 제공하는 데이터 세트
- 피처 처리
  - `sklearn.preprocessing`: 데이터 처치리에 필요한 다양한 가공 기능 제공
  - `sklearn.feature_selection`: 알고리즘에 큰 영향을 미치는 피처를 우선순위대로 선택 작업을 수행하는 다양한 기능 제공
  - `sklearn.feature_extraction`: 텍스트 데이터나 이미지 데이터의 벡터화된 피처를 추출하는데 사용됨
- 피처 처리와 차원 축소
  - `sklearn.decomposition`: 차원 축소와 관련한 알고리즘을 지원
- 데이터 분리, 검증과 파라미터 튜닝
  - `sklearn.model_selection`: 교차 검증을 위한 학습용/테스트용 분리
- 평가
  - `sklearn.metrics`: 분류, 회귀, 클러스터링, 페어와이즈에 대한 다양한 성능 측정 방법 제공
- ML 알고리즘
  - `sklearn.ensemble`: 앙상블 알고리즘 제공(랜덤 포레스트, 에이다 부스트, 그래디언트 부스팅 등)
  - `sklearn.linear_model`: 회귀 관련 알고리즘 지원
  - `sklearn.naive_bayes`: 나이브 베이즈 알고리즘 제공
  - `sklearn.neighbors`: 최근접 이웃 알고리즘 제공
  - `sklearn.svm`: 서포트 벡터 머신 알고리즘 제공
  - `sklearn.tree`: 의사 결정 트리 알고리즘 제공
  - `sklearn.cluster`: 비지도 클러스터링 알고리즘 제공
- 유틸리티
  - `sklearn.pipeline`: 피처 처리 등의 변환과 ML 알고리즘 학습, 예측 등을 함께 묶어서 실행할 수 있는 유틸리티 제공
- 내장된 예제 데이터 세트
  - 분류나 회귀를 연습하기 위한 예제용도 데이터 세트
    - 분류 용도: `datasets.load_breast_cancer()`, `datasets.load_digits()`, `datasets.load_iris()`
    - 회귀 용도: `datasets.load_boston()`, `datasets.load_diabetes()`
  - 분류나 클러스터링을 위해 표본 데이터로 생성될 수 있는 데이터 세트
    - 분류를 위한: `datasets.make_classifications()`

- 클러스터링을 위한: `datasets.make_blobs()`
- 대부분 딕셔너리 형태의 값을 반환
- fetch 계열의 명령
  - 데이터의 크기가 커서 패키지에 저장되어 있지 않고 인터넷에서 내려받아야 하는 데이터
  - `fetch_covtype()`, `fetch_20newsgroups()` 등
- 키
  - `data`, `target`, `target_names`, `feature_names`, `DESCR`로 구성
    - 넘파이 배열 타입: `data`, `target`
    - 넘파이 배열 또는 파이썬 리스트 타입: `target_names`, `feature_names`
    - 스트링 타입: `DESCR`
  - 피처의 데이터 값을 반환받기 위해서는 내장 데이터 세트 API 호출 후 Key 값 지정

## ✓ 04. Model Selection 모듈 소개

- 학습 데이터와 테스트 데이터 세트를 분리하거나 교차 검증 분할 및 평가, Estimator의 하이퍼 파라미터를 튜닝하기 위한 다양한 함수와 클래스를 제공
- `train_test_split()`: 학습/테스트 데이터 세트 분리
  - 예측을 수행하는 데이터 세트는 학습을 수행한 학습용 데이터 세트가 아닌 전용의 테스트 데이터 세트여야 함
  - 첫 번째 파라미터 피처 데이터 세트, 두 번째 파라미터 레이블 데이터 세트
  - 선택적 파라미터: `test_size`, `train_size`, `shuffle`, `random_state`
- 교차 검증
  - 과적합으로 인한 성능 저하 문제 개선을 위함
    - 과적합: 모델이 학습 데이터에만 과도하게 최적화되어, 실제 예측을 다른 데이터로 수행할 경우에는 예측 성능이 과도하게 떨어지는 것
  - 비유하자면 본고사를 치르기 전에 모의고사를 여러 번 보는 것
  - 데이터 편증을 막기 위해 별도의 여러 세트로 구성된 학습 데이터 세트와 검증 데이터 세트에서 학습과 평가를 수행하는 것



