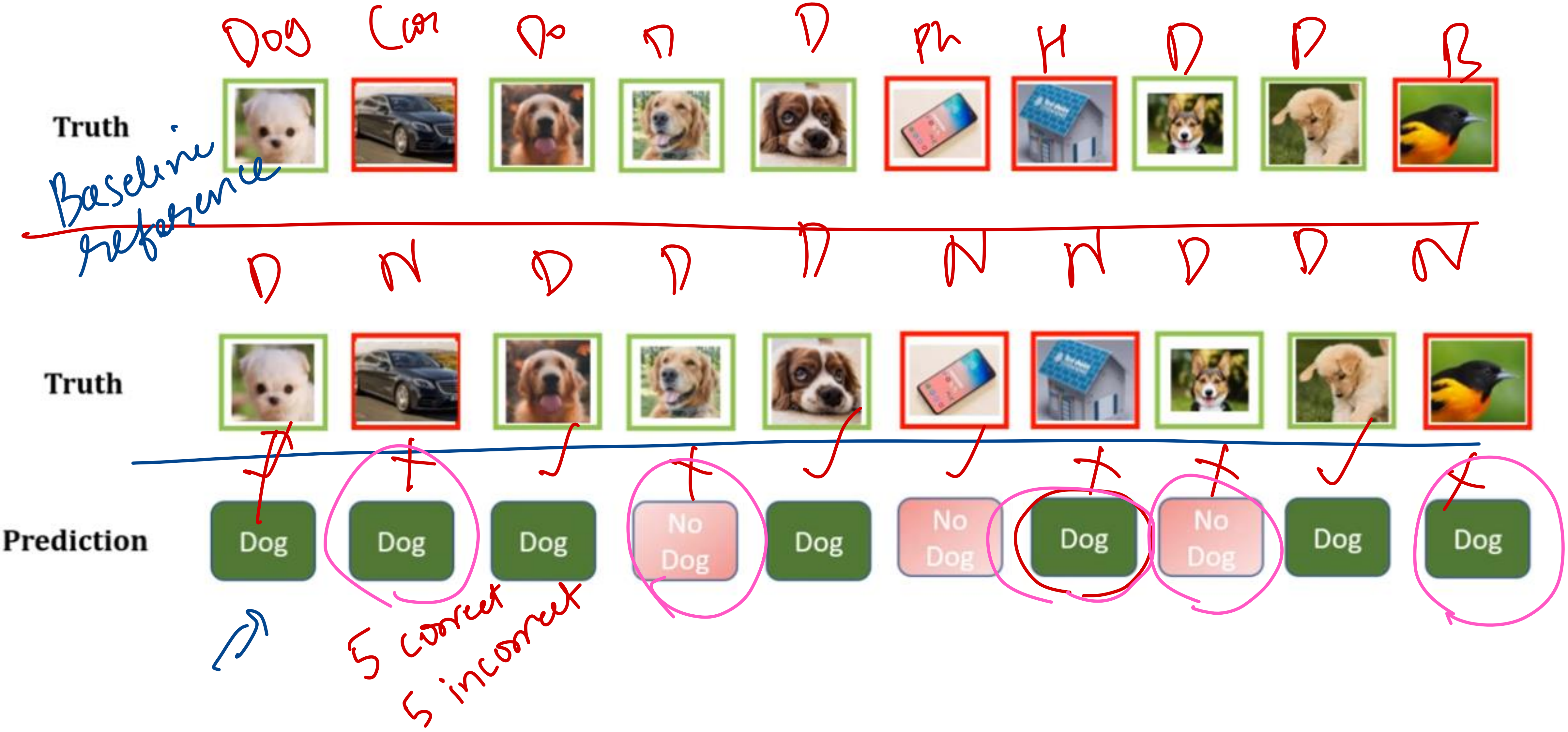
Lecture 6

Logistic Regression & Classification

TOPIC 5
Classification

Truth vs. Prediction



Is it a dog?

Class 1: (+ve) class ⇒ DOG

Class 2: (-ve) class ⇒ No-Dog.

TP = 1

TN = 4

FP = 3

FN = 2

Pr: $\frac{TP}{TP+FP} = \frac{1}{4}$

Re: $\frac{TP}{TP+FN} = \frac{1}{3}$

True Positive = 4

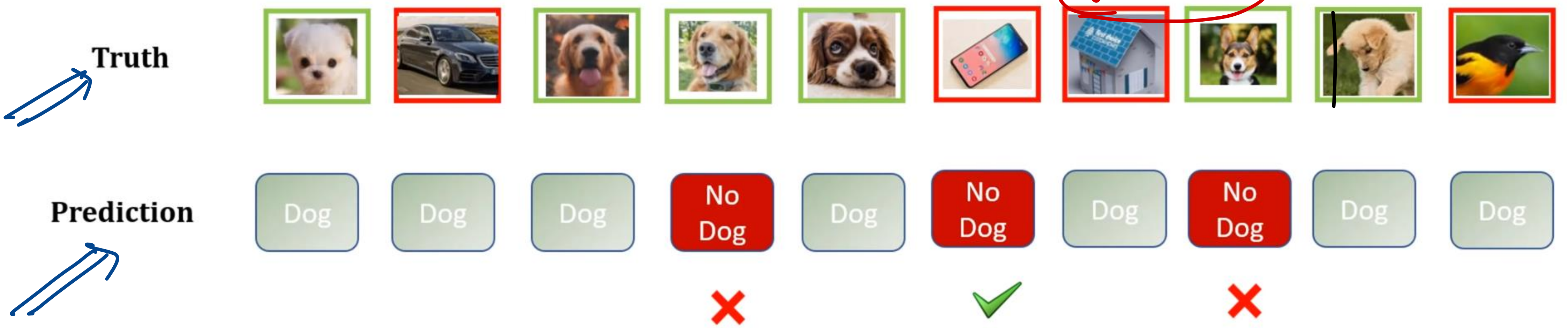
False Positive = 3

Out of all dog truth samples how many did you get ~~right~~

Recall

Out of all dog predictions how many did you get right?

Precision



1/3

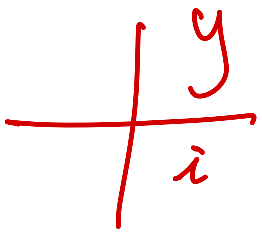
True Negative = 1 **False Negative = 2**

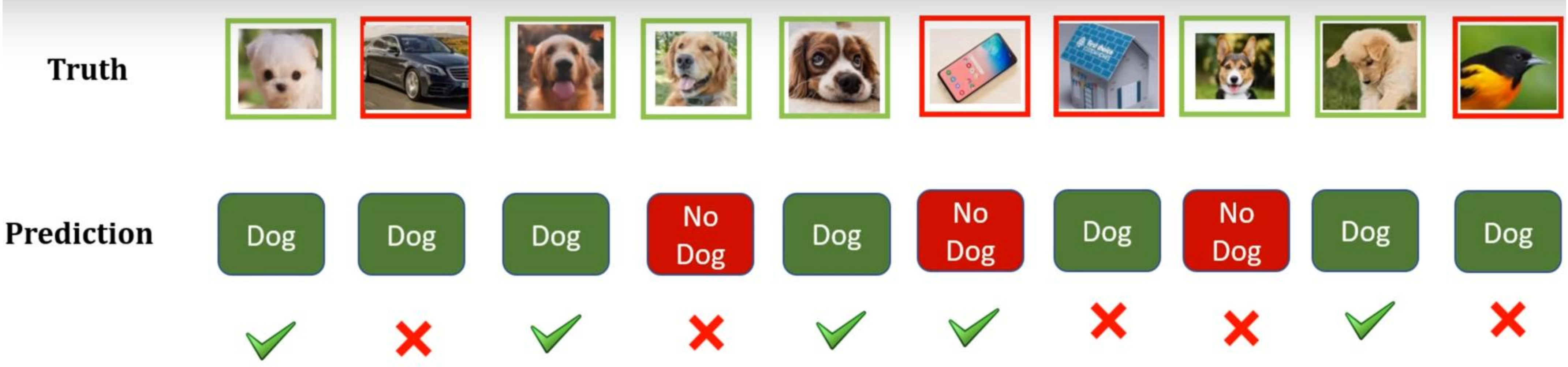
$$\text{Precision} = \frac{TP}{TP+FP} =$$

$$\text{Recall} = \frac{TP}{TP+FN} =$$

	D	N-D
Precision	$\frac{4}{4+3} = \frac{4}{7}$	$\frac{1}{4}$
Recall	$\frac{4}{4+2} = \frac{2}{3}$	$\frac{1}{3}$

$$\frac{\text{No. of correct}}{\text{Total instances}} = \frac{5}{10} = 50\%$$





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

How many we got right? → 5

Accuracy → 5/10 → 0.5

0 1 2 5 6 7 8 9 1 0

0 1 0 5 6 7 8 9 1 0

Prediction: N N N N N N N N N

6 2 3 4 5 1 1 1 0 4

6 3 2 4 5 1 1 1 0 4

N N N N N N N N N N

Acc => 90%

Identify all 2's

2

Not 2.

$\frac{18}{20} = 90\%$

Overall

TP=0

2

Precision vs. Recall

- Precision is Out of all dog predictions how many you got right (Prediction baseline)
- Precision = 4/7
- Precision = TP / (TP+FP)
- Recall (truth is the baseline)
- Out of all dog truth samples how many did you get right?
- Total dog truth samples = 6; True positive = 4
- Recall = TP / (TP+FN)
- Exercise: What is precision and recall for no-dog class ?

1. **Precision:**
   - Precision is the ratio of correctly predicted positive observations to the total predicted positives. It assesses the accuracy of the positive predictions made by the model.
   - The precision formula is given by:
     $$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$
   - In other words, precision is the ability of the classifier not to label as positive a sample that is actually negative.
2. **Recall (Sensitivity or True Positive Rate):**
   - Recall is the ratio of correctly predicted positive observations to the total actual positives. It measures the ability of the model to capture all the relevant positive instances.
   - The recall formula is given by:
     $$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$
   - Recall is concerned with the ability of the classifier to find all the positive instances.

MNIST Data Set

- 70,000 images handwritten by school students
- This “5-detector” will be an example of a *binary classifier*, capable of distinguishing between just two classes, 5 and non-5
- Let’s use stochastic grad. Descent SGDClassifier -

```
from sklearn.linear_model import SGDClassifier

sgd_clf = SGDClassifier(random_state=42)
sgd_clf.fit(X_train, y_train_5)
```

```
from sklearn.datasets import fetch_openml

mnist = fetch_openml('mnist_784', as_frame=False)
```

- X – data (2D Numpy Array)- 28x28 = 784 pixel image. 70000
- y - target (1D Numpy Array as the target) – 70000
- Training first 60,000; test last 10,000 ⟹
- K-fold cross-validation
- Accuracy 95% across all 3 sets ?
- Dummy classifier for non-5s:

```
>>> from sklearn.model_selection import cross_val_score
>>> cross_val_score(sgd_clf, X_train, y_train_5, cv=3, scoring="accuracy")
array([0.95035, 0.96035, 0.9604 ])
```

```
from sklearn.dummy import DummyClassifier

dummy_clf = DummyClassifier()
dummy_clf.fit(X_train, y_train_5)
print(any(dummy_clf.predict(X_train)))  # prints False: no 5s detected
```

Can you guess this model's accuracy? Let's find out:

```
>>> cross_val_score(dummy_clf, X_train, y_train_5, cv=3, scoring="accuracy")
array([0.90965, 0.90965, 0.90965])
```

A: 95%

P: 84%

R: 65%

Confusion Matrices

```
>>> from sklearn.metrics import confusion_matrix
>>> cm = confusion_matrix(y_train_5, y_train_pred)
>>> cm
array([[53892,   687],
       [ 1891,  3530]])
```

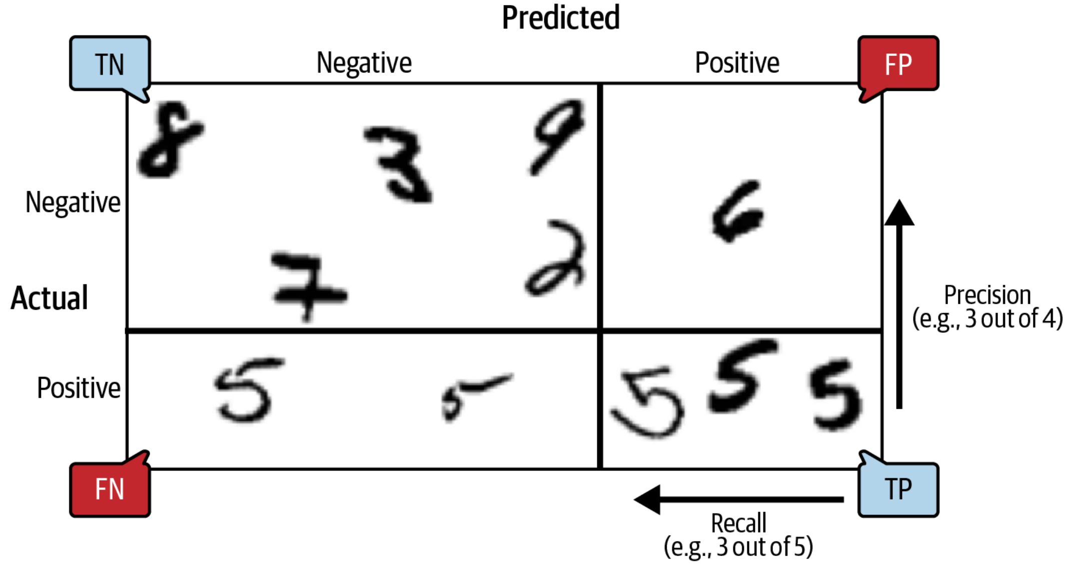
Equation 3-1. Precision

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

TP is the number of true positives, and *FP* is the number of false positives.

```
>>> y_train_perfect_predictions = y_train_5  # pretend we reached perfection
>>> confusion_matrix(y_train_5, y_train_perfect_predictions)
array([[54579,     0],
       [    0,  5421]])
```

Recall, also called *sensitivity* or the *true positive rate* (TPR): this is the ratio of positive instances that are correctly detected by the classifier.



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

```
>>> from sklearn.metrics import precision_score, recall_score
>>> precision_score(y_train_5, y_train_pred)  # == 3530 / (687 + 3530)
0.8370879772350012
>>> recall_score(y_train_5, y_train_pred)  # == 3530 / (1891 + 3530)
0.6511713705958311
```

- When it claims an image represents a 5, it is correct only 83.7% (PRECISION) of the time. Moreover, it only detects 65.1% of the 5s (RECALL).

Evaluation Metrics

Metric	Formula
Average classification accuracy	(TN + TP) / (TN+TP+FN+FP)
Type I error (false positive rate)	FP / (TN + FP)
Type II error (false negative rate)	FN / (FN + TP)
True positive rate	TP / (TP + FN)
True negative rate	TN / (TN + FP)

- Type I error or false positive rate: The chance of incorrectly classifying a (randomly selected) sample as positive
- Type II error or false negative rate: The chance of incorrectly classification a (randomly selected) sample as negative

Evaluation Metrics

Metric	Formula
Precision	TP / (TP + FP)
Recall	TP / (TP + FN)

Precision: Fraction of retrieved instances that are relevant

Recall: Fraction of relevant instances that are retrieved

Evaluation Metrics

Metric	Formula
Sensitivity	TP / (TP + FN)
Specificity	TN / (TN + FP)
Predictive value for a positive result (PV+)	TP / (TP + FP)
Predictive value for a negative result (PV-)	TN / (TN + FN)

Sensitivity: Proportion of actual positives which are correctly identified

Specificity: Proportion of actual negatives which are correctly identified

Sensitivity: The chance of correctly identifying positive samples. A sensitive test helps rule out disease (when the result is negative)

Specificity: The chance of correctly classifying negative samples. A very specific test rules in disease with a higher degree of confidence.

F1 score

```
>>> from sklearn.metrics import f1_score
>>> f1_score(y_train_5, y_train_pred)
0.7325171197343846
```

- A single metric to combine precision and recall - The harmonic mean
- Unlike regular mean, harmonic mean gives more weight to low values.
- Therefore, the classifier's F1 score is only high if both recall and precision are high.
- The F1 score favors classifiers that have similar precision and recall

a, b

$$\frac{2}{\frac{1}{a}+\frac{1}{b}}$$

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{TP}{TP + \frac{FN + FP}{2}}$$

$$\left(\frac{2ab}{a+b}\right)$$

GM

AM

HM

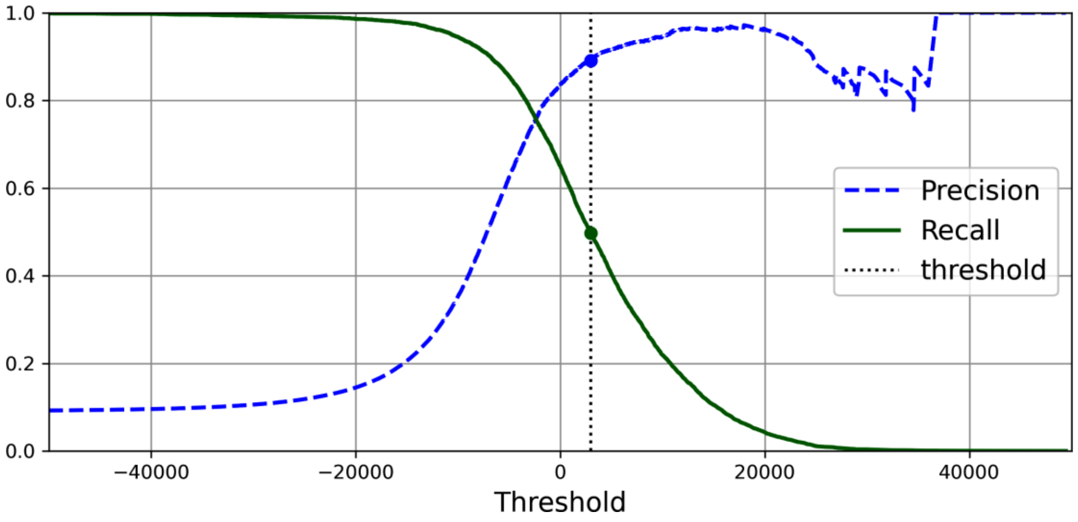
Class 10

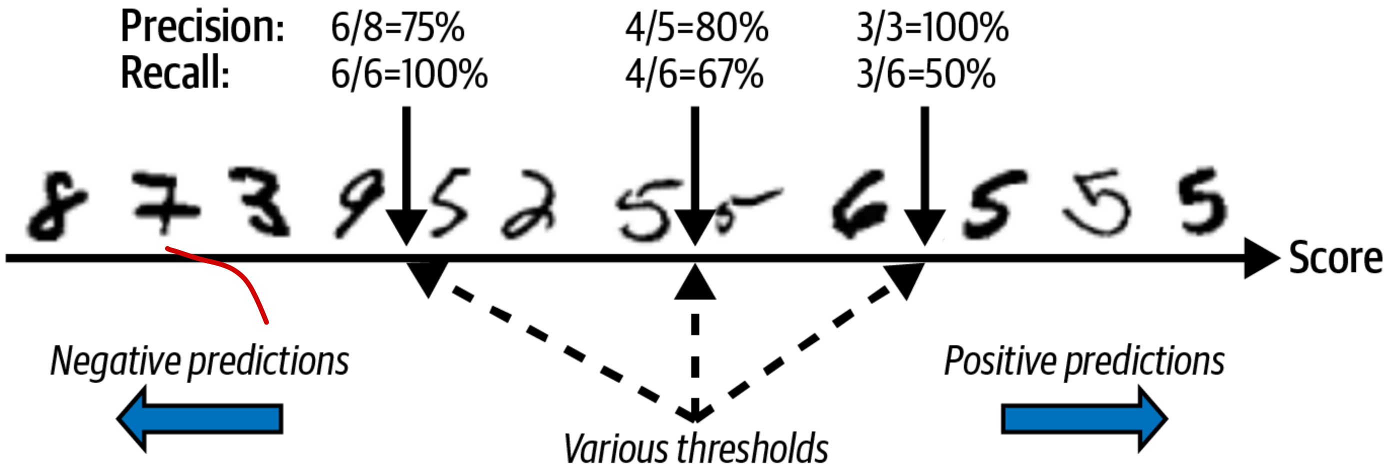
Precision / Recall Tradeoff

- Based on the threshold increasing / decreasing recall / precision.
  - Scikit-Learn does not let you set the threshold directly, but it does give you access to the decision scores that it uses to make predictions

TP= 6 FN=0
FP= 2

TN= FP=0
FN=2 TP=~~2~~ 4





Code ...

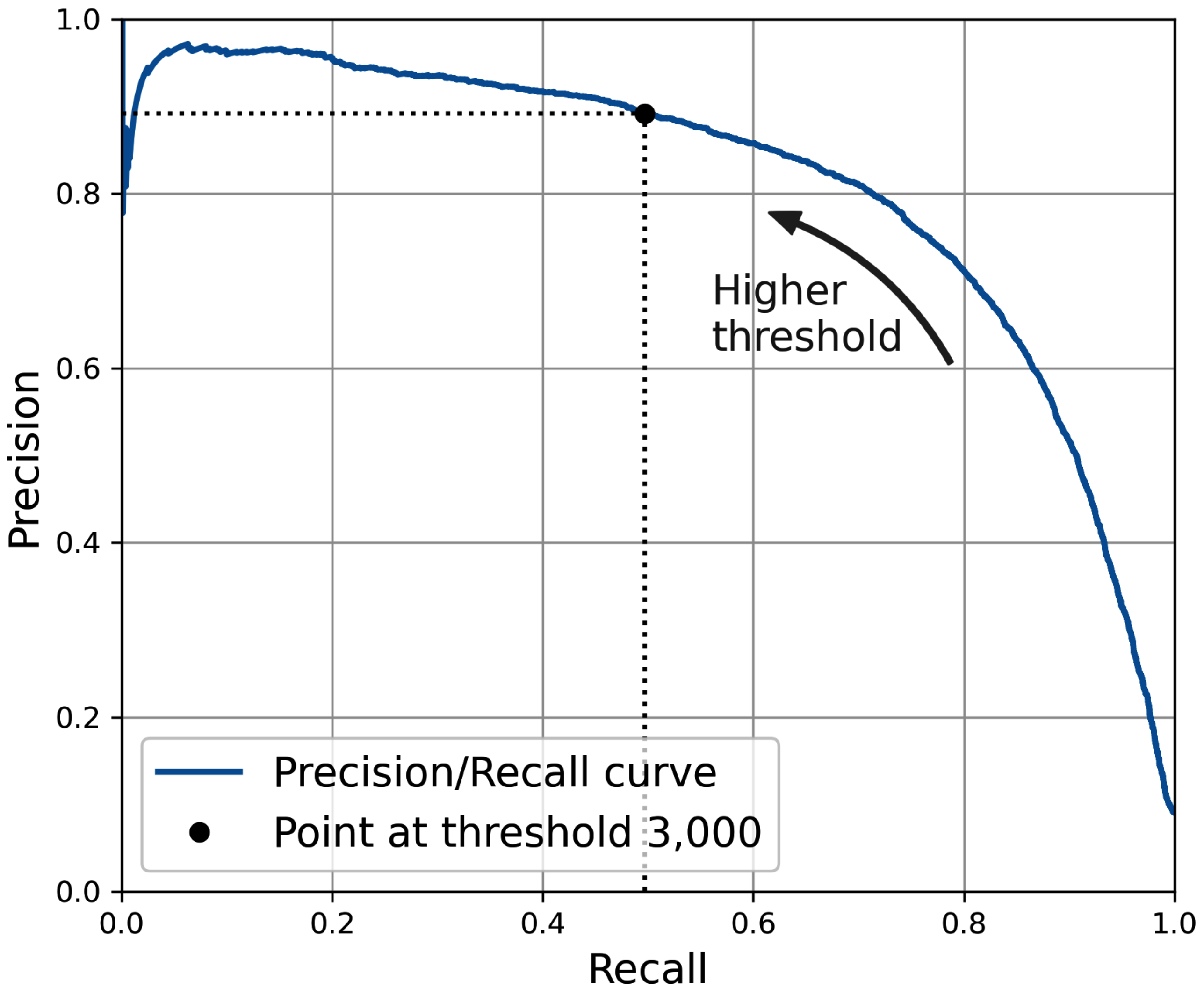
```
>>> y_scores = sgd_clf.decision_function([some_digit])
>>> y_scores
array([2164.22030239])
>>> threshold = 0
>>> y_some_digit_pred = (y_scores > threshold)
array([ True])
```

The `SGDClassifier` uses a threshold equal to 0, so the preceding code returns the same result as the `predict()` method (i.e., `True`). Let's raise the threshold:

```
>>> threshold = 3000
>>> y_some_digit_pred = (y_scores > threshold)
>>> y_some_digit_pred
array([False])
```

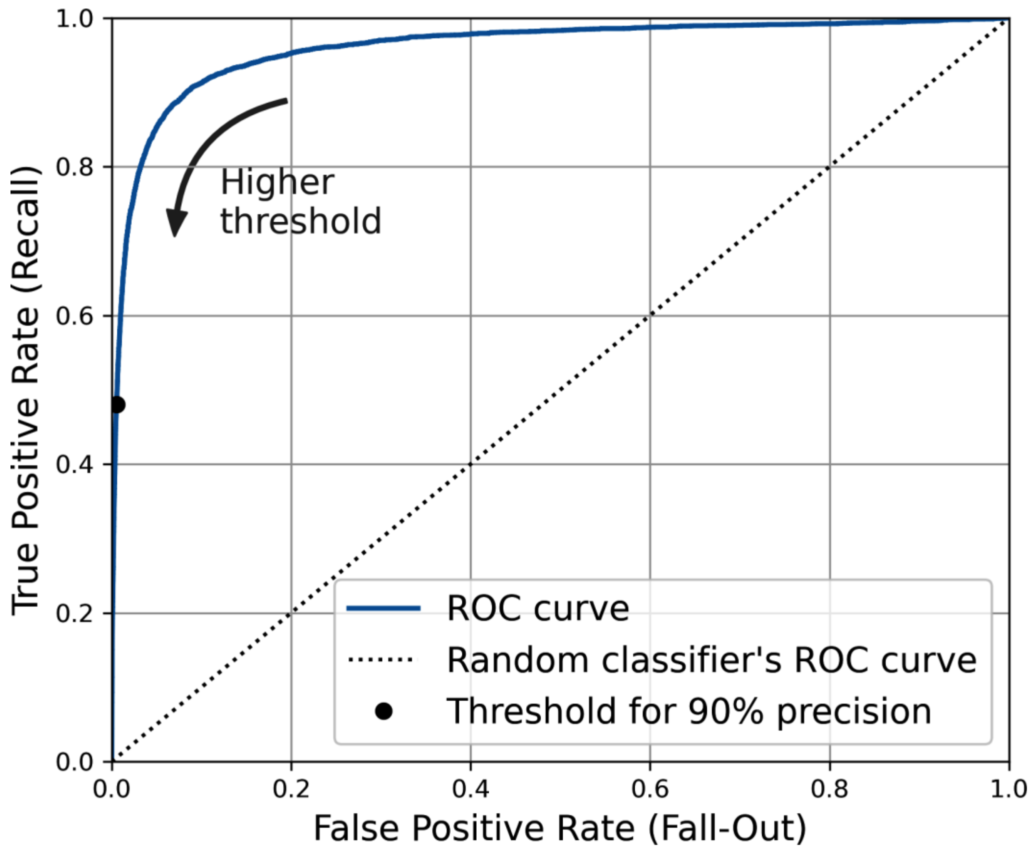
Precision vs. Recall

- A high precision classifier is not that useful if the recall is low. 48% recall is not great.
- Something around 60% Recall in this case is a decent solution.



ROC Curve

- Receiver Operating Characteristic (ROC)
- Plots TPR (Recall) vs. FPR (Fall out)
- FPR = 1-TNR
- Sensitivity vs. 1-specificity
- There is a trade-off: the higher the recall (TPR), the more false positives (FPR) the classifier produces.
- The dotted line represents the ROC curve of a purely random classifier; a good classifier stays as far away from that line as possible (toward the top-left corner).
- One way to compare classifiers is to measure the *area under the curve* (AUC). A perfect classifier will have a ROC AUC equal to 1, whereas a purely random classifier will have a ROC AUC equal to 0.5.
- Scikit-Learn provides a function to estimate the ROC AUC:

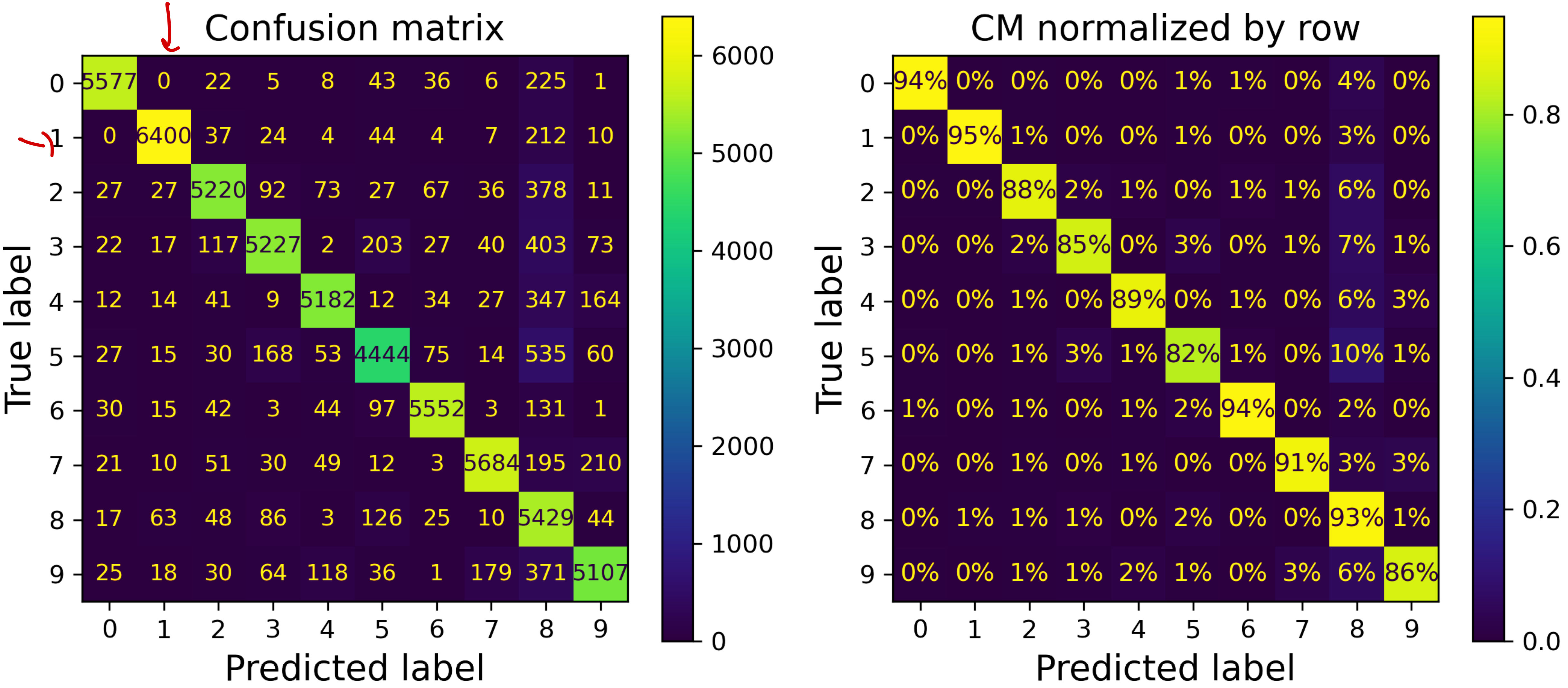



```
>>> from sklearn.metrics import roc_auc_score
>>> roc_auc_score(y_train_5, y_scores)
0.9604938554008616
```

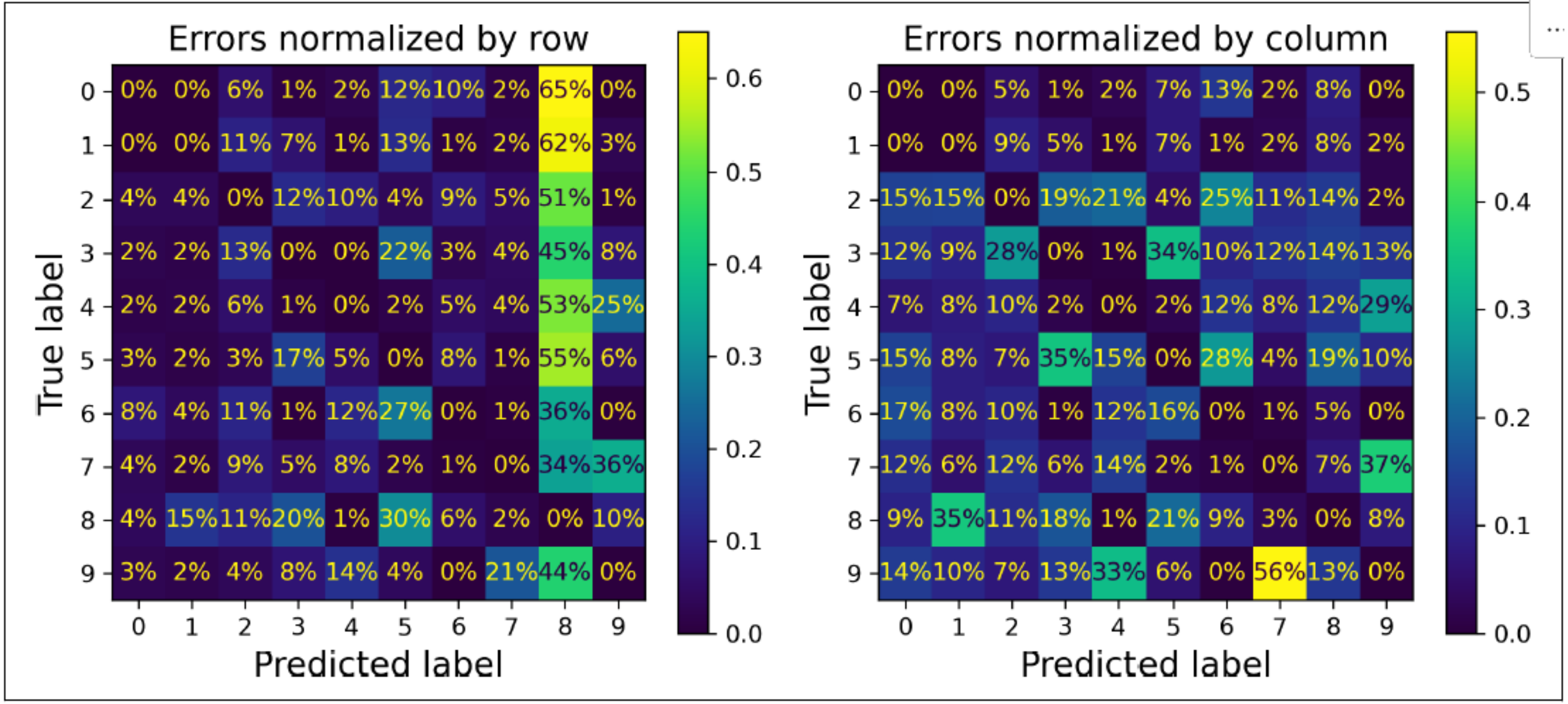
Multiclass Classifier

- Can distinguish between 2 or more classes.
- LogisticRegression, RandomForestClassifier, GaussianNB – multiclass
- SGDClassifier, SVC – are binary classifiers.
- One-versus-one (OvO) strategy
- One-versus-all (OvA) strategy

Error Analysis: Confusion matrix (left) and the same CM normalized by row (right)



- Confusion matrix with errors only, normalized by row (left) and columns (right)



Multilabel Classification

- Each instance has always been assigned to just one class. But in some cases you may want your classifier to output multiple classes for each instance.
- Consider a face-recognition classifier: what should it do if it recognizes several people in the same picture? It should attach one tag per person it recognizes.
- Say the classifier has been trained to recognize three faces: Alice, Bob, and Charlie.
- Then when the classifier is shown a picture of Alice and Charlie, it should output [True, False, True] (meaning “Alice yes, Bob no, Charlie yes”).

- **Self Reading** task: Multioutput Classification

Cross Validation (Self Reading)

- It is used for

– Performance evaluation: Evaluate the performance of a classifier using the given data

– Model Selection: Compare the performance of two or more algorithms (DT classifier and neural network) to determine the best algorithm for the given data

– Tuning model parameters: Compare the performance of two variants of a parametric model

- K-Fold Cross-Validation
  - Data is partitioned into k equal folds (partitions). k-1 folds are used for training and 1-fold for testing
  - The procedure is repeated k times

Test	Train	Train	Train
Train	Test	Train	Train
Train	Train	Test	Train
Train	Train	Train	Test

- Leave-One-Out Cross-Validation
  - Special case of k-fold cross validation where k=number of instances in the data
  - Testing is performed on a single instance and the remaining are used for training

- **TYPES of cross Validation**
- Resubstitution Validation
- Hold-Out Validation
- K-Fold Cross-Validation
- Leave-One-Out Cross-Validation
- Repeated K-Fold Cross-Validation

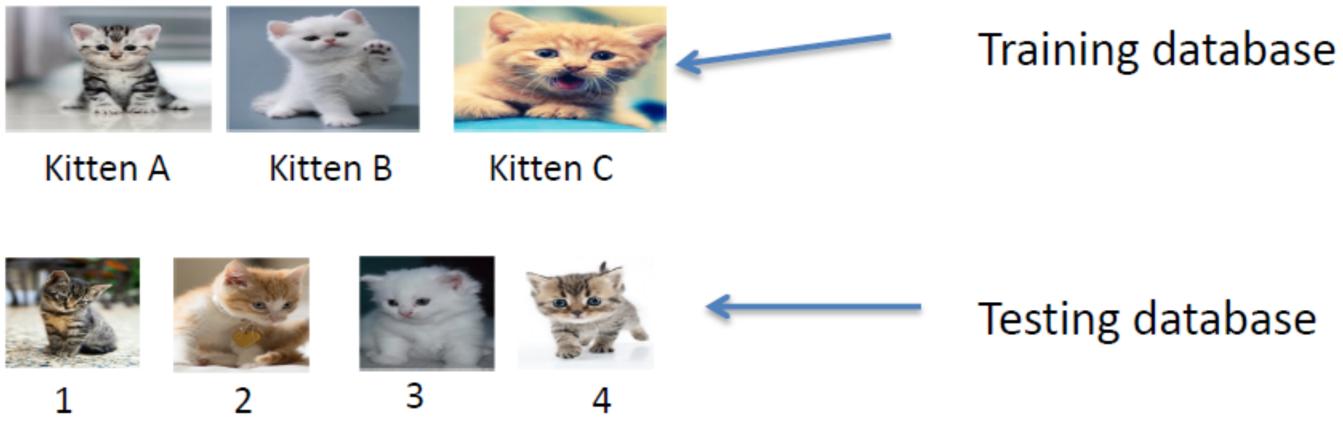
- Repeated K-Fold Cross-Validation
  - Repeat k-fold cross validation multiple times

- Hold-Out Validation
  - The database is partitioned into two non-overlapping parts, one for training and other for testing
    - The results depend a lot on the partition, may be skewed if the test set is too easy or too difficult

- Resubstitution Validation
  - All the available data is used for training and the same data is used for testing
    - Does not provide any information about generalizability

Cross Validation

- "Cross-Validation is a statistical method of evaluating and comparing learning algorithms."
- The data is divided into two parts:
  - – Training: to learn or train a model
  - – Testing: to validate the model



Validation Method	Advantages	Disadvantages
Resubstitution	Simple	Overfitting
Hold-out validation	Independent training and testing sets	Reduced data for training and testing
K-fold cross validation	Accurate performance estimation	Small sample for performance estimation, underestimated performance variance or overestimated degree of freedom for comparison
Leave-one-out cross validation	Unbiased performance estimation	Very large variance
Repeated k-fold cross-validation	Large number of performance estimates	Overlapped training and test data between each round, underestimated performance variance or overestimated degree of freedom for comparison