MACHINE LEARNING

Scikit-learn

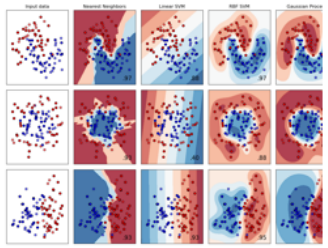
Scikit-lean

▸머신러닝을 위한 매우 다양한 알고리즘과 개발을 위한 편리한 프레임워크와 api를 제공하는 라이브러리

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...

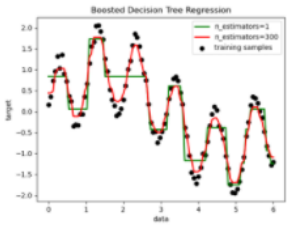


Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...

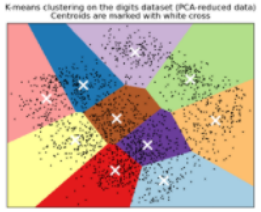


Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...

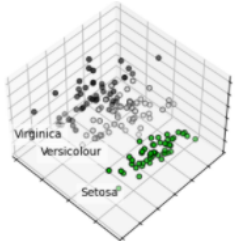


Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: k-Means, feature selection, non-negative matrix factorization, and more...

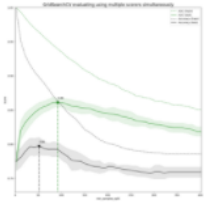


Examples

Model selection

Comparing, validating and choosing parameters and models.

Applications: Improved accuracy via parameter tuning
Algorithms: grid search, cross validation, metrics, and more...

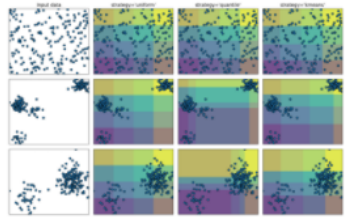


Examples

Preprocessing

Feature extraction and normalization.

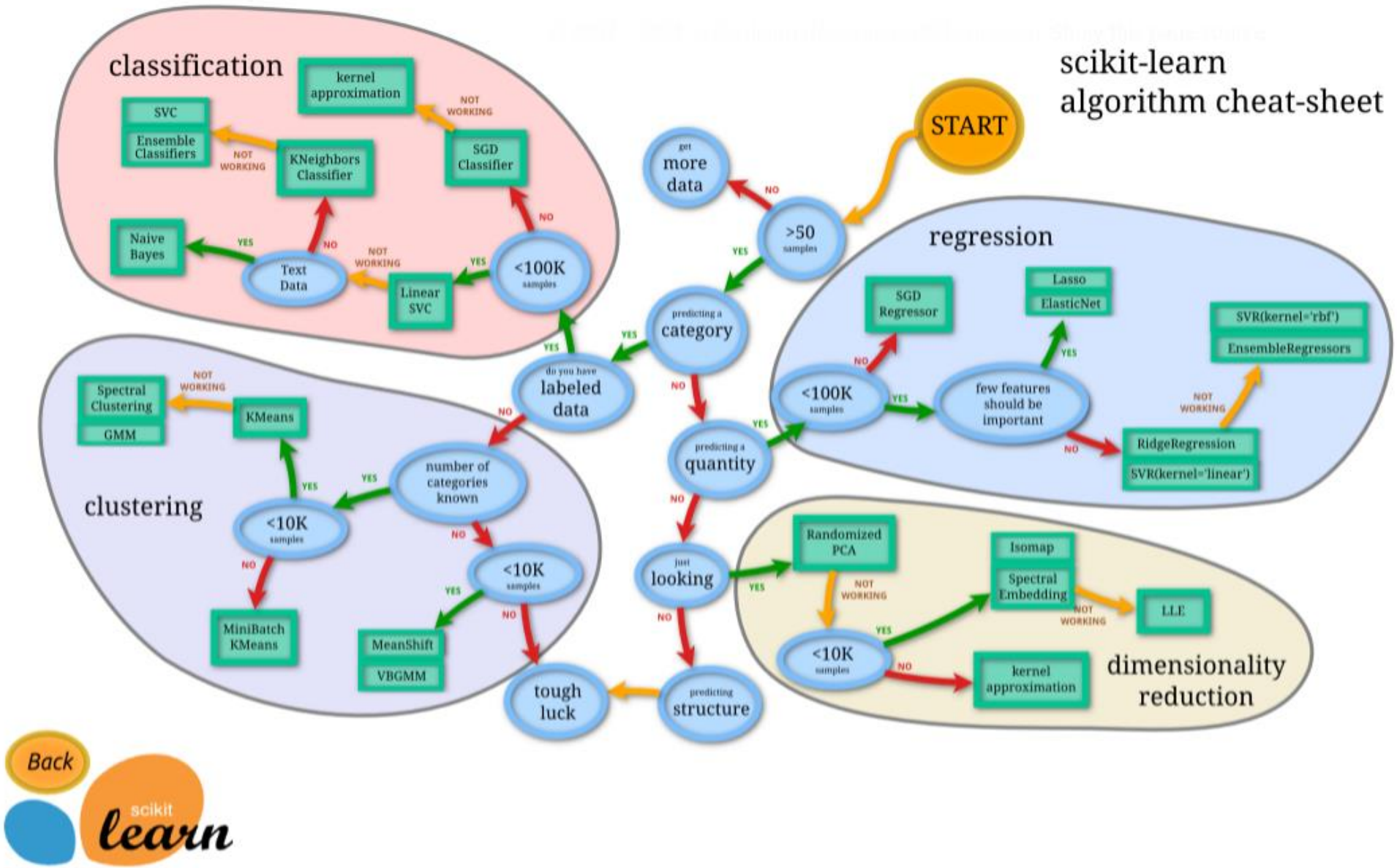
Applications: Transforming input data such as text for use with machine learning algorithms.
Algorithms: preprocessing, feature extraction, and more...