- K 폴드 교차 검증

- K개의 데이터 폴드 세트를 만들어서 K번만큼 각 폴드 세트에 학습과 검증 평가를 반복적으로 수행
- 학습 데이터 세트와 검증 데이터 세트를 점진적으로 변경하면서 수행
- 사이킷런에서는 KFold와 StratifiedKFold 클래스 제공
- KFold 객체는 `split()`을 호출하면 학습용/검증용 데이터로 분할할 수 있는 인덱스를 반환

- Stratified K 폴드

- 불균형한 분포도를 가진 레이블 데이터 집합을 위한 K 폴드 방식
  - 불균형한 분포도: 특정 레이블 값이 특이하게 많거나 매우 적어서 값의 분포가 한쪽으로 치우치는 것
- K 폴드가 레이블 데이터 집합이 원본 데이터 집합의 레이블 분포를 학습 및 테스트 세트에 제대로 분배하지 못하는 경우의 문제를 해결
- 원본 데이터의 레이블 분포를 먼저 고려한 뒤 이 분포와 동일하게 학습과 검증 데이터 세트를 분배
- StratifiedKFold는 `split()` 메서드에 인자로 피쳐 데이터 세트 뿐만 아니라 레이블 데이터 세트도 반드시 필요
- 일반적으로 분류에서의 교차 검증은 Stratified K 폴드로 분할되어야 함
- 회귀에서는 지원되지 않음, 결정값이 연속된 숫자값이기 때문

- `cross_val_score()`

- 폴드 세트 설정, for 루프에서 반복으로 학습 및 테스트 데이터의 인덱스 추출, 반복적으로 학습과 예측을 수행하고 예측 성능 반환의 과정을 한꺼번에 수행
- `estimator`, `X`, `y`, `scoring`, `cv`가 주요 파라미터

- estimator: 사이킷런의 분류 알고리즘 클래스 또는 회귀 알고리즘 클래스
- X: 피쳐 데이터 세트
- y: 레이블 데이터 세트
- scoring: 예측 성능 평가 지표
- cv: 교차 검증 폴드 수
- cv로 지정된 횟수만큼 scoring 파라미터로 지정된 평가 지표로 평가 결과값을 배열로 반환, 이를 평균해 평가 수치로 사용
- cross\_validate()와 비슷
  - cross\_val\_score()과 다르게 여러 개의 평가 지표 반환 가능
  - 학습 데이터에 대한 성능 평가 지표와 수행 시간도 같이 제공
- GridSearchCV - 교차 검증과 최적 하이퍼 파라미터 튜닝을 한 번에
  - 하이퍼 파라미터를 순차적으로 입력하면서 편리하게 최적의 파라미터를 도출할 수 있는 방안을 제공
  - 순차적으로 파라미터를 테스트하므로 수행시간이 상대적으로 오래 걸리는 것에 유념해야 함
  - 주요 파라미터
    - estimator: classifier, regressor, pipeline 사용 가능
    - param\_grid: key + 리스트 값을 가지는 딕셔너리가 주어짐
    - scoring: 예측 성능을 측정할 평가 방법을 지정
    - cv: 교차 검증을 위해 분할되는 학습/테스트 세트의 개수를 지정
    - refit: 디폴트가 True, True로 생성 시 가장 최적의 하이퍼 파라미터를 찾은 뒤 estimator 객체를 해당 하이퍼 파라미터로 재학습
  - fit 메서드
    - 학습 데이터 세트를 인자로 입력
    - 수행하면 학습 데이터를 cv에 기술된 폴딩 세트로 분할해 param\_grid에 기술된 하이퍼 파라미터를 순차적으로 변경하면서 학습/평가를 수행, 결과를 cv\_results\_ 속성에 기록
      - cv\_results\_는 gridsearchcv의 결과 세트로, 딕셔너리 형태로 key 값과 리스트 형태의 value 값을 가짐
    - 최고 성능을 나타낸 하이퍼 파라미터 값과 그때의 평가 결과 값이 각각 best\_params\_, best\_score\_ 속성에 기록

