03. Classification

MNIST

미국 고교생 및 인구 조사국 직원들의 숫자 손글씨(70,000 샘플)

손글씨 digit image(28x28 grayscale pixel array), 숫자 레이블을 포함

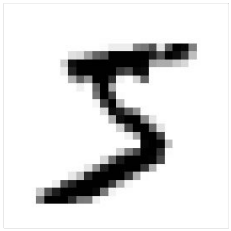
새로운 분류 알고리즘 등장시 성능의 척도로 활용

MNIST

```
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt

some_digit = X[0]
some_digit_image = some_digit.reshape(28, 28)
plt.imshow(some_digit_image, cmap=mpl.cm.binary)
plt.axis("off")

save_fig("some_digit_plot")
plt.show()
```



```
y[0] == '5'
```

https://github.com/rickiepark/handson-ml/blob/master/03_classification.ipynb

Binary classification

True / False 를 구분하는 구분기

- 스팸 메일 분류기 등

본 예제에서는

- ‘이 손글씨가 5인지의 여부’ 에 대해 진행
- Train : Test = 6:1

```
y_train_5 = (y_train == 5)
y_test_5 = (y_test == 5)
```

```
X_train, X_test, y_train, y_test = X[:60000], X[60000:], y[:60000], y[60000:]
```

Train and validate a binary classification

Stochastic gradient descent classifier

- step 마다 gradient 를, 임의의 데이터 하나를 선택해 계산
- 즉 학습에 랜덤성이 존재

```
from sklearn.linear_model import SGDClassifier

sgd_clf = SGDClassifier(max_iter=1000, tol=1e-3, random_state=42)
sgd_clf.fit(X_train, y_train_5)
```

Stratified k-fold cross validation

- stratified 방식은 class 별 비율이 유지되도록 fold 를 만듬
- 주로 classification 문제에 주로 활용

```
from sklearn.model_selection import cross_val_score
skfolds = StratifiedKFold(n_splits=3, random_state=42)
cross_val_score(sgd_clf, X_train, y_train_5, cv=skfolds, scoring="accuracy")
```

Is the ‘Accuracy’ accurate enough?

0~9 의 이미지가 n개씩 총 10n개의 손글씨 데이터에 대해, 모든 손글씨에 ‘5 아님’을 출력하는 모델의 ‘Accuracy’ 는?

Is the ‘Accuracy’ accurate enough?

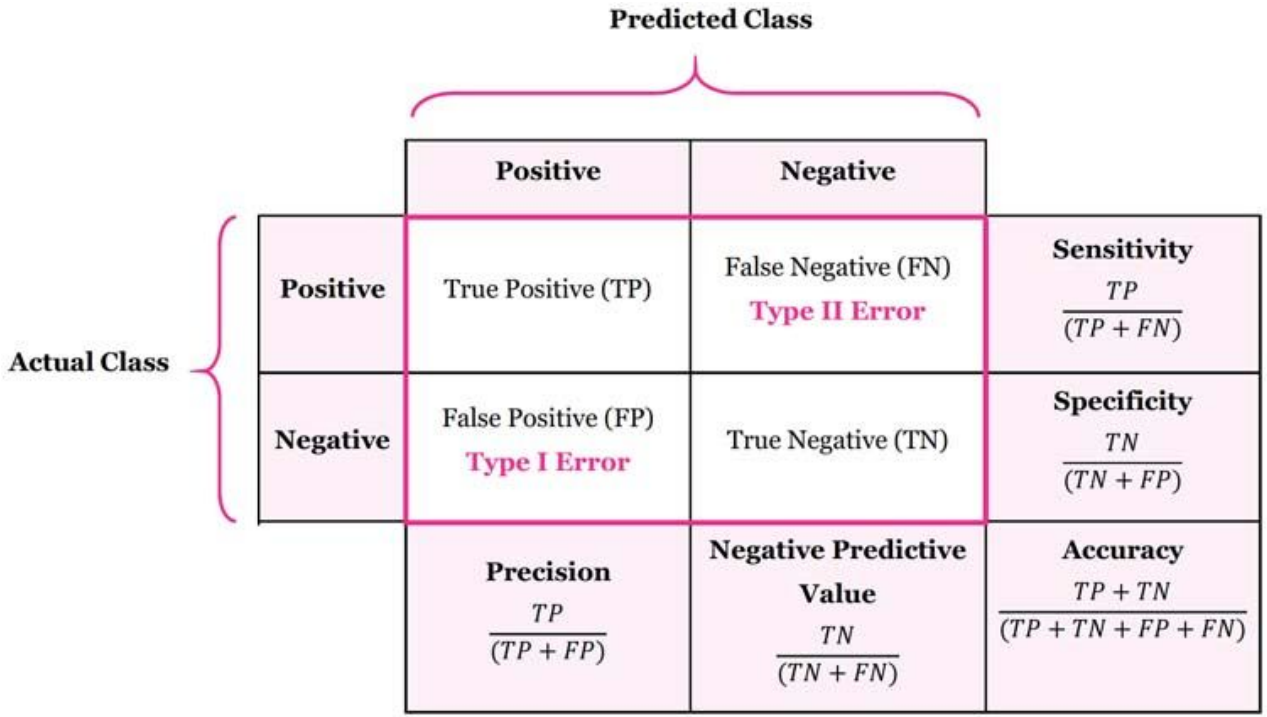
0~9 의 이미지가 n개씩 총 10n개의 손글씨 데이터에 대해, 모든 손글씨에 ‘5 아님’을 출력하는 모델의 ‘Accuracy’ 는?