Examples

https://scikit-learn.org/stable/index.html

Scikit-lean https://scikit-learn.org/stable/tutorial/machine_learning_map/



Scikit-lean https://scikit-learn.org/stable/tutorial/machine_learning_map/

모듈	설명
sklearn.datasets	내장된 예제 데이터 세트
sklearn.preprocessing	다양한 데이터 전처리 기능 제공 (변환, 정규화, 스케일링 등)
sklearn.feature_selection	특징(feature)를 선택할 수 있는 기능 제공
sklearn.feature_extraction	특징(feature) 추출에 사용
sklearn.decomposition	차원 축소 관련 알고리즘 지원 (PCA, NMF, Truncated SVD 등)
sklearn.model_selection	교차 검증을 위해 데이터를 학습/테스트용으로 분리, 최적 파라미터를 추출하는 API 제공 (GridSearch 등)
sklearn.metrics	분류, 회귀, 클러스터링, Pairwise에 대한 다양한 성능 측정 방법 제공 (Accuracy, Precision, Recall, ROC-AUC, RMSE 등)
sklearn.pipeline	특징 처리 등의 변환과 ML 알고리즘 학습, 예측 등을 묶어서 실행할 수 있는 유틸리티 제공
sklearn.linear_model	선형 회귀, 릿지(Ridge), 라쏘(Lasso), 로지스틱 회귀 등 회귀 관련 알고리즘과 SGD(Stochastic Gradient Descent) 알고리즘 제공
sklearn.svm	서포트 벡터 머신 알고리즘 제공
sklearn.neighbors	최근접 이웃 알고리즘 제공 (k-NN 등)
sklearn.naive_bayes	나이브 베이즈 알고리즘 제공 (가우시안 NB, 다항 분포 NB 등)
sklearn.tree	의사 결정 트리 알고리즘 제공
sklearn.ensemble	앙상블 알고리즘 제공 (Random Forest, AdaBoost, GradientBoost 등)
sklearn.cluster	비지도 클러스터링 알고리즘 제공 (k-Means, 계층형 클러스터링, DBSCAN 등)

사이킷런의 주요 모듈

분류	모듈명	설명
예제 데이터	sklearn.datasets	사이킷런에 내장되어 예제로 제공하는 데이터 세트
데이터 분리, 검증 & 파라미터 튜닝	sklearn.model_selection	교차 검증을 위한 학습용/테스트용 분리, 그리드 서치(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 피처를 추출하는 데 사용됨. 예를 들어 텍스트 데이터에서 Count Vectorizer 나 Tf-Idf Vectorizer 등을 생성하는 기능 제공. 텍스트 데이터의 피처 추출은 sklearn.feature_extraction.text 모듈에, 이미지 데이터의 피처 추출은 sklearn.feature_extraction.image 모듈에 지원 API가 있음.
피처 처리 & 차원 축소	sklearn.decomposition	차원 축소와 관련한 알고리즘을 지원하는 모듈임. PCA, NMF, Truncated SVD 등을 통해 차원 축소 기능을 수행할 수 있음

분류	모듈명	설명
평가	sklearn.metrics	분류, 회귀, 클러스터링, 페어와이즈(Pairwise)에 대한 다양한 성능 측정 방법 제공 Accuracy, Precision, Recall, ROC-AUC, RMSE 등 제공
ML 알고리즘	sklearn.ensemble	앙상블 알고리즘 제공 랜덤 포레스트, 에이다 부스트, 그래디언트 부스팅 등을 제공
	sklearn.linear_model	주로 선형 회귀, 릿지(Ridge), 라쏘(Lasso) 및 로지스틱 회귀 등 회귀 관련 알고리즘을 지원. 또한 SGD(Stochastic Gradient Descent) 관련 알고리즘도 제공
	sklearn.naive_bayes	나이브 베이즈 알고리즘 제공. 가우시안 NB , 다항 분포 NB 등.
	sklearn.neighbors	최근접 이웃 알고리즘 제공. K-NN 등
	sklearn.svm	서포트 벡터 머신 알고리즘 제공
	sklearn.tree	의사 결정 트리 알고리즘 제공
	sklearn.cluster	비지도 클러스터링 알고리즘 제공 (K-평균, 계층형, DBSCAN 등)
유틸리티	sklearn.pipeline	피처 처리 등의 변환과 ML 알고리즘 학습, 예측 등을 함께 묶어서 실행할 수 있는 유틸리티 제공