## ✓ 05. 데이터 전처리

- 데이터에 대해 미리 처리해야 할 기본 사항
  - 결손값(NaN, Null) 값은 허용되지 않음

- 피쳐 값 중 Null 값이 얼마 되지 않으면 평균값 등으로 간단히 대체 가능
  - Null 값이 대부분이라면 해당 피쳐는 드롭하는 것이 좋음
- 문자열 값을 입력 값으로 허용하지 않음
  - 인코딩 돼서 숫자 형으로 변환해야 함
  - 문자열 피쳐: 일반적으로 카테고리형과 텍스트형
    - 텍스트형은 피쳐 벡터화 등의 기법으로 벡터화하거나 불필요한 피쳐라고 판단되면 삭제
- 레이블 인코딩
  - 카테고리 피쳐를 코드형 숫자 값으로 변환하는 것
  - LabelEncoder 클래스로 구현, 객체로 생성한 후 fit()과 transform()을 호출
  - classes\_ 속성은 0번부터 순서대로 변환된 인코딩 값에 대한 원본값을 가짐
  - inverse\_transform()을 통해 인코딩된 값 다시 디코딩 가능
  - 몇몇 ML 알고리즘에는 이를 적용할 경우 예측 성능이 떨어지는 경우 발생
    - 숫자 값의 크고 작음에 대한 특성 때문, 가중치 부여 가능성 존재
    - 선형 회귀에서는 적용 X, 트리 계열은 적용해도 됨
- 원-핫 인코딩
  - 피쳐 값의 유형에 따라 새로운 피쳐를 추가해 고유 값에 해당하는 칼럼에만 1 표시, 나머지는 0
  - OneHotEncoder 클래스로 쉽게 변환 가능
    - 변환 전 모든 문자열 값이 숫자형 값으로 변환되어야 함
    - 입력 값으로 2차원 데이터가 필요
  - get\_dummies() 이용하면 문자열 카테고리 값을 숫자 형으로 변환할 필요 없음
- 피쳐 스케일링과 정규화
  - 피쳐 스케일링: 서로 다른 변수의 값 범위를 일정한 수준으로 맞추는 작업
    - 대표적으로 표준화와 정규화
      - 표준화: 데이터의 피쳐 각각이 평균이 0이고 분산이 1인 가우시안 정규 분포를 가진 값으로 변환
        - $xi\_new = (xi - \text{mean}(x)) / (\text{stdev}(x))$
      - 정규화: 서로 다른 피쳐의 크기를 통일하기 위해 크기를 변환
        - $xi\_new = (xi - \text{min}(x)) / (\text{max}(x) - \text{min}(x))$
  - 전처리에서 제공하는 Normalizer 모듈은 일반 정규화와 다름
    - 개별 벡터의 크기를 맞추기 위해 변환



- $xi\_new = xi / ((xi^2 + yi^2 + zi^2)^{0.5})$
- StandardScaler: 표준화를 쉽게 지원하기 위한 클래스
  - 개별 피처를 평균이 0이고 분산이 1인 값으로 변환
  - RBF 커널을 이용하는 서포트 벡터 머신, 선형 회귀, 로지스틱 회귀는 데이터가 가우시안 분포를 가지고 있다고 가정하고 구현했기 때문에 사전에 표준화를 적용하는 것이 예측 성능 향상에 도움
  - fit()과 transform() 메서드에 변환 대상 피처 데이터 세트를 입력하고 호출하면 간단하게 변환
- MinMaxScaler: 데이터값을 0과 1사이의 범위 값으로 변환 (음수가 있으면 -1에서 1값으로 변환)
  - 데이터의 분포가 가우시안 분포가 아닐 경우에 적용 가능
- Scaler 객체를 이용해 데이터의 스케일링 변환 시 fit(), transform(), fit\_transform() 메소드 이용
  - fit(): 데이터 변환을 위한 기준 정보 설정
  - transform(): 설정된 정보를 이용해 데이터를 변환
  - fit\_transform(): fit()과 transform()을 한번에 적용
- 학습 데이터와 테스트 데이터의 스케일링 변환 시 유의점
  - fit()과 transform() 적용 시 주의 필요
    - Scaler 객체를 이용해 학습 데이터 세트로 fit()과 transform()을 적용하면 테스트 데이터 세트로 다시 fit()을 수행하지 않고 학습 데이터 세트로 fit()을 수행한 결과를 이용해 transform() 변환을 적용해야 함
      - 머신러닝 모델은 학습 데이터를 기반으로 학습되기 때문에 반드시 테스트 데이터는 학습 데이터의 스케일링 기준에 따라야 하며, 테스트 데이터의 1값은 학습 데이터와 동일하게 0.1 값으로 변환되어야 함
  - fit\_transform()은 fit()과 transform()을 순차적으로 수행하는 메소드이므로 테스트 데이터에서는 절대 사용하면 안됨