- **9n / 10n = 90%**

알파벳 중 한글자의 판단이라면, 25 / 26 ~= 96 %

지구상 생물 종 중 1종류를 찾는 거라면, (8.7Mil - 1) / 8.7Mil ~= 99.99998 %

Confusion matrix

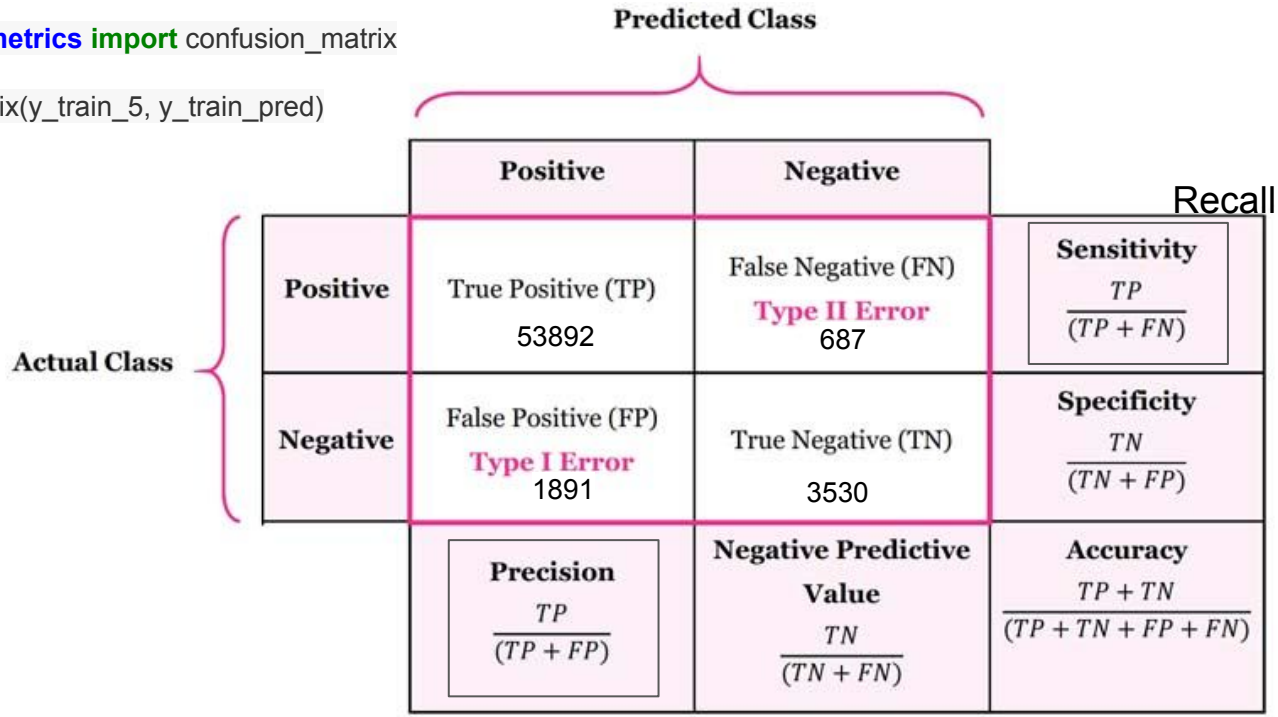


https://manisha-sirsat.blogspot.com/2019/04/confusion-matrix.html

Confusion matrix

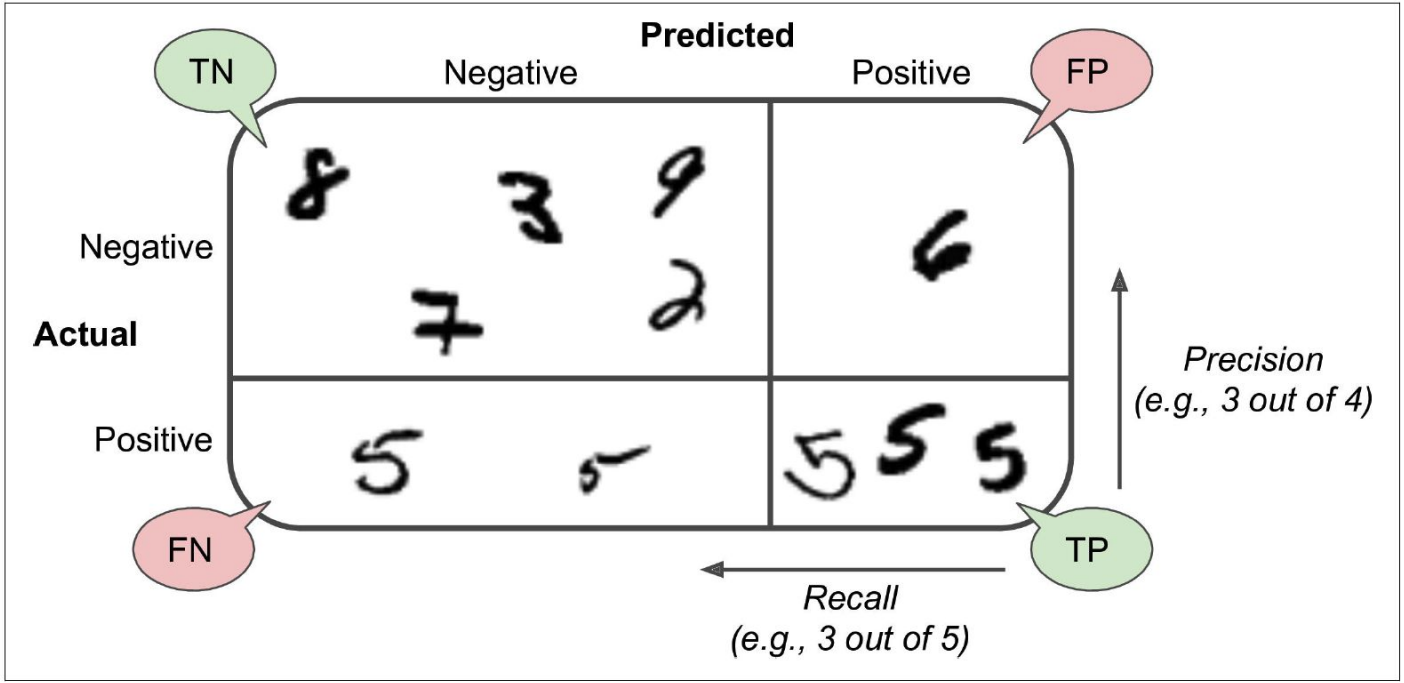
```
from sklearn.metrics import confusion_matrix

confusion_matrix(y_train_5, y_train_pred)
```



