

# 다중 분류의 평가지표

## 1.1.1 이진 분류의 평가지표

## 1.1.2 임계값과 평가지표

## 1.1.3 평가지표 - ROC 커브, AUC

## 1.1.4 다중 분류의 평가지표

### 학습 내용

- 다중 분류에서의 평가는 어떻게 할 수 있을까? 알아본다.

### 목차

- [01. 데이터 준비 및 라이브러리 импорт](#)
- [02. 모델 구축하기 - 로지스틱 회귀](#)
- [03. 정밀도-재현율의 커브\(랜덤 포레스트 모델\)](#)
- [04. 평균 정밀도 확인](#)
- [05. ROC 곡선과 AUC 알아보기](#)

## 01. 데이터 준비 및 라이브러리 импорт

[목차로 이동하기](#)

In [23]:

```
import matplotlib
from matplotlib import font_manager, rc
import platform
import warnings
import numpy as np
warnings.filterwarnings(action='ignore')
```

In [24]:

```
### 한글
path = "C:/Windows/Fonts/malgun.ttf"
if platform.system() == "Windows":
    font_name = font_manager.FontProperties(fname=path).get_name()
    rc('font', family=font_name)
elif platform.system() == "Darwin":
    rc('font', family='AppleGothic')
else:
    print("Unknown System")

matplotlib.rcParams['axes.unicode_minus'] = False
```

## 데이터 준비

- 데이터 셋 : 손글씨 데이터 셋
- 입력은 이미지의 픽셀 정보
- 출력은 target의 결과 : 0~9

In [25]:

```
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
```

In [26]:

```
digits = load_digits()
```

In [27]:

```
X = digits.data
y = digits.target

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    random_state=0)
```

In [28]:

```
unique, counts = np.unique(y_test, return_counts=True)

print( np.asarray((unique, counts)).T )
```

```
[[ 0 37]
 [ 1 43]
 [ 2 44]
 [ 3 45]
 [ 4 38]
 [ 5 48]
 [ 6 52]
 [ 7 48]
 [ 8 48]
 [ 9 47]]
```

## 02 모델 구축하기 - 로지스틱 회귀

[목차로 이동하기](#)

- Parameter - solver
  - 'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga' , default='lbfgs'
  - liblinear : 작은 데이터셋의 경우 좋은 선택
  - 'sag', 'saga' : 큰 데이터셋의 경우 더 빠름.
- Parameter - multi\_class
  - 'auto', 'ovr', 'multinomial', default = 'auto'
  - 'ovr' : 각 레이블에 이항 문제가 적합, 다항의 경우 전체 확률 분포에 걸쳐 다항 손실이 최소화

- 'auto' : solver='liblinear' 때는 'ovr' 선택됨. 그렇지않으면 multinomial을 선택
- 0.22버전에서 기본값이 0.22에서 'ovr'에서 'auto'로 변경

In [29]:

```
lr = LogisticRegression(solver='liblinear',
                        multi_class='ovr').fit(X_train, y_train)
pred = lr.predict(X_test)

print("정확도 : {:.3f}".format(accuracy_score(y_test, pred)))
print("오차 행렬 :\n", confusion_matrix(y_test, pred))
```

정확도 : 0.953

오차 행렬 :

```
[[37  0  0  0  0  0  0  0  0  0]
 [ 0 39  0  0  0  0  2  0  2  0]
 [ 0  0 41  3  0  0  0  0  0  0]
 [ 0  0  1 43  0  0  0  0  0  1]
 [ 0  0  0  0 38  0  0  0  0  0]
 [ 0  1  0  0  0 47  0  0  0  0]
 [ 0  0  0  0  0  0 52  0  0  0]
 [ 0  1  0  1  1  0  0 45  0  0]
 [ 0  3  1  0  0  0  0  0 43  1]
 [ 0  0  0  1  0  1  0  0  1 44]]
```

모델의 정확도는 95.3%로 꽤 좋은 성능 좋다.