## ✓ 07. 정리

- 사이킷런은 많은 머신러닝 클래스와 다양한 지원 모듈과 더불어 머신러닝 애플리케이션을 쉽게 구현
  - 머신러닝 애플리케이션은 데이터의 전처리 작업, 데이터 세트 분리 작업을 거친 후, 학습 데이터를 기반으로 머신러닝 알고리즘 적용, 모델을 학습 시킴
  - 학습된 모델을 기반으로 예측 수행 및 모델에 대한 평가
- 데이터 전처리: 데이터 클렌징 작업, 인코딩 작업, 데이터의 스케일링/정규화 작업 등

- 교차 검증: KFold, StratifiedKFold, cross\_val\_score() 등 다양한 클래스와 함수 제공, GridSearchCV 제공

## ✓ Chapter 03. 평가

분류의 성능 평가 지표: 정확도, 오차행렬, 정밀도, 재현율, F1 스코어, ROC AUC

### ✓ 01. 정확도

정확도: 실제 데이터에서 예측 데이터가 얼마나 같은지를 판단하는 지표

- 정확도 = (예측 결과가 동일한 데이터 건수) / (전체 예측 데이터 건수)
- 직관적으로 모델 예측 성능을 나타내는 평가 지표
- 이진 분류의 경우 데이터의 구성에 따라 ML 모델의 성능을 왜곡할 수 있어 정확도 수치 하나만으로 성능을 평가하지 않음
- 불균형한 레이블 값 분포에서 ML 모델의 성능을 판단할 경우, 적합한 평가 지표가 아님
  - 이러한 한계를 극복하기 위해 여러 가지 분류 지표와 함께 적용

### ✓ 02. 오차 행렬

오차 행렬: 학습된 분류 모델이 예측을 수행하면서 얼마나 헛갈리고 있는지도 함께 보여주는 지표

- 4분면 행렬에서 실제 레이블 클래스 값과 예측 레이블 클래스 값이 어떠한 유형을 가지고 매핑되는지를 나타냄
  - 4분면의 왼쪽, 오른쪽을 예측된 클래스 값을 기준으로 Negative와 Positive로 분류, 4분면의 위, 아래를 실제 클래스 값 기준으로 Negative와 Positive로 분류
    - 예측 클래스와 실제 클래스 값 유형에 따라 결정되는 TN, FP, FN, TP 형태로 오차 행렬의 4분면이 채워짐
      - 앞 문자 True/False는 예측값과 실제값이 같은가/틀린가
      - 뒤 문자 Negative/Positive는 예측 결과 값이 부정(0)/긍정(1)
- confusion\_matrix() API 제공
- TP, TN, FP, FN 값은 Classifier 성능의 여러 면모를 판단할 수 있는 기반 정보 제공
  - 정확도, 정밀도, 재현율 값 파악 가능
- 오차 행렬 상의 정확도:  $(TN + TP) / (TN + FP + FN + TP)$

		예측 클래스 (Predicted Class)	
		Negative(0)	Positive(1)
실제 클래스 (Actual Class)	Negative(0)	<b>TN</b> (True Negative)	<b>FP</b> (False Positive)
	Positive(1)	<b>FN</b> (False Negative)	<b>TP</b> (True Positive)

### ✓ 03. 정밀도와 재현율

정밀도와 재현율: Positive 데이터 세트의 예측 성능에 좀 더 초점을 맞춘 평가 지표

정밀도 =  $TP / (FP + TP)$

재현율 =  $TP / (FN + TP)$

- 정밀도: 예측을 Positive로 한 대상 중에 예측과 실제 값이 Positive로 일치한 데이터의 비율
  - 공식의 분모는 Positive로 한 모든 데이터 건수, 분자는 예측 값과 실제 값이 Positive로 일치한 데이터 건수
  - 실제 Negative 음성 데이터를 Positive로 잘못 판단하게 되면 업무상 큰 영향이 발생하는 경우에 중요
- 재현율: 실제 값이 Positive인 대상 중에 예측과 실제 값이 Positive로 일치한 데이터의 비율
  - 공식의 분모는 실제 값이 Positive인 모든 데이터 건수, 분자는 예측 값과 실제 값이 Positive로 일치한 데이터 건수
  - 실제 Positive 양성 데이터를 Negative로 잘못 판단하게 되면 업무상 큰 영향이 발생하는 경우에 중요
- 재현율은 FN을 낮추는 데, 정밀도는 FP를 낮추는 데 초점
- 정밀도/재현율 트레이드오프
  - 한쪽을 강제로 높이면 다른 하나의 수치는 떨어지기 쉬움
  - `predict_proba()` 제공