https://manisha-sirsat.blogspot.com/2019/04/confusion-matrix.html

Confusion matrix



https://dojinkimm.github.io/ml/2019/10/26/handson-ch3.html

Scores for estimate model

Precision : 전체 정답수중 정답으로 예측한 비

Recall : 실제 정답 중 정답을 맞춘 비율

F1 : Precision, Recall 의 조화평균

- Precision, Recall 이 비슷할때 높은 점수
- F1 점수가 높다고 모델이 꼭 좋은 것은 아

  - 어린이용 영상물 필터링
  - 방범 CCTV

$$precision = \frac{TP}{TP + FP}$$

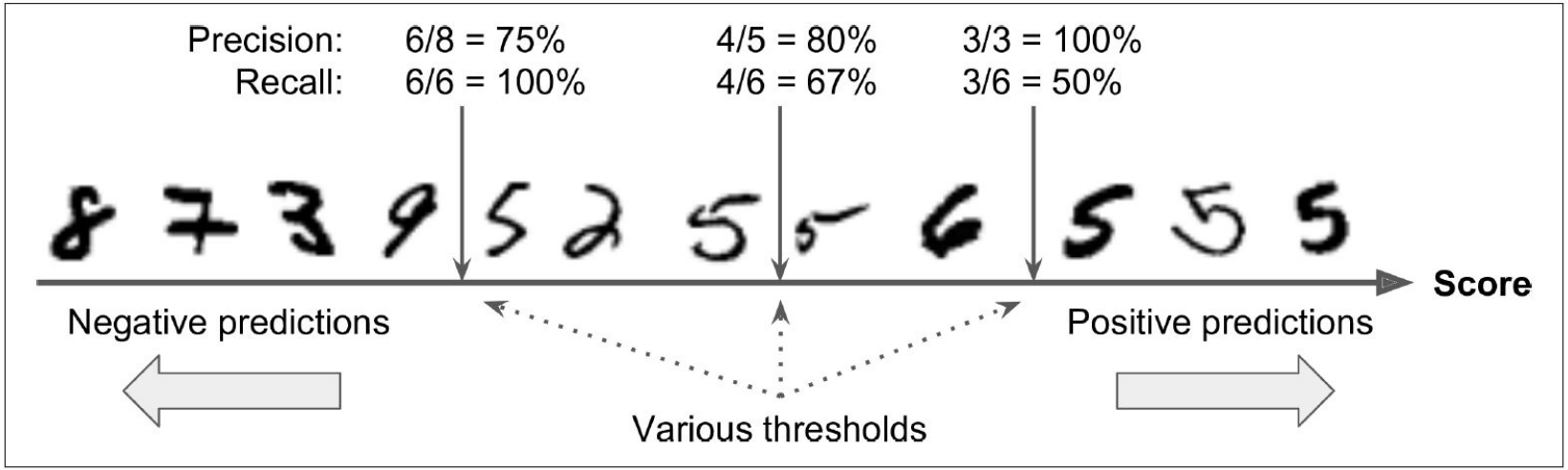
$$recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

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

https://stackoverflow.com/questions/35365007/tensorflow-precision-recall-f1-score-and-confusion-matrix