Scikit-Learn의 데이타 표현방식

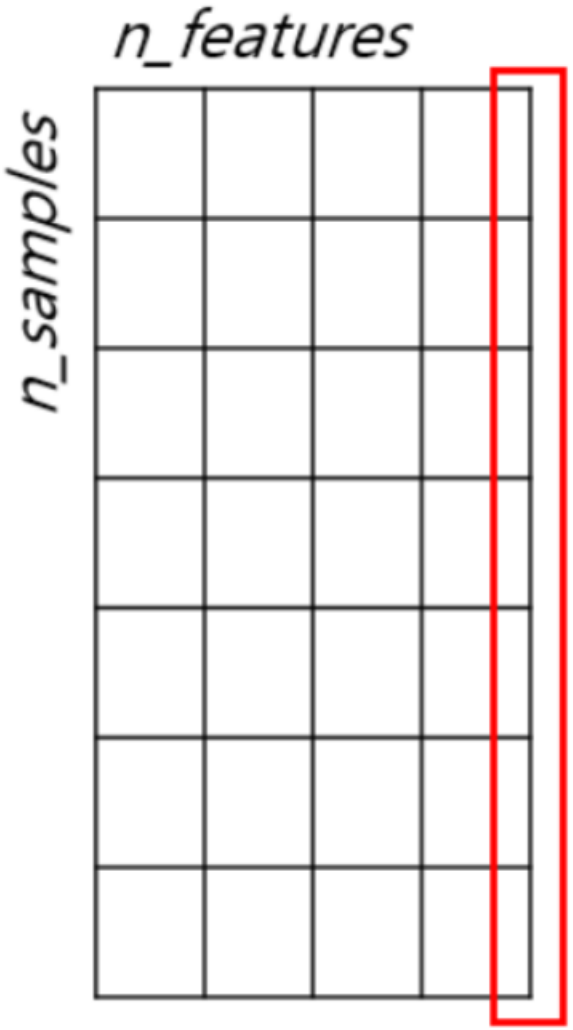
특징행렬(Feature Matrix) : 사이킷런의 데이타셋. 학습 데이타(Train Data)

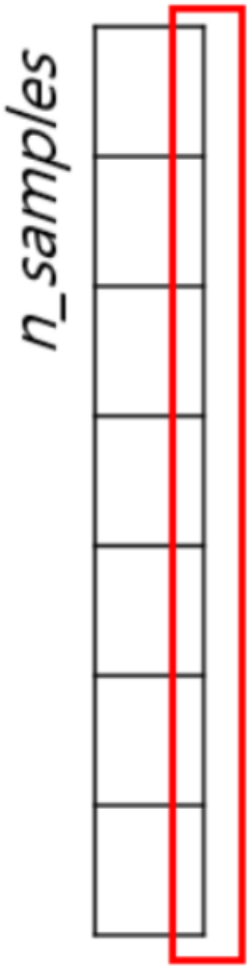
- 표본(Sample) : 개별 객체. 행렬의 행
- 특징(feature) : 연속적인 수치값, 부울값, 이산값으로 표현하는 개별 관측치
- 행의 개수 : n_samples
- 열의 개수 : n_features
- X = [n_samples, n_features] 형태의 2차원 배열 구조를 사용
  (Numpy 배열, Pandas DataFrame, SciPy 희소행렬)

대상 벡터 (Target Vector). 테스트 데이타(Test Data)

- 연속적인 수치값, 이산 클래스/레이블을 가짐
- 길이 n_samples
- 관례적으로 대상벡터는 변수 y에 저장
- 1차원 배열 구조를 사용 (주로 Numpy 배열, Pandas Series를 사용)
- 특징 행렬로부터 예측하고자 하는 값의 벡터
- 종속 변수, 출력 변수, 결과 변수, 반응 변수

머신러닝을 위한 용어 정리





머신러닝 프로세스

데이터를 학습 데이터와 테스트 데이터로 분리합니다

학습 데이터를 기반으로 ML 알고리즘을 적용해 모델을 학습시킵니다

학습된 ML 모델을 이용해 테스트 데이터의 분류(즉, 붓꽃 종류)를 예측합니다

이렇게 예측된 결괏값과 테스트 데이터의 실제 결괏값을 비교해 ML 모델 성능을 평가합니다.

머신러닝 예측 프로세스 : XOR

XOR Gate

A	B	OUT
0	0	0
0	1	1
1	0	1
1	1	0

	0	1	2
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0

```
1 X = df[[0,1]]
2 X
```

	0	1
0	0	0
1	0	1
2	1	0
3	1	1

```
1 y = df[2]
2 y
```

0	0
1	1
2	1
3	0

```
1 # 데이터 학습 => 모델링
2 model = svm.SVC()
3 model.fit(X, y)
```

```
SVC()
```

```
1 # 데이터 테스트 => 예측
2 model.predict(X)
```

```
array([0, 1, 1, 0], dtype=int64)
```

```
1 model.predict([[0,0], [1,0]])
```

```
array([0, 1], dtype=int64)
```

```
1 # 정답률
2 result = metrics.accuracy_score(y, model.predict(X))
3 print(f'정답률은? {result*100} %')
```

```
정답률은? 100.0 %
```

```
1 # 정답률
2 result = metrics.mean_squared_error(y, y)
3 print(f'에러률은? {result*100} %')
```

```
에러률은? 0.0 %
```

사이킷런의 데이타셋

`load_boston`: 보스톤 집값 데이터
`load_iris`: 아이리스 붓꽃 데이터
`load_diabetes`: 당뇨병 환자 데이터
`load_digits`: 손글씨 데이터
`load_linnerud`: multi-output regression 용 데이터
`load_wine`: 와인 데이터
`load_breast_cancer`: 위스콘신 유방암 환자 데이터

```
1  from sklearn.datasets import load_iris
2
3  iris = load_iris()
4  print(type(iris))
5
```

```
<class 'sklearn.utils.Bunch'>
```

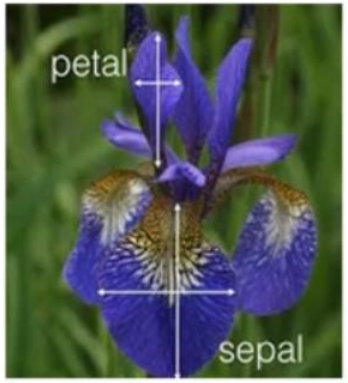
```
1  keys = iris.keys()
2  print('붓꽃 데이터 세트의 키들:', keys)
```

```
붓꽃 데이터 세트의 키들: dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'filename'])
```

- data는 피처의 데이터 세트를 가리킵니다.
- target은 분류 시 레이블 값, 회귀일 때는 숫자 결괏값 데이터 세트입니다..
- target_names는 개별 레이블의 이름을 나타냅니다.
- feature_names는 피처의 이름을 나타냅니다.
- DESCR은 데이터 세트에 대한 설명과 각 피처의 설명을 나타냅니다.