- 파라미터는 보통 테스트 피쳐 데이터 세트를 입력
- 반환되는 ndarray는 첫 번째 칼럼이 클래스 값 0에 대한 예측 확률, 두 번째 칼럼이 클래스 값 1에 대한 예측 확률
- predict()는 predict\_proba() 호출 결과로 반환된 배열에서 분류 결정 임계값보다 큰 값이 들어 있는 칼럼의 위치를 받아서 최종적으로 예측 클래스를 결정
- Positive 예측값이 많아지면 상대적으로 재현율 값이 높아짐
- precision\_recall\_curve() 제공
  - 입력 파라미터:
    - y\_true: 실제 클래스값 배열
    - probas\_pred: Positive 칼럼의 예측 확률 배열
  - 반환값: 정밀도, 재현율
- 정밀도가 100% 되는 방법
  - 확실한 기준이 되는 경우만 Positive로 예측, 나머지는 모두 Negative로 예측
- 재현율이 100% 되는 방법
  - 전체를 Positive로 예측
- 정밀도와 재현율의 수치가 적절하게 조합되어 분류의 종합적 성능 평가에 사용될 수 있도록 하는 것이 중요!

## ✓ 04. F1 스코어

F1 스코어: 정밀도와 재현율을 결합한 지표  $F1 = 2 / ((1/recall) + (1/precision)) = 2 \times (precision \times recall) / (precision + recall)$

- f1\_score()라는 API 제공

## ✓ 05. ROC 곡선과 AUC

ROC 곡선과 AUC: 이진 분류의 예측 성능 측정에서 중요하게 사용되는 지표

ROC 곡선: FPR이 변할 때 TPR이 어떻게 변하는지를 나타내는 곡선

- FPR을 X축으로, TPR을 Y축으로 잡으면 FPR의 변화에 따른 TPR의 변화가 곡선 형태로 나타남
- TPR은 재현율(민감도)을 의미, TNR은 특이성을 의미
- $TNR = TN / (FP + TN)$
- $FPR = FP / (FP + TN)$ ,  $1 - TNR$ 로 표현 가능
- FPR을 0부터 1까지 변경하면서 TPR의 변화 값을 구함

- FPR을 0으로 만들려면 분류 결정 임계값을 1로 지정하면 됨
- FPR을 1로 만들려면 TN을 0으로 만들면 됨, TN을 0으로 만들려면 분류 결정 임계값을 0으로 지정하면 됨
- roc\_curve() API 제공
  - 입력 파라미터: y\_true, y\_score
  - 반환값: fpr, tpr, thresholds

AUC 값: ROC 곡선 밑의 면적을 구한 것으로서 일반적으로 1에 가까울수록 좋음

- 수치가 커지려면 FPR이 작은 상태에서 얼마나 큰 TPR을 얻을 수 있느냐가 관건
- 가운데 직선에서 멀어지고 왼쪽 상단 모서리 쪽으로 곡선이 가파르게 이동할수록 면적은 1에 가까워짐

## ✓ 07. 정리

- 성능 평가 지표: 정확도, 오차 행렬, 정밀도, 재현율, F1 스코어, ROC-AUC
  - 이진 분류의 레이블 값이 불균형하게 분포될 경우 정확도만으로는 평가 불가
  - 오차 행렬은 TN, FP, FN, TP로 매핑되는 4분면 행렬을 기반으로 평가
  - 정밀도와 재현율은 Positive 데이터 세트의 예측 성능에 좀 더 초점을 맞춘 지표
  - F1 스코어는 정밀도와 재현율을 결합한 평가 지표
  - ROC-AUC의 AUC 값은 ROC 곡선 밑의 면적을 구한 것으로서 1에 가까울수록 좋음