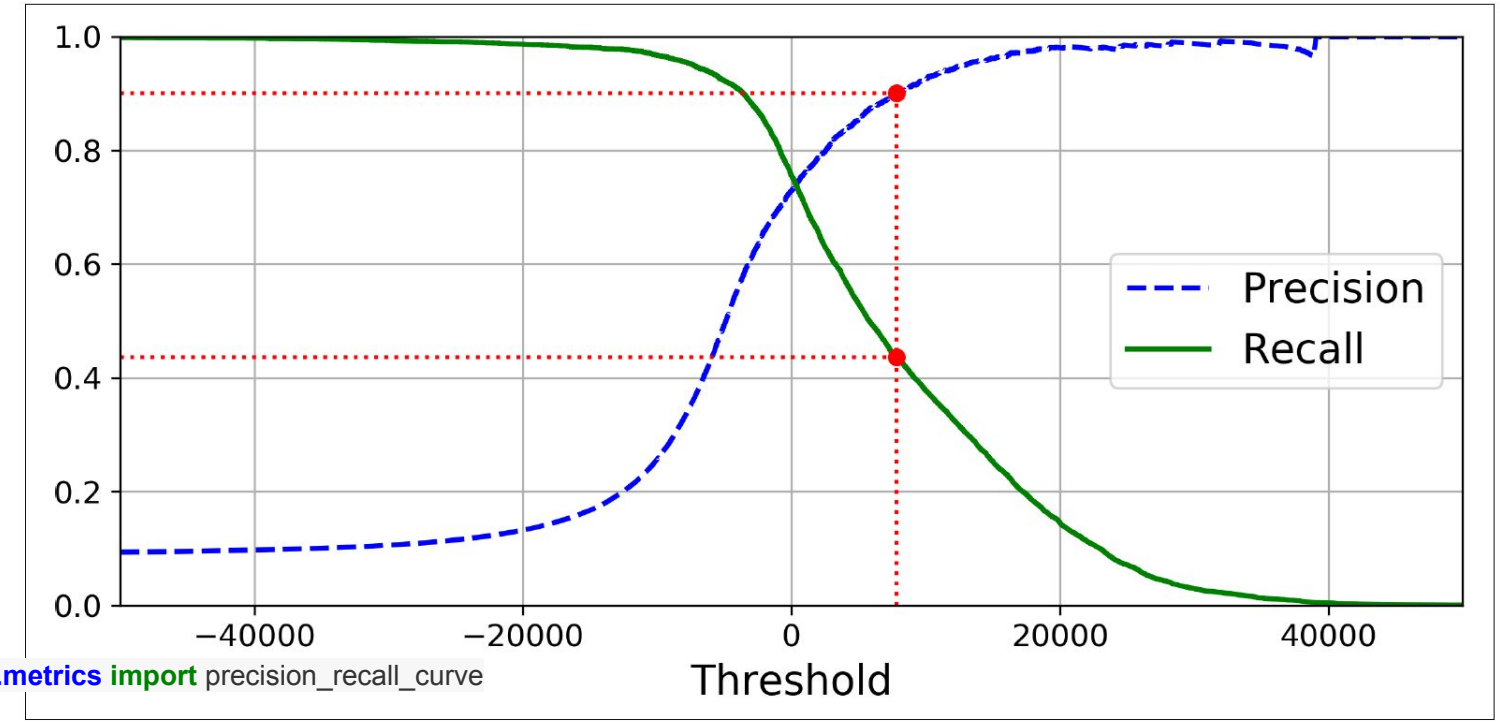
Tradeoffs : Precision vs Recall




```
y_scores = sgd_clf.decision_function([some_digit]) # 점수 추출
```

https://dojinkimm.github.io/ml/2019/10/26/handson-ch3.html

Tradeoffs : Precision vs Recall

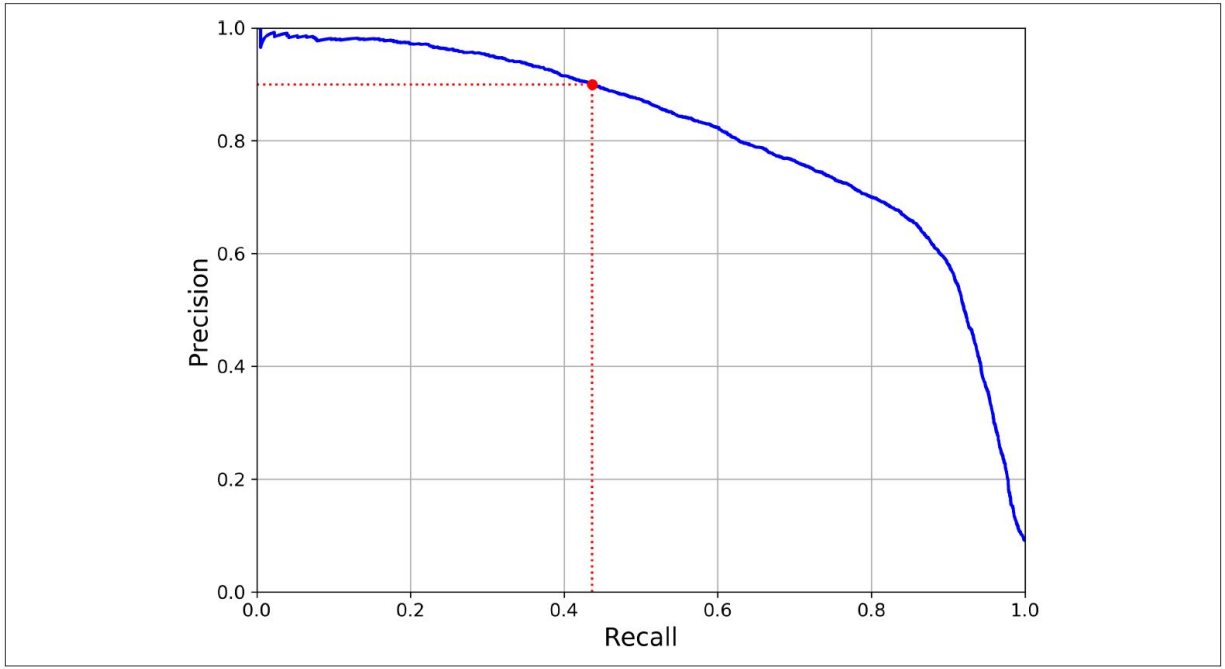



```
from sklearn.metrics import precision_recall_curve

precisions, recalls, thresholds = precision_recall_curve(y_train_5, y_scores)
```

https://dojinkimm.github.io/ml/2019/10/26/handson-ch3.html

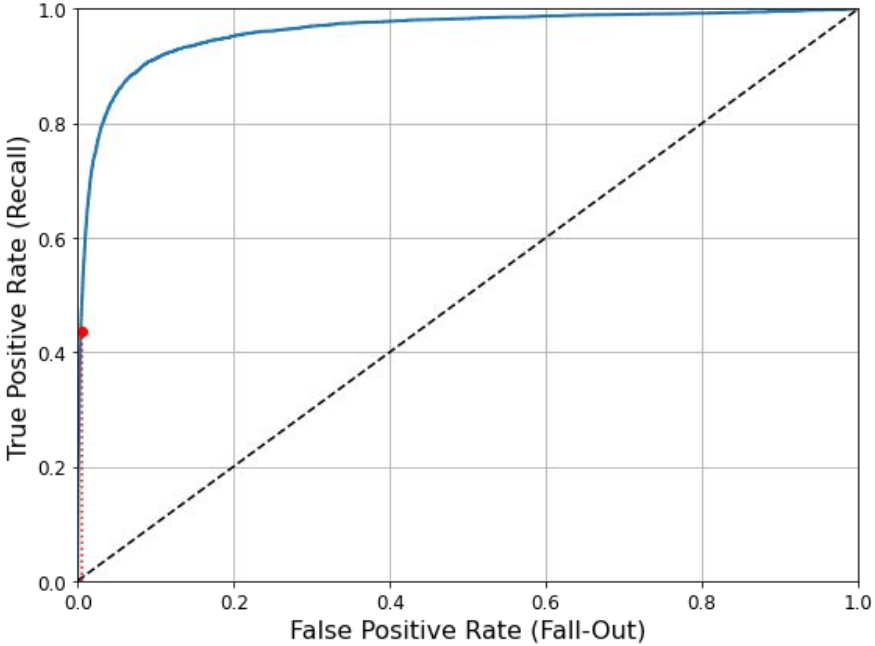
Tradeoffs : Precision vs Recall



https://dojinkimm.github.io/ml/2019/10/26/handson-ch3.html

ROC Curve

Positive 로 예측된 데이터 중, 정상 분류된 수과, 잘못 분류된 수의 비율을 나타내는 그래프



https://dojinkimm.github.io/ml/2019/10/26/handson-ch3.html

ROC Curve

기준치

- True Positive Rate(TPR)
  - **양성 사례를 양성으로 잘 예측**한 비율
  - 예) 암환자를 진찰해서 암이라고 진단
- False Positive Rate(FPR)
  - **음성 사례를 양성으로 잘못 예측**한 비율
  - 예) 암환자가 아닌데 암이라고 진단 함

좋은 지표인 TPR 은 높게 가져가고,

나쁜 지표인 FPR을 적게 가져가는 것이 목표

predicted → / real ↓	*Class_pos*	*Class_neg*
Class_pos	TP	FN
Class_neg	FP	TN

$$\text{TPR (sensitivity)} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

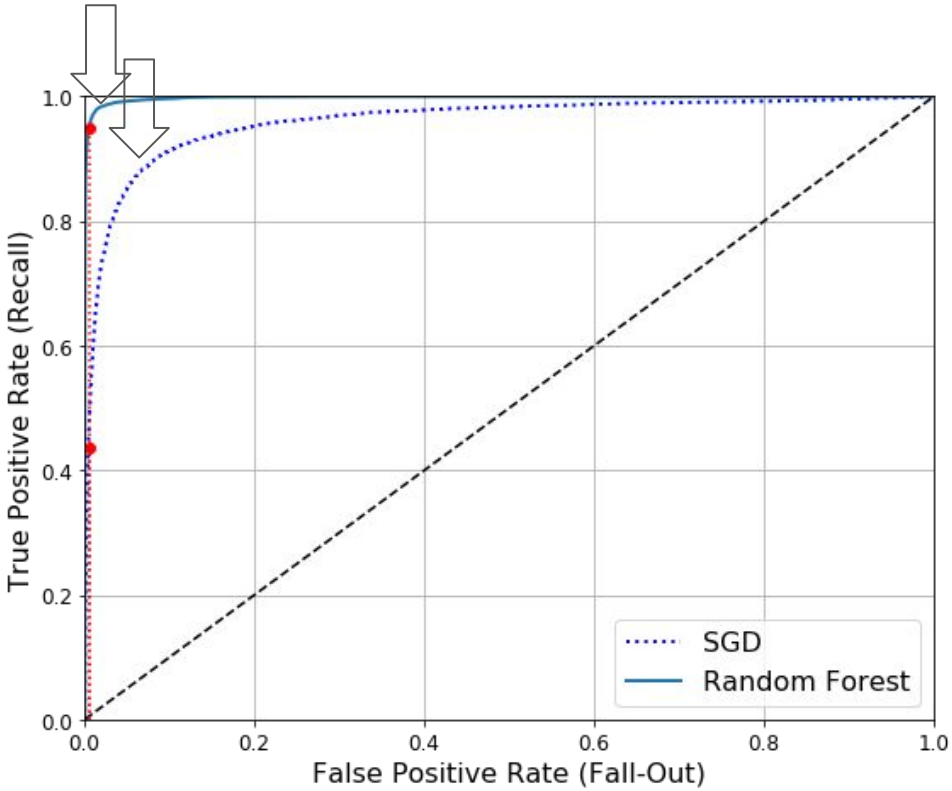
$$\text{FPR (1-specificity)} = \frac{\text{FP}}{\text{TN} + \text{FP}}$$

https://algolytics.com/how-to-assess-quality-and-correctness-of-classification-models-part-4-roc-curve/
https://newsight.tistory.com/53

ROC Curve

AUC

- ROC Curve 아래 면적
- AUC 값이 클수록 성능이 좋은 모델



https://dojinkimm.github.io/ml/2019/10/26/handson-ch3.html

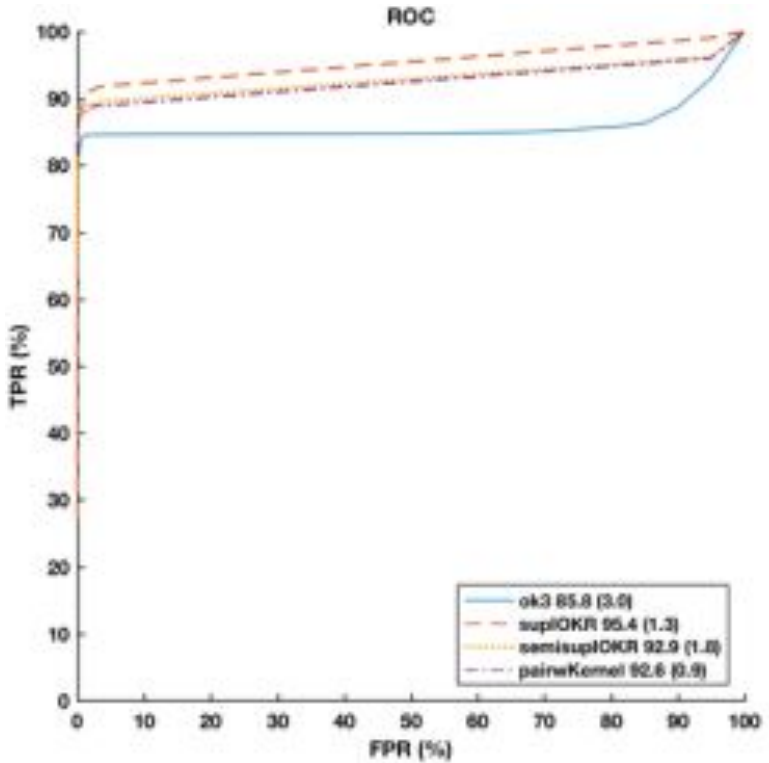
ROC vs PR

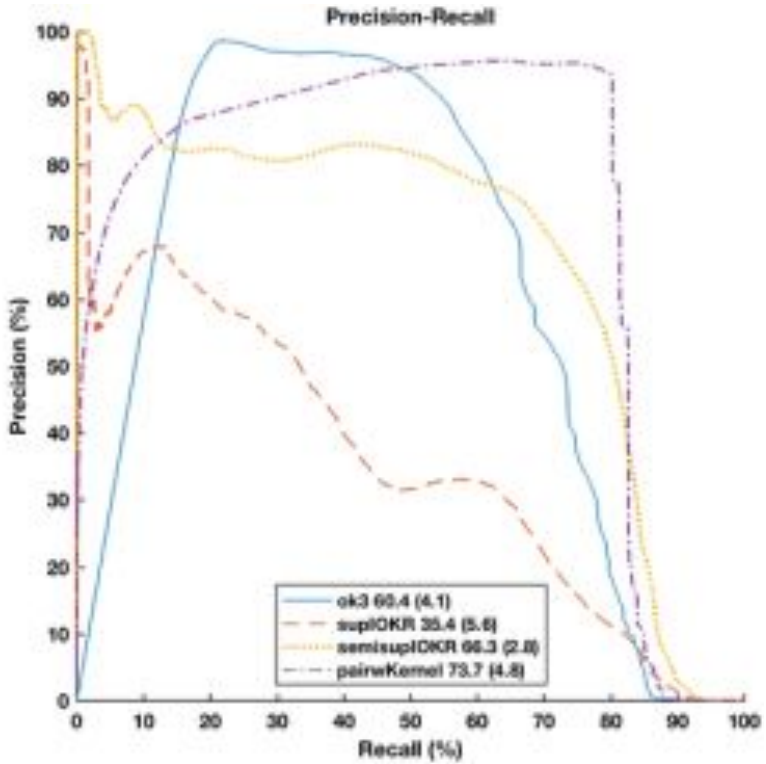
대부분의 경우 ROC 가 활용되나, 아래의 경우 PR 을 활용하기도 한다.

- 모델에 어떤 데이터셋에서 좋은 성능을 보이는지 알고 싶을 때
- Positive 가 매우 적은(편향된) 데이터

주요 평가 Metrics 정리 · lshhhhh/deep-learning-study Wiki · GitHub

ROC vs PR





주요 평가 Metrics 정리 · lshhhhh/deep-learning-study Wiki · GitHub

232

The key difference is that ROC curves will be the same no matter what the baseline probability is, but PR curves may be more useful in practice for needle-in-haystack type problems or problems where the "positive" class is more interesting than the negative class.

To show this, first let's start with a very nice way to define precision, recall and specificity. Assume you have a "positive" class called 1 and a "negative" class called 0. $\hat{Y}$ is your estimate of the true class label Y. Then:

$$\begin{aligned} \text{Precision} &= P(Y = 1|\hat{Y} = 1) \\ \text{Recall= Sensitivity} &= P(\hat{Y} = 1|Y = 1) \\ \text{Specificity} &= P(\hat{Y} = 0|Y = 0) \end{aligned}$$

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)},$$

The key thing to note is that sensitivity/recall and specificity, which make up the ROC curve, are probabilities *conditioned on the true class label*. Therefore, they will be the same regardless of what $P(Y = 1)$ is. Precision is a probability conditioned on *your estimate of the class label* and will thus vary if you try your classifier in different populations with different baseline $P(Y = 1)$. However, it may be more useful in practice if you only care about one population with known background probability and the "positive" class is much more interesting than the "negative" class. (IIRC precision is popular in the document retrieval field, where this is the case.) This is because it directly answers the question, "What is the probability that this is a real hit given my classifier says it is?".

https://stats.stackexchange.com/questions/7207/roc-vs-precision-and-recall-curves

Multiclass classification

분류기를 여러 개 훈련시켜, 각 분류기 점수 중 최고점 class 선택하는 전략

One vs one : labeled class 들의 모든 binary 조합 (1번 2번, 1번 3번, ...) 에 대한 classifier 를 학습시킴 $\binom{10}{2} = 45$

- 예) MNIST(10개 class) 의 경우 45개의 classifier 가 필요

웹 상에 관련 자료나 샘플이 거의 없었음. 잘 사용하지 않는 듯.

```
from sklearn.multiclass import OneVsOneClassifier
ovo_clf = OneVsOneClassifier(SVC(gamma="auto", random_state=42))
# ...
len(ovo_clf.estimators_) # 45
```

Multiclass classification

One vs the rest : 각 class 별로 판단하는 classifier 를 학습시킴

- 예) MNIST 의 경우 10개의 classifier 가 필요

여러 모델을 각각 학습 후, 예측 시 모든 모델의 예측값을 모아서 판단하는 방법이 앙상블 모델과 유사해 보임

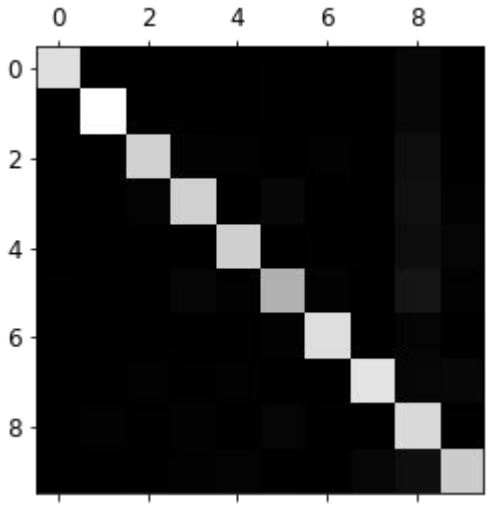
```
from sklearn.multiclass import OneVsRestClassifier
ovr_clf = OneVsRestClassifier(SVC(gamma="auto", random_state=42))

# ...
len(ovr_clf.estimators_) # 10
```

Analyze errors

Confusion matrix

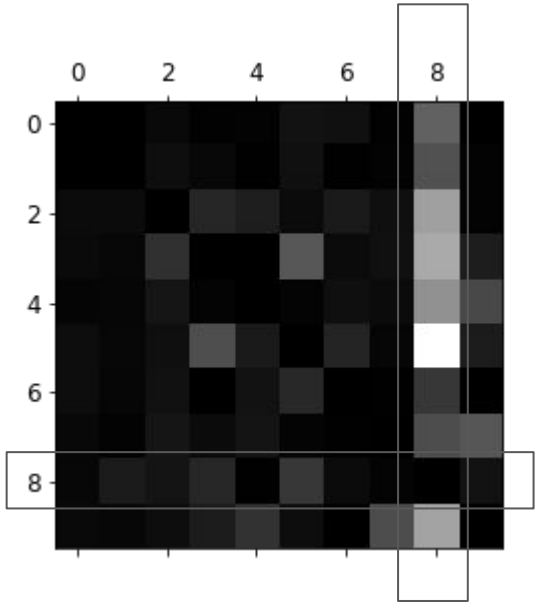
- 해당하는 값이 클수록 white 에 가까움
- diagonal 이 정답을 맞춘 수인데, 정확도가 높다는 것은 이 값들이 크므로, 우리가 알고 싶은 false error 들에 대한 색 구분이 어려움
- MNIST 는 class 별 데이터수가 동일하지 않으므로, matrix 상의 어두운 부분의 diagonal(5) 의 판별력이 낮다고 단정할 수는 없음



https://github.com/rickiepark/handson-ml/blob/master/03_classification.ipynb

Analyze errors

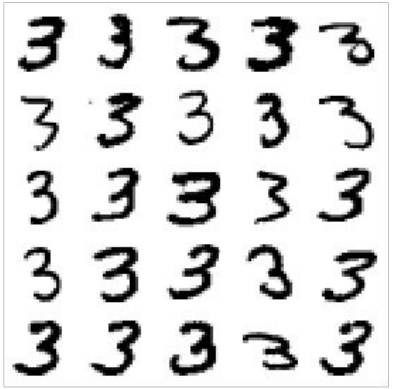
- 각 class 별 이미지 수 만큼으로 나누기하고, diagonal 을 0으로 채워서 에러율 matrix 를 좀 더 눈에 보이게 함
- 숫자 8이 아닌 숫자를 8로 잘못 예측한 경우가 많음 (8th column)
- 숫자 8에 대해 오판하는 경우는 적음(8th row)



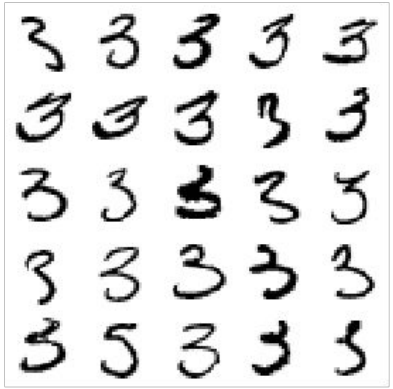
https://github.com/rickiepark/handson-ml/blob/master/03_classification.ipynb

Analyze errors

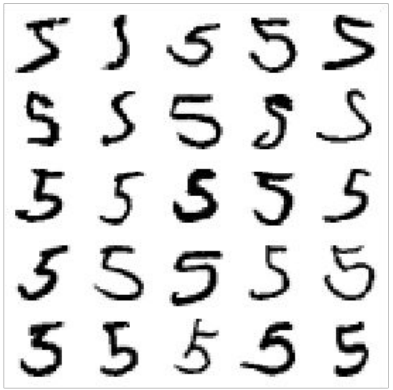
True 3



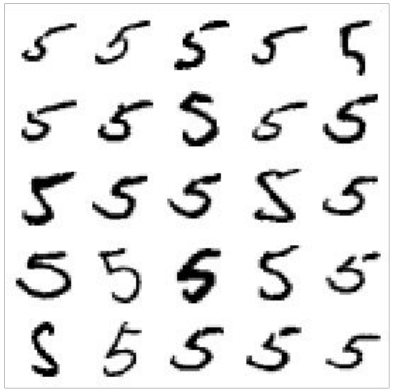
False 5



False 3



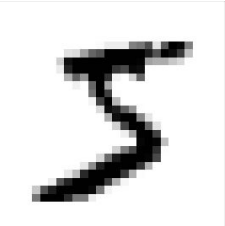
True 5



https://github.com/rickiepark/handson-ml/blob/master/03_classification.ipynb

Multilabel classification

- 데이터로부터 여러 레이블 정보를 학습하고 예측하기 위한 모델
  - 예) 사진에서 여러 사람의 얼굴을 인식
- 학습시 여러 종류의 레이블을 y 로 입력
  - 예) [7보다 큰 수, 홀수 여부] 를 label 로 한다면



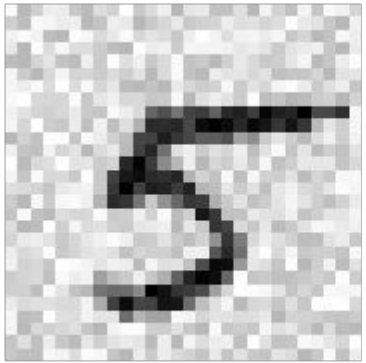
(False : 0, True : 1)

[0, 1]

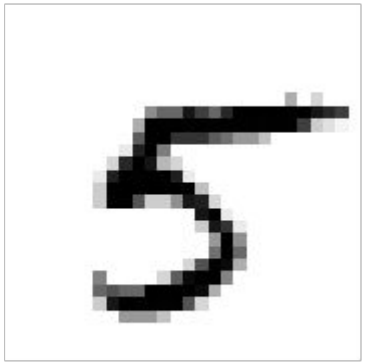
https://github.com/rickiepark/handson-ml/blob/master/03_classification.ipynb

Multioutput classification

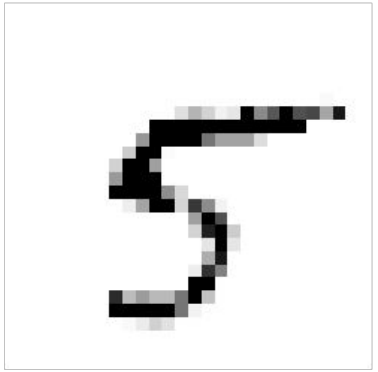
- Multilabel 에서 한 label 이 multiclass 가 될 수 있도록 하는 방법
  - 예) image noise reduction



X



y



pred(X)

https://github.com/rickiepark/handson-ml/blob/master/03_classification.ipynb