각 행은 실제 정답 레이블에 해당하며, 열은 예측 레이블에 해당

## 오차 행렬 그래프로 표시

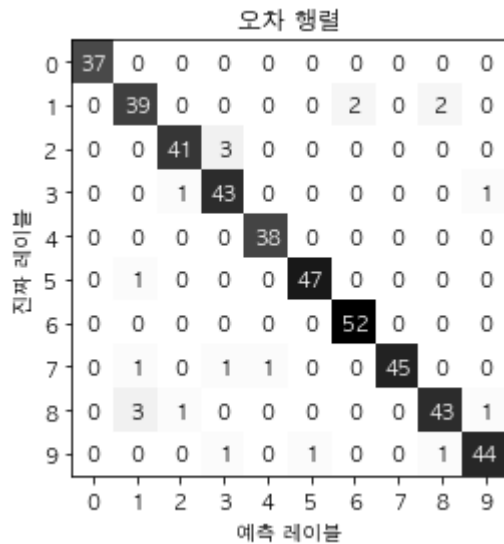
In [30]:

```
import mglearn
import matplotlib.pyplot as plt
```

In [31]:

```
scores_image = mglearn.tools.heatmap(
    confusion_matrix(y_test, pred),
    xlabel='예측 레이블', ylabel='진짜 레이블',
    xticklabels=digits.target_names,
    yticklabels=digits.target_names, cmap=plt.cm.gray_r, fmt="%d")

plt.title("오차 행렬")
plt.gca().invert_yaxis()
```



- 첫번째 클래스는 숫자 0인 샘플이 총 37개, 모두 클래스를 0으로 분류. 클래스 0에는 거짓 음성(FN)이 없음.
- 첫번째 열의 다른 항목들이 모두 0이므로 거짓 양성(FP)가 없음.

### 03. 정밀도, 재현율, f1-score 확인

[목차로 이동하기](#)

- classification\_report 함수를 사용한 정밀도, 재현율, f1-score 점수 확인

In [32]:

```
from sklearn.metrics import classification_report
```

In [33]:

```
print(classification_report(y_test, pred))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	37
1	0.89	0.91	0.90	43
2	0.95	0.93	0.94	44
3	0.90	0.96	0.92	45
4	0.97	1.00	0.99	38
5	0.98	0.98	0.98	48
6	0.96	1.00	0.98	52
7	1.00	0.94	0.97	48
8	0.93	0.90	0.91	48
9	0.96	0.94	0.95	47
accuracy			0.95	450
macro avg	0.95	0.95	0.95	450
weighted avg	0.95	0.95	0.95	450

In [37]:

```
## 혼동 행렬
from IPython.display import display, Image
#display(Image(filename='img/model_validation01.png'))
mglearn.plots.plot_binary_confusion_matrix()
```

		predicted negative	predicted positive
negative class	TN		FP
positive class	FN		TP

$$\text{정밀도(precision)} = \frac{\text{잘 예측(TP)}}{\text{예측을 양성으로 한 것 전체(TP+FP)}}$$

$$\text{민감도(recall, 재현율)} = \frac{\text{잘 예측(TP)}}{\text{실제 값이 양성인것 전체(TP+FN)}}$$

- 0에는 오차가 없으므로 정밀도와 재현율은 모두 1로 완벽
- 정밀도
  - 클래스 7은 다른 클래스가 7로 잘못 분류한 것이 없어서 정밀도가 1이다. 45개 중에 45개 맞혔음.
    - (45)/45
  - 클래스 8 => 43/(43+3) = 0.934(반올림:0.93)

- 클래스 3 =>  $43/(43+5) = 0.895$ (반올림:0.90)
- 재현율
  - 클래스 2 =>  $41/(41+3) = 0.931$ (반올림:0.93)
  - 클래스 6은 거짓 음성(FN)이 없어서 재현율이 1이다. => 52/52
  - 클래스 8은 =>  $43/(43+5) = 0.895$
- 이 모델은 1,3,8을 분류하는데 어려움을 겪고 있다. (F1-score: 0.90, 0.92, 0.91)

**다중 클래스용 f1-score점수는 한 클래스를 양성 클래스로 두고, 나머지 클래스를 음성 클래스로 간주하여 클래스마다 f1-score를 계산**

다중 분류에서 불균형 데이터셋을 위해 가장 널리 사용하는 평가 지표는 f1-score점수의 다중 분류 버전.

In [38]:

```
from sklearn.metrics import f1_score
```

In [39]:

```
print("micro 평균 f1점수 : {:.3f}".format(f1_score(y_test,
                                                    pred, average='micro')))
```

```
print("macro 평균 f1점수 : {:.3f}".format(f1_score(y_test,
                                                    pred, average='macro')))
```

```
micro 평균 f1점수 : 0.953
macro 평균 f1점수 : 0.954
```

- 각 샘플을 똑같이 간주한다면 "micro"평균 f1-score점수를 권장
- 각 클래스를 동일한 비중을 고려한다면 "macro" 평균 f1-score점수 권장

교육용으로 작성된 것으로 배포 및 복제시에 사전 허가가 필요합니다.

Copyright 2022 LIM Co. all rights reserved.

In [ ]: