평가 지표 및 측정

1.1.1 최종 목표를 기억하라

1.1.2 이진 분류의 평가지표

1.1.3 다중 분류의 평가지표

1.1.4 회귀의 평가 지표

학습 내용

- 이진 분류의 평가 지표에 대해 알아본다.
- 불균형 데이터 셋일때의 정확도.
- 정밀도, 민감도, 특이도, FPRate, F-score에 대해 알아본다.
- 함수를 활용하여 각각의 모델별 정밀도, 민감도, F-score를 확인해 본다.

In [32]:
```
from IPython.display import display, Image
```

In [33]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
```

01 데이터 셋 준비

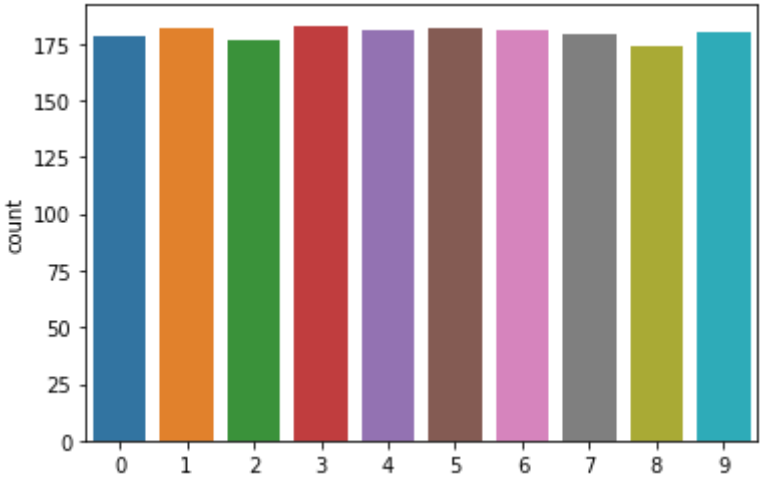
데이터 셋

- 손글씨 데이터
- data : 1797장, 64개의 pixel 데이터
  - images : 1797, 8, 8
- target : 0~9까지의 손글씨 값

In [34]:
```
from sklearn.datasets import load_digits

digits = load_digits()
print(digits.data.shape)
print(digits.keys(), digits.target)
print(np.unique( digits.target ) )
sns.countplot(digits.target)
```

```
(1797, 64)
dict_keys(['data', 'target', 'frame', 'feature_names', 'target_names', 'images', 'DESC
R']) [0 1 2 ... 8 9 8]
[0 1 2 3 4 5 6 7 8 9]
C:WUsersWfrontWanaconda3WlibWsite-packagesWseabornW_decorators.py:36: FutureWarning: P
ass the following variable as a keyword arg: x. From version 0.12, the only valid posi
tional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```

Out[34]: <AxesSubplot:ylabel='count'>



타깃이 9:1의 비율을 갖도록 하기

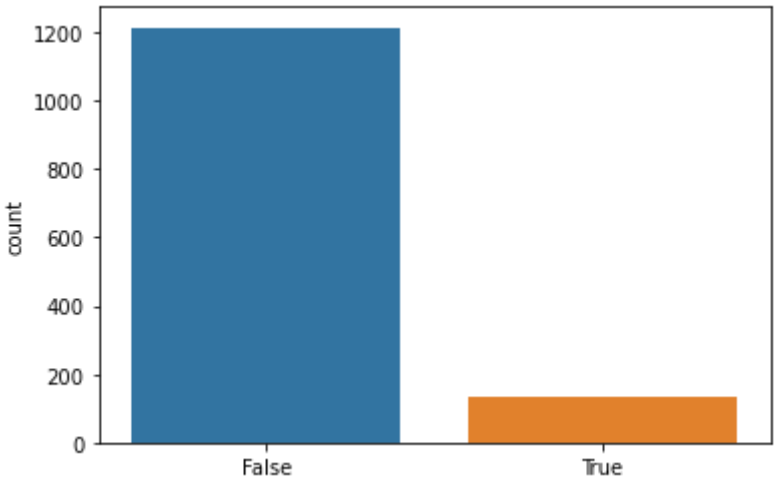
- 9이면 True
- 9가 아니면 False

In [35]:
```
X = digits.data
y = digits.target == 9

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

In [36]:
```
sns.countplot(y_train)
```

```
C:WUsersWfrontWanaconda3WlibWsite-packagesWseabornW_decorators.py:36: FutureWarning: P
ass the following variable as a keyword arg: x. From version 0.12, the only valid posi
tional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```

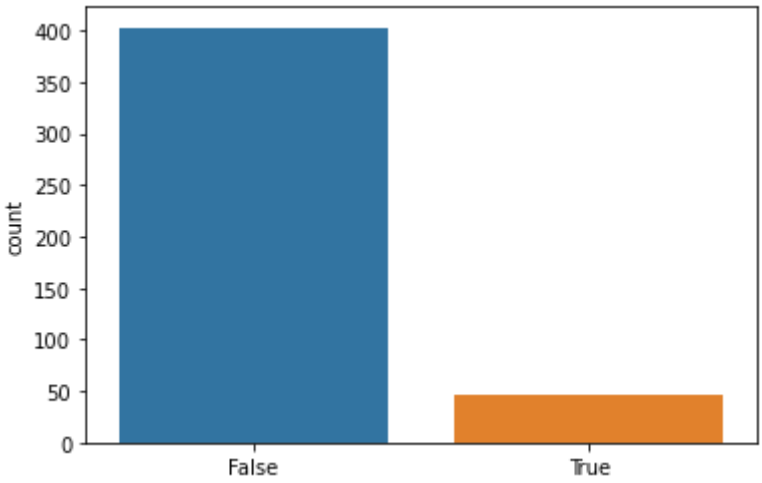
Out[36]: <AxesSubplot:ylabel='count'>



In [37]:
```
sns.countplot(y_test)
```

```
C:WUsersWfrontWanaconda3WlibWsite-packagesWseabornW_decorators.py:36: FutureWarning: P
ass the following variable as a keyword arg: x. From version 0.12, the only valid posi
tional argument will be `data`, and passing other arguments without an explicit keywor
d will result in an error or misinterpretation.
  warnings.warn(
```

Out[37]: <AxesSubplot:ylabel='count'>



02 여러가지 모델을 활용한 정확도(accuracy) 확인

02-01 기본 모델 DummyClassifier를 사용한 정확도(accuracy) 계산

- 간단한 룰을 사용하여 예측을 수행한다.
- 실제 문제에서는 사용하지 않으며, 간단한 베이스라인 모델로서 사용된다.
- DummyClassifier(strategy='most_frequent') : 학습용 세트에서 가장 빈번한 라벨을 예측한다.
  - stratified : 클래스 분포를 존중하여 예측을 생성
  - uniform : 무작위로 균일하게 예측을 생성 , 기타 : prior, constant

- 아래 모델은 가장 많은 레이블을 가진 False만 예측하게 된다.

In [38]:
```
from sklearn.dummy import DummyClassifier
dummy_majority = DummyClassifier(strategy='most_frequent').fit(X_train, y_train)
pred_most_frequent = dummy_majority.predict(X_test)
print("예측된 레이블의 고유값: {}".format(np.unique(pred_most_frequent)))
print("테스트 점수: {:.2f}".format(dummy_majority.score(X_test, y_test)))
```

```
예측된 레이블의 고유값: [False]
테스트 점수: 0.90
```

02-02 DummyClassifier를 이용한 예측

- 매개변수 없을 때의 기본 동작
  - stratified : 클래스 분포를 가만하여 예측

- 클래스의 9:1 분포를 가만하여 예측

In [39]:
```
dummy = DummyClassifier().fit(X_train, y_train)
pred_dummy = dummy.predict(X_test)
print("예측된 레이블의 고유값: {}".format(np.unique(pred_dummy)))
print("dummy 점수: {:.2f}".format(dummy.score(X_test, y_test)))
```

```
예측된 레이블의 고유값: [False  True]
dummy 점수: 0.82
C:₩Users₩front₩anaconda3₩lib₩site-packages₩sklearn₩dummy.py:131: FutureWarning: The de
fault value of strategy will change from stratified to prior in 0.24.
  warnings.warn("The default value of strategy will change from "
```

02-03 실제 분류기를 사용해보기 - DecisionTreeClassifier

In [40]:
```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(max_depth=2).fit(X_train, y_train)
```

```
pred_tree = tree.predict(X_test)
print("테스트 점수: {:.2f}".format(tree.score(X_test, y_test)))
```

```
테스트 점수: 0.92
```

- 실제 분류기와 기본적인 모델 dummy 분류기와 성능차이가 거의 없다.

02-04 LogisticRegression 모델 확인

In [41]:
```
from sklearn.linear_model import LogisticRegression

logreg = LogisticRegression(C=0.1).fit(X_train, y_train)
pred_logreg = logreg.predict(X_test)
print("logreg 점수: {:.2f}".format(logreg.score(X_test, y_test)))
```

```
logreg 점수: 0.98
C:₩Users₩front₩anaconda3₩lib₩site-packages₩sklearn₩linear_model₩_logistic.py:762: Conv
ergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

하나만 예측하는 기본 모델도 90% 이상의 정확도를 갖는다.

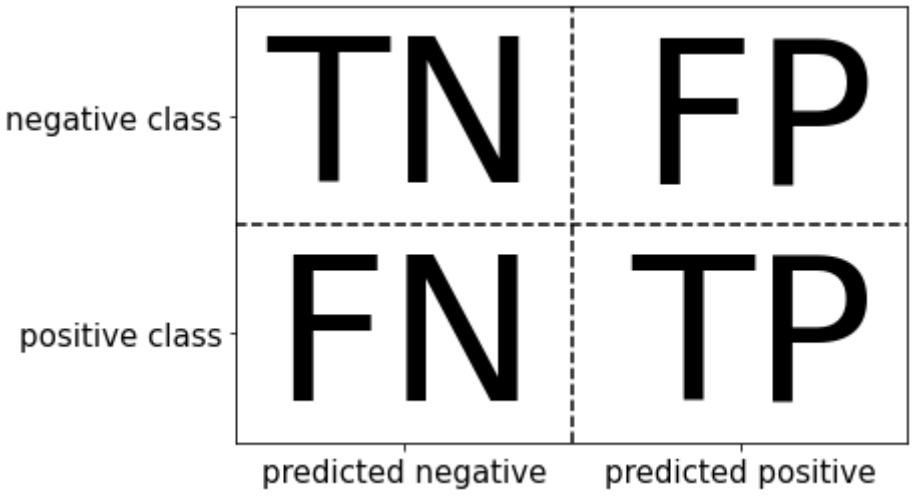
- 정확도는 때로는 평가지표로 사용하기에 부족한 부분이 있다.

정확도 대신에 사용할 지표가 무엇이 있을까?

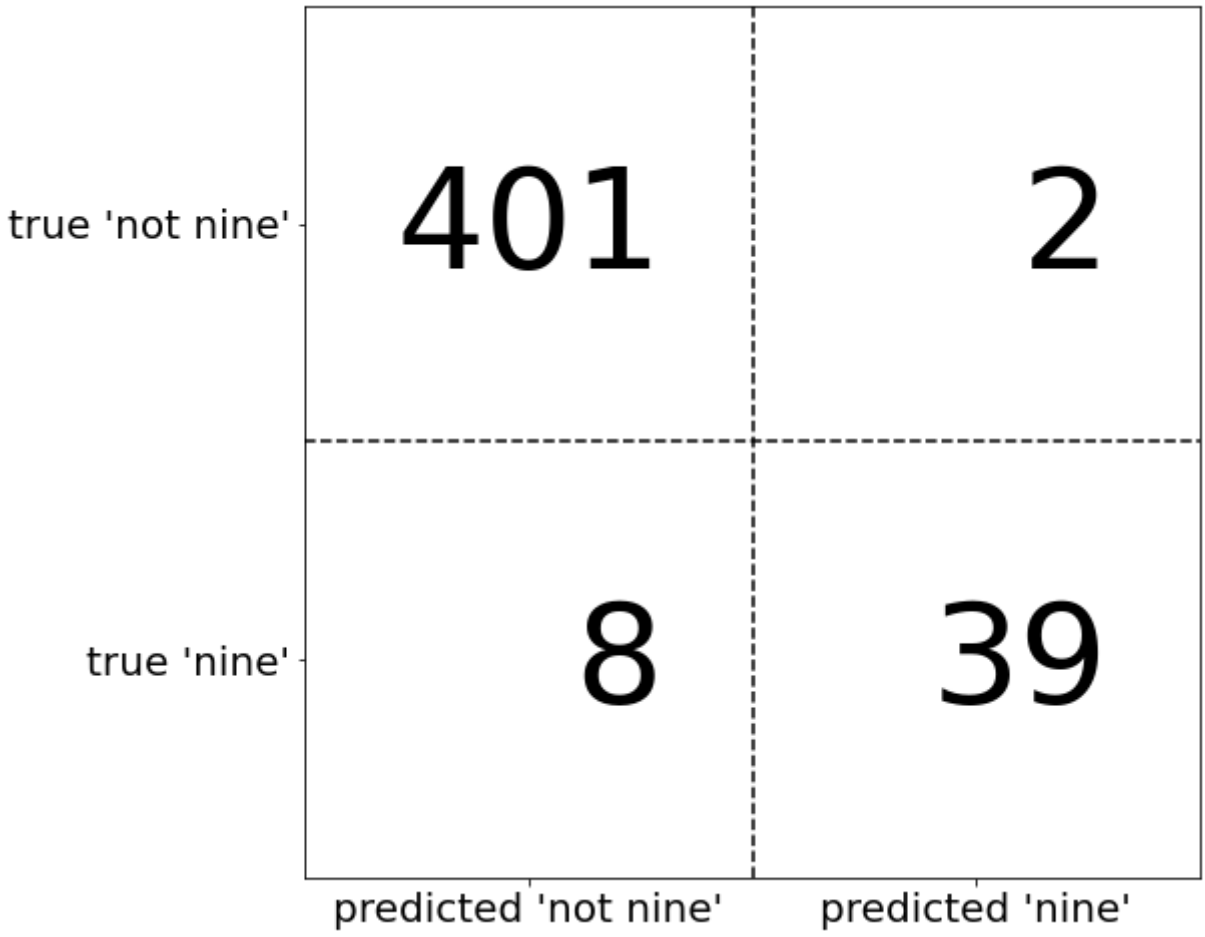
03 오차행렬(confusion matrix)을 이용하기

In [42]:
```
import mglearn
```

In [43]:
```
mglearn.plots.plot_binary_confusion_matrix()
```



In [44]:
```
mglearn.plots.plot_confusion_matrix_illustration()
```



In [45]:
```
from sklearn.metrics import confusion_matrix

confusion = confusion_matrix(y_test, pred_logreg)
print("오차 행렬:₩n{}".format(confusion))
```

```
오차 행렬:
[[402   1]
 [  6  41]]
```

3-1 각각의 예측값에 대한 오차행렬을 확인해보기

In [46]:
```
print("빈도 기반 더미 모델:")
print(confusion_matrix(y_test, pred_most_frequent))

print("₩n무작위 더미 모델:")
print(confusion_matrix(y_test, pred_dummy))

print("₩n결정 트리:")
print(confusion_matrix(y_test, pred_tree))

print("₩n로지스틱 회귀")
print(confusion_matrix(y_test, pred_logreg))
```

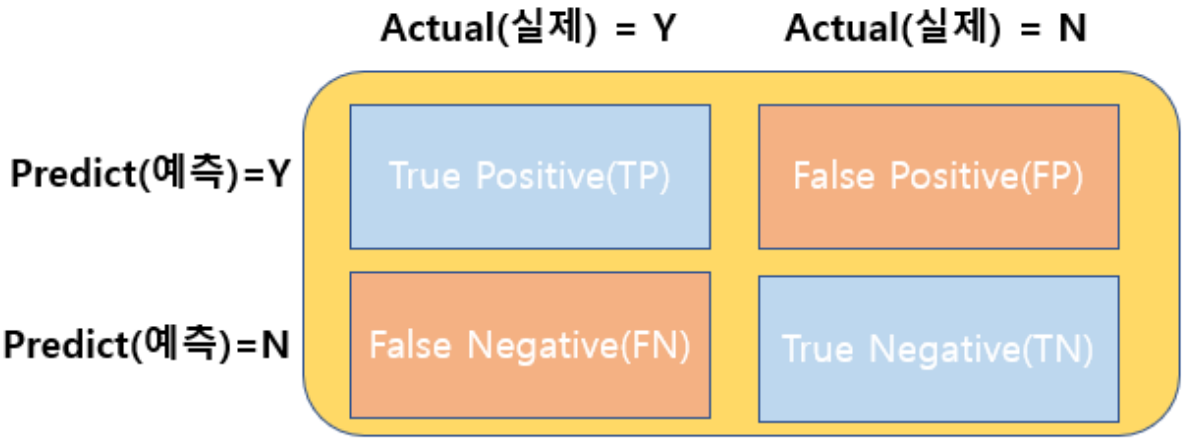
```
빈도 기반 더미 모델:
[[403   0]
 [ 47   0]]

무작위 더미 모델:
[[370  33]
 [ 41   6]]

결정 트리:
[[390  13]
 [ 24  23]]
```

```
로지스틱 회귀
[[402   1]
 [  6  41]]
```

3-2 분류의 다양한 평가지표를 살펴보기

- 정확도(accuracy) : 정확하게 예측/전체 예측수
- 정밀도
- 민감도
- 특이도
- Fprate
- F-score
- AUC

In [47]:
```
## 머신러닝 작업 flow
display(Image(filename='img/model_validation01.png'))
```



분류의 평가지표를 살펴보자.

정확도(accuracy) : 정확하게 예측/전체 예측수

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

정밀도(precision) : 예측을 양성(Positive)으로 한것 전체-TP+FP중에 맞는 것(T*)

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

- 언제 사용하는가? : 거짓 양성(FP)의 수를 줄일 때 사용
  - 약이 효과 있다고 예측(positive), 하지만 예측 결과가 틀림
  - 암이 아닌데 암이라고 예측

민감도(sensitivity), 재현율(recall, TPRate)

- 실제 데이터의 양성 데이터(TP + FN)중에 얼마나 많은 샘플을 양성으로 잘 분류했나?(TP)
- TP/(TP + FN)

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

- 재현율이 높아지면 FP는 상대적으로 낮아짐.
- 언제 사용? FN(음성 예측. 잘못 예측함.)을 줄일 때, 성능 지표로 사용합니다.
- (암인데 음성(아니라고-Negative)예측하여, 실수. 엄청난 실수 **실제 암인데(Positive)인데,
  - 이를 병원에서 암이 아니다(Negative)**로 예측하면 얼마나 큰일인가?

- 다른 말로 **민감도(sensitivity), 적중률(hit rate), 진짜 양성 비율(TPR)**이라고 합니다.
- 따라서 병원의 암 예측 같은 경우는 FN를 최소화시켜 재현율을 줄이면, 상대적으로 정밀도를 최대화된다.

특이도

- 실제 데이터의 음성 데이터(FP + TN)중에 얼마나 많은 샘플을 잘 분류했나?(TN)
- TN/(FP + TN)

$$\text{특이도} = \frac{\text{잘 예측(TN)}}{\text{실제 값이 음성인것 전체(FP + TN)}}$$

FPRate

- 실제 데이터의 음성 데이터(FP + TN)중에 잘 분류하지 못한 것?(FP)
- FP/(FP + TN)

$$\text{FPRate} = \frac{\text{틀린 예측(FP)}}{\text{실제 값이 음성인것 전체(FP + TN)}}$$

다양한 분류 측정 방법

- https://en.wikipedia.org/wiki/Sensitivity_and_specificity
- 이진 분류에서는 정밀도와 재현율을 가장 많이 사용.
  - 분야마다 다른 지표를 사용할 수 있다.

F-score

- 정밀도와 민감도(recall,재현율)을 하나만 가지고 측정이 안된다. 정밀도(precision)와 재현율(recall)의 조화 평균인 f-점수 또는 f-측정은 이 둘을 하나로 요약을 해 줍니다.

F = 2 x (정밀도*재현율)/(정밀도 + 재현율)

- 다른 말로 F1-score라고 한다.

3-3 f1-score를 확인해보기

각각의 모델 예측값을 f1-score로 예측

In [48]:
```
from sklearn.metrics import f1_score

# Dummy분류
print("무작위 더미 모델의 f1 score: {:.2f}".format(f1_score(y_test, pred_dummy)))

# 의사결정트리
print("트리 모델의 f1 score: {:.2f}".format(f1_score(y_test, pred_tree)))

# 로지스틱
print("로지스틱 회귀 모델의 f1 score: {:.2f}".format(
    f1_score(y_test, pred_logreg)))
```

```
무작위 더미 모델의 f1 score: 0.14
트리 모델의 f1 score: 0.55
로지스틱 회귀 모델의 f1 score: 0.92
```

f1-score를 요약해서 보여주기

- classification_report() : 정밀도, 재현율, f1-score을 모두 한번에 계산
- support는 단순히 샘플의 수

In [49]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, pred_most_frequent))
```

```
              precision    recall  f1-score   support

       False       0.90      1.00      0.94       403
        True       0.00      0.00      0.00        47

    accuracy                           0.90       450
   macro avg       0.45      0.50      0.47       450
weighted avg       0.80      0.90      0.85       450

C:₩Users₩front₩anaconda3₩lib₩site-packages₩sklearn₩metrics₩_classification.py:1221: Un
definedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in la
bels with no predicted samples. Use `zero_division` parameter to control this behavio
r.
  _warn_prf(average, modifier, msg_start, len(result))
```

dummyClassifier 모델

In [50]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, pred_dummy,
                           target_names=["not 9", "is 9"]))
```

```
              precision    recall  f1-score   support

      not 9       0.90      0.92      0.91       403
       is 9       0.15      0.13      0.14        47

    accuracy                           0.84       450
   macro avg       0.53      0.52      0.52       450
weighted avg       0.82      0.84      0.83       450
```

로지스틱 회귀

In [51]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,
                           pred_logreg,
                           target_names=["not 9", "is 9"]))
```

```
              precision    recall  f1-score   support

      not 9       0.99      1.00      0.99       403
       is 9       0.98      0.87      0.92        47

    accuracy                           0.98       450
   macro avg       0.98      0.93      0.96       450
weighted avg       0.98      0.98      0.98       450
```