머신러닝 예측 프로세스 : 붓꽃 데이타 예

붓꽃 데이터 피처



- Sepal length
- Sepal width
- Petal length
- Petal width

붓꽃 데이터 품종(레이블)

Setosa

Vesicolor

Virginica

피처 레이블

학습 데이터

꽃받침 길이	꽃받침 너비	꽃잎 길이	꽃잎 너비	Iris 꽃 종류
5.1	3.5	1.4	0.2	Setosa
4.9	3.0	1.4	0.2	Setosa
6.4	3.5	4.5	1.2	Versicolor

테스트 데이터

꽃받침 길이	꽃받침 너비	꽃잎 길이	꽃잎 너비	Iris 꽃 종류는?
5.3	3.2	1.1	0.1	?
4.2	2.0	2.4	0.4	?
6.5	3.8	5.5	1.1	?

머신러닝 예측 프로세스 : 와인 품종 분류기 예

- **이탈리아의 같은 지역내의 3개의 다른 경작지에서 재배된 와인의 화학적 분석결과 데이타셋**
- **https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html**

```
1 from sklearn.datasets import load_wine
2 wine = load_wine()
```

알콜(Alcohol)

말산(Malic acid)

회분(Ash)

회분의 알칼리도(Alcalinity of ash)

마그네슘(Magnesium)

총 폴리페놀(Total phenols)

플라보노이드 폴리페놀(Flavanoids)

비 플라보노이드 폴리페놀(Nonflavanoid phenols)

프로안토시아닌(Proanthocyanins)

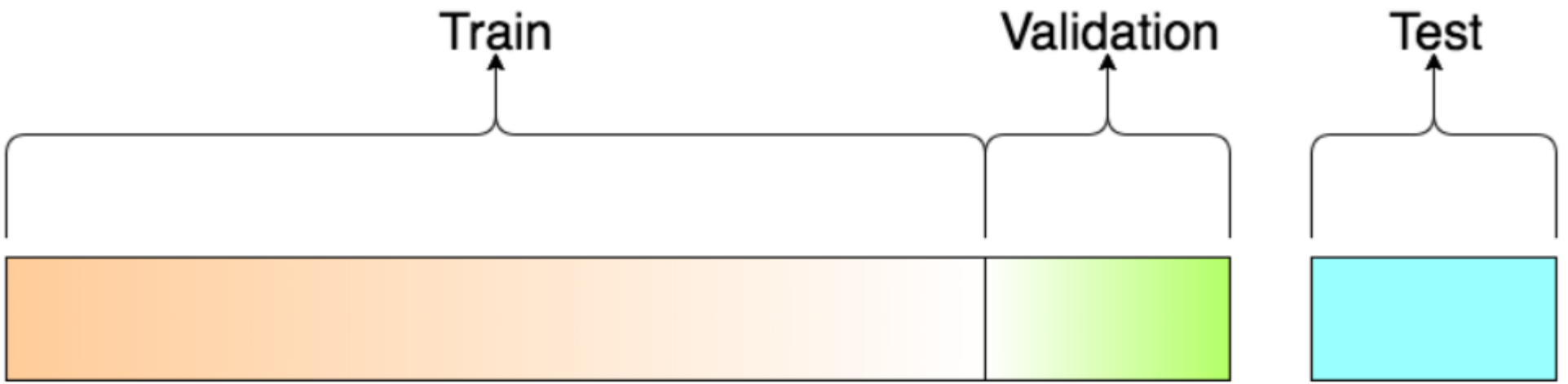
색상의 강도(Color intensity)

색상(Hue)

희석 와인의 OD280/OD315 비율 (OD280/OD315 of diluted wines)

프롤린(Proline)

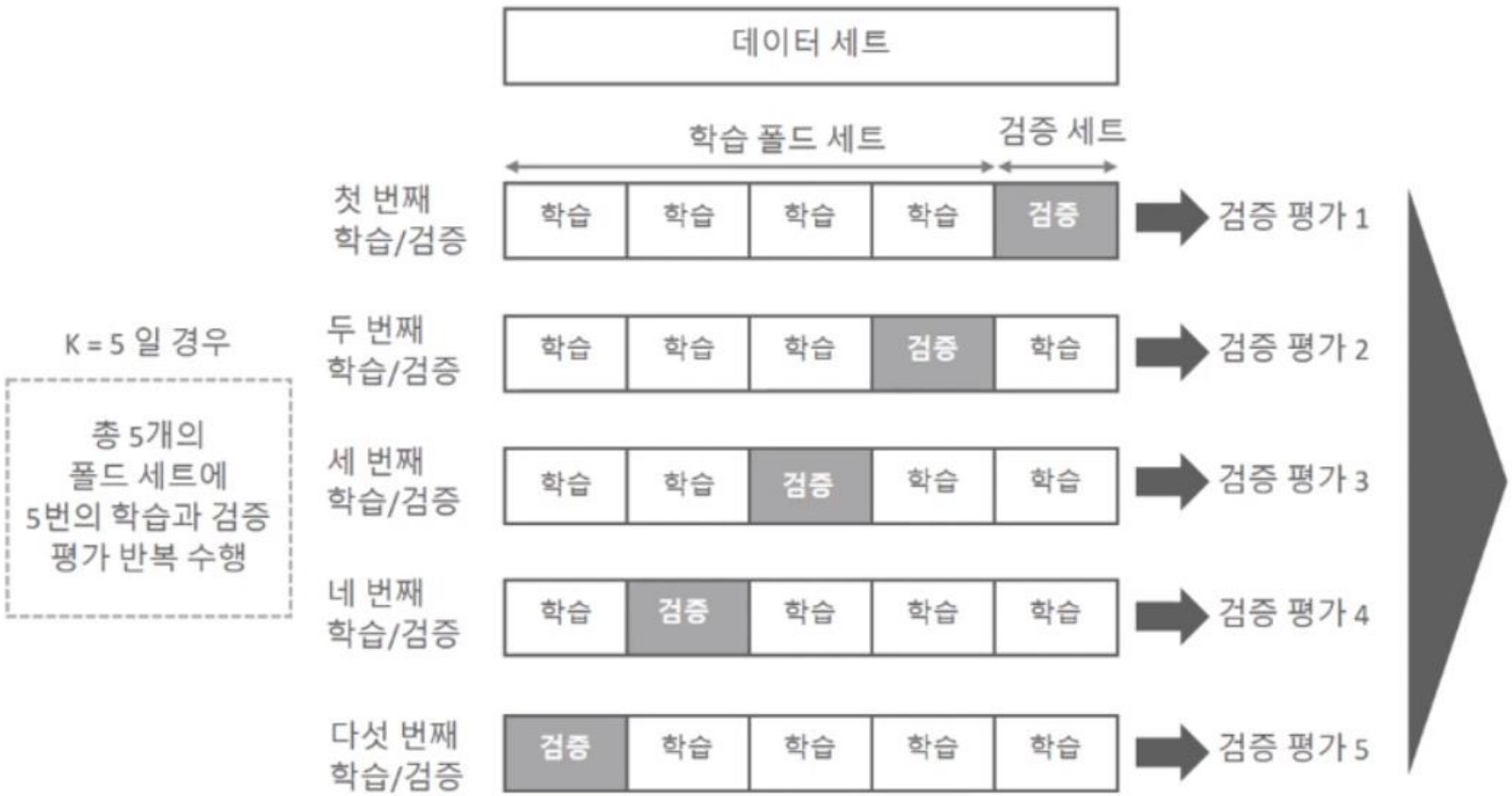
학습과 테스트 세트 분리




```
from sklearn.model_selection import train_test_split

train_test_split( data, label,
                  test_size=float, train_size=float,
                   shuffle=True/False,
                   random_state=Number)
```

K 폴드



- 폴더 세트 설정
- for 루프에서 반복적으로 학습과 테스트 데이터의 인덱스 추출
- 반복적으로 학습과 예측을 수행하고 예측 성능을 반환한다

Stratified K 폴드

- 데이타의 분포도가 유사해야한다는 kFold의 특징을 보완
- 불균형한 분포도를 가진 레이블 데이타 집합을 위한 k폴드 방식
- 불균형한 분포도를 가진 레이블 데이터 집합은 특정 레이블 값이 특이하게 많거나 매우 적어서 레이블의 값의 분포가 한쪽으로 치우친다.

예) 대출사기 데이타. 대출사기 레이블이 전체의 0.0001% 에 밖에 해당하지 못한다.

-분류 모델에는 적합하나 회기에서는 연속된 숫자값이기 때문에 적합하지 못하다.

```
# StratifiedKFold의 split( ) 호출시 반드시 레이블 데이터 셋도 추가 입력 필요
for train_index, test_index  in skfold.split(features, label):
    # 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]

    # 모델 학습 및 예측
    예측기.fit(X_train , y_train)
    pred = dt_clf.predict(X_test)

    # 반복 시 마다 정확도 측정해서 정확도 리스트에 추가
    accuracy = np.round(accuracy_score(y_test,pred), 4)
    cv_accuracy.append(accuracy)
```

cross_val_score( )

사이킷런에서 제공하는 교차 검증을 편리하게 수행할 수 있게 도와주는 API

1. 폴드 세트 설정
2. For 루프에서 반복적으로 학습/검증 데이터 추출 및 학습과 예측 수행
3. 폴드 세트별로 예측 성능을 평균하여 최종 성능 평가

cross_val_score( ) 함수로
폴드 세트 추출, 학습/예측, 평가를
한번에 수행

```
1  iris_data = load_iris()
2  dt_clf = DecisionTreeClassifier(random_state=156)
3
4  data = iris_data.data
5  label = iris_data.target
6
7  # 성능 지표는 정확도(accuracy) , 교차 검증 세트는 3개 => cv
8  scores = cross_val_score(dt_clf , data , label , scoring='accuracy',cv=3)
9
10 print('교차 검증별 정확도:',np.round(scores, 4))
11 print('평균 검증 정확도:', np.round(np.mean(scores), 4))
12
```

```
교차 검증별 정확도: [0.98 0.94 0.98]
평균 검증 정확도: 0.9667
```

GridSearchCV

분류 알고리즘이나 회귀 알고리즘에 사용되는 하이퍼파라미터를 순차적으로 입력해 학습을 하고 측정을 하면서 가장 좋은 파라미터를 알려주는 역할을 한다.

```
grid_parameters = {'max_depth': [1, 2, 3],
'min_samples_split': [2, 3]
}
```

CV 세트가 3 이라면

파라미터 순차 적용 횟수		CV 세트수		학습/검증 총 수행횟수
6	X	3		18

순번	max_depth	min_samples_split
1	1	2
2	1	3
3	2	2
4	2	3
5	3	2
6	3	3

```
(1) 딕셔너리 형태로 파라미터를 지정
parameters = {'max_depth':[1, 2, 3], 'min_samples_split':[2,3]}

(2) GridSearchCV 메서드 적용
grid_dtree = GridSearchCV(dtree, param_grid=parameters, cv=3, refit=True, return_train_score=True)
```

데이터 인코딩

원본 데이터

상품 분류	가격
TV ·	1,000,000
냉장고	1,500,000
전자렌지	200,000
컴퓨터	800,000
선풍기	100,000
선풍기	100,000
믹서	50,000
믹서	50,000

상품 분류를 레이블 인코딩 한 데이터

상품 분류	가격
0	1,000,000
1	1,500,000
4	200,000
5	800,000
3	100,000
3	100,000
2	50,000
2	50,000

[TV, 냉장고, 전자레인지, 컴퓨터, 선풍기, 믹서]

[0, 1, 4, 5, 3, 2]

데이터 인코딩

TV:0, 냉장고:1, 전자레인지:4, 컴퓨터:5, 선풍기: 3, 믹서:2 를 원-핫 인코딩

원본 데이터

상품 분류
TV
냉장고
전자렌지
컴퓨터
선풍기
선풍기
믹서
믹서

원-핫 인코딩

상품분류_TV	상품분류_냉장고	상품분류_믹서	상품분류_선풍기	상품분류_전자렌지	상품분류_컴퓨터
1	0	0	0	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	0	0	0

데이터 인코딩

판다스의 get_dummies()를 이용한 원핫 인코딩

	item
0	TV
1	냉장고
2	전자렌지
3	컴퓨터
4	선풍기
5	선풍기
6	믹서
7	믹서

```
1 # 고유값 레벨로 변경. item_원본레벨값 =
2 pd.get_dummies(df)
```

	item_TV	item_냉장고	item_믹서	item_선풍기	item_전자렌지	item_컴퓨터
0	1	0	0	0	0	0
1	0	1	0	0	0	0
2	0	0	0	0	1	0
3	0	0	0	0	0	1
4	0	0	0	1	0	0
5	0	0	0	1	0	0
6	0	0	1	0	0	0
7	0	0	1	0	0	0

피처 스케일링

표준화(Standardzation)와 정규화(Normalization)

$$x_i\_new = \frac{x_i - mean(x)}{stdev(x)}$$

$$x_i\_new = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

StandardScaler

평균이 0이고 분산이 1인 정규분포 형태로 변환

MinMaxScaler

데이타값을 0과 1 사이의 범위값으로 변환한다. 음수값이 있으면 -1에서 1값으로 변환한다

타이타닉 생존자 예측

데이터 전처리

- Null 처리
- 불필요한 속성 제거
- 인코딩 수행

모델 학습 및 검증/예측/평가

- 결정트리, 랜덤포레스트, 로지스틱 회귀 학습 비교
- K 폴드 교차 검증
- cross_val_score( )와 GridSearchCV( ) 수행

판다스 apply lambda

일반 함수

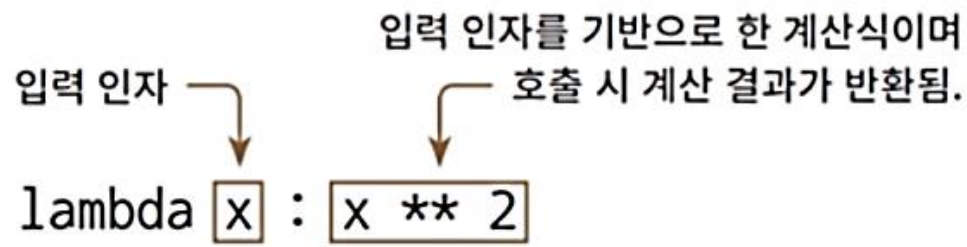
```
def get_square(a):
    return a**2

print('3의 제곱은:',get_square(3))
```

파이썬 lambda식

```
lambda_square = lambda x : x ** 2

print('3의 제곱은:',lambda_square(3))
```




```
titanic_df['Name_len']= titanic_df['Name'].apply(lambda x : len(x))
```

평가 – 성능 평가지표 (Evaluation Metric)

Accuracy(정확도)

오차행렬(Confusion Matrix)

정밀도(Precision)

재현율(Recall)

F1 Score

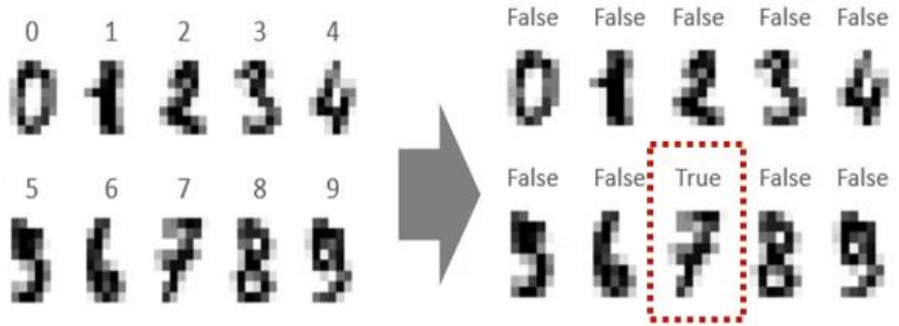
ROC AUC

정확도의 문제점

타이타닉 생존자 예측에서 여성은 모두 생존으로 판별

If Sex = '여성'
생존

MNIST 데이터셋을 multi classification에서 binary classification 으로 변경



$$\text{정확도(Accuracy)} = \frac{\text{예측 결과가 동일한 데이터 건수}}{\text{전체 예측 데이터 건수}}$$

오차행렬(Confusion Metrix)

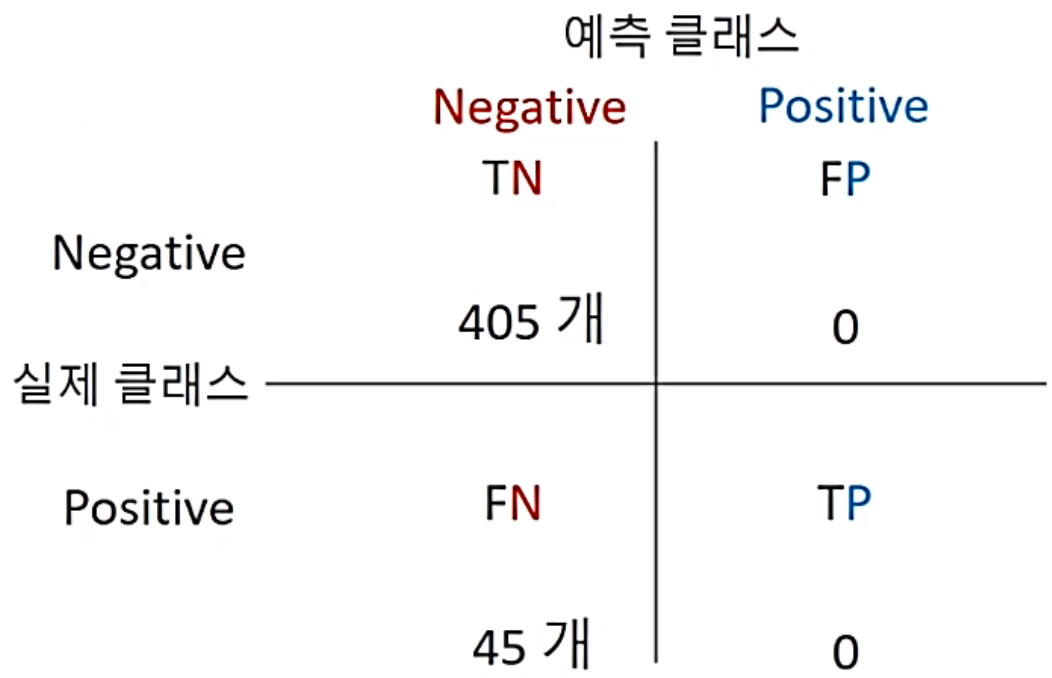
실제 클래스 (Actual Class) \ 예측 클래스(Predicted Class)	Negative(0)	Positive (1)
Negative(0)	TN (True Negative)	FP (False Positive)
Positive(1)	FN (False Negative)	TP (True Positive)

실제 클래스 \ 예측 클래스	Negative	Positive
Negative	TN 예측 : Negative (7 이 아닌 Digit) 405 개 실제 : Negative (7 이 아닌 Digit)	FP 예측 : Positive (Digit 7) 0 실제: Negative (7 이 아닌 Digit)
Positive	FN 예측 : Negative (7 이 아닌 Digit) 45 개 실제 : Positive (Digit 7)	TP 예측: Positive (Digit 7) 0 실제 : Positive (Digit 7)

```
confusion_matrix(y_test, pred)
```

정밀도(Precision)와 재현율(Recall)

- **정밀도 = TP / (FP + TP)**
- **재현율 = TP / (FN + TP)**

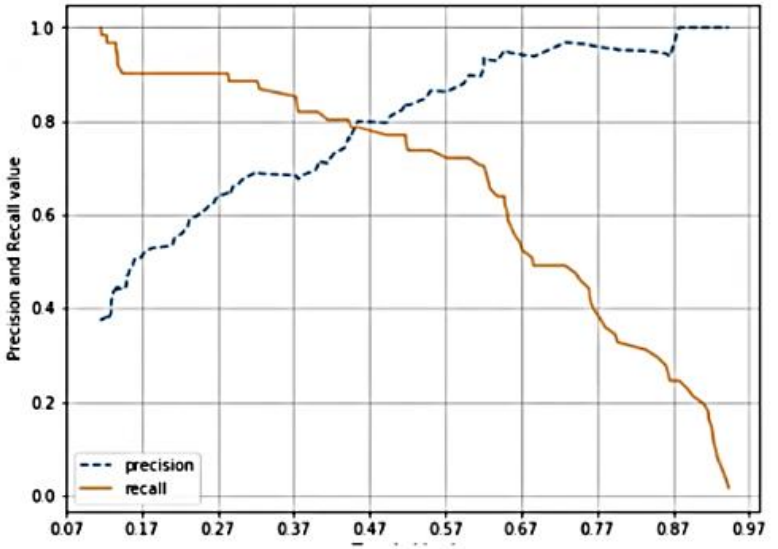



```
from sklearn.metrics import accuracy_score, precision_score , recall_score

precision_score(y_test, pred)
recall_score(y_test, pred))
```

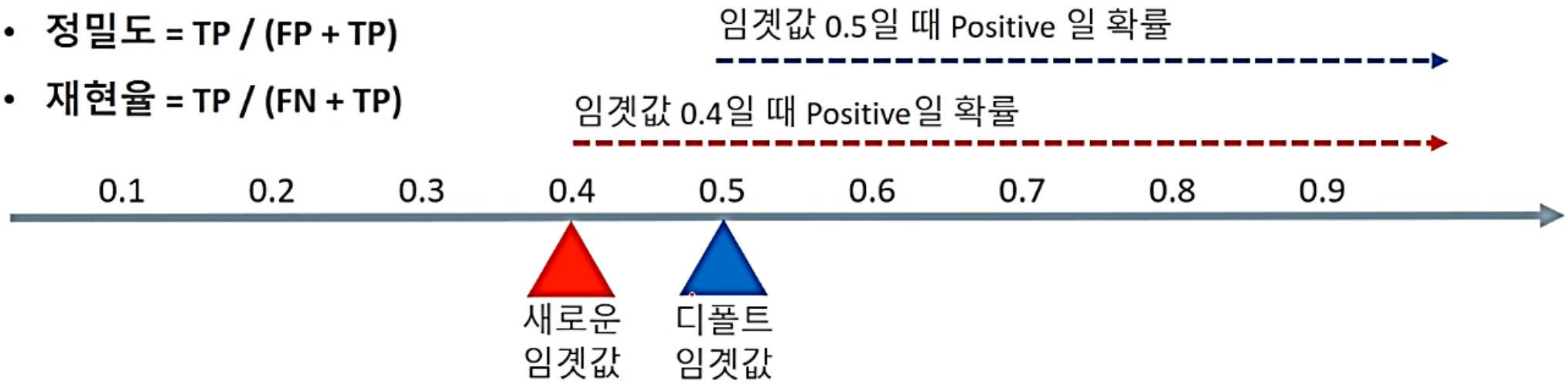
정밀도(Precision)와 재현율(Recall) 의 Trade Off

- 분류하려는 업무의 특성상 정밀도 또는 재현율이 특별히 강조돼야 할 경우 분류의 결정 임곗값(Threshold)을 조정해 정밀도 또는 재현율의 수치를 높일 수 있습니다.
- 하지만 정밀도와 재현율은 상호 보완적인 평가 지표이기 때문에 어느 한쪽을 강제로 높이면 다른 하나의 수치는 떨어지기 쉽습니다. 이를 정밀도/재현율의 트레이드오프(Trade-off)라고 부릅니다.



정밀도(Precision)와 재현율(Recall) 의 Trade Off

- **정밀도 = TP / (FP + TP)**
- **재현율 = TP / (FN + TP)**



분류 결정 임곗값이 낮아질 수록 Positive로 예측할 확률이 높아짐. 재현율 증가

- 사이킷런 Estimator 객체의 predict_proba( ) 메소드는 분류 결정 예측 확률을 반환합니다.
- 이를 이용하면 임의로 분류 결정 임곗값을 조정하면서 예측 확률을 변경할 수 있습니다.

F1 스코어

F1 스코어(Score)는 정밀도와 재현율을 결합한 지표입니다. F1 스코어는 정밀도와 재현율이 어느 한쪽으로 치우치지 않는 수치를 나타낼 때 상대적으로 높은 값을 가집니다. F1 스코어의 공식은 다음과 같습니다

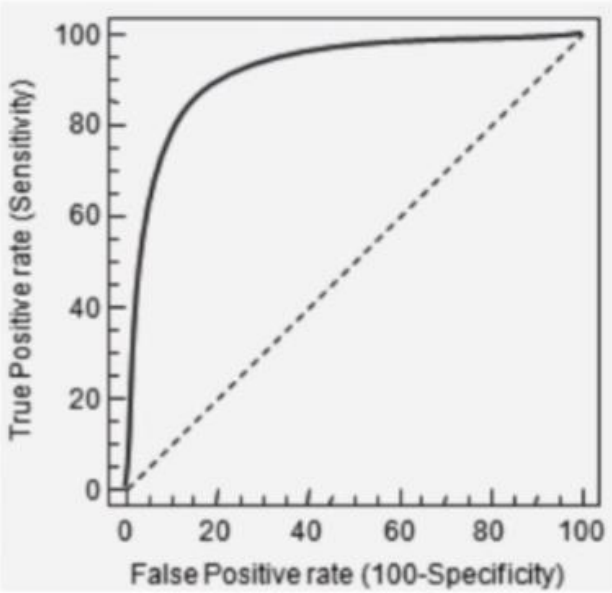
$$F1 = \frac{2}{\frac{1}{recall} + \frac{1}{precision}} = 2 * \frac{precision * recall}{precision + recall}$$

```
from sklearn.metrics import f1_score

f1 = f1_score(y_test , pred)
```

ROC 곡선

FPR의 변화에 따른 TPR의 변화 곡선



- **TPR**은 True Positive Rate의 약자이며, 이는 재현율을 나타냅니다. 따라서 **TPR은 TP / (FN +TP)**입니다. TPR, 즉 재현율은 민감도로도 불립니다.

- **FPR**은 실제 Negative(음성)을 잘못 예측한 비율을 나타냅니다. 즉 실제는 Negative인데 Positive 또는 Negative로 예측한 것 중 Positive로 잘못 예측한 비율입니다. **FPR = FP / (FP + TN)** 입니다.