MACH

Classification

MNIST Dataset as a running example

➔ A set of 70,000 small images of digits handwritten by high school students and employees of the US Census Bureau.

- Each image is labeled with the digit it represents.
- A.k.a. the « hello world » of Machine Learning.

```
from sklearn.datasets import fetch_openml

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

➔ as_frame = False to get the data as Numpy arrays instead of Panda DataFrame.

➔ 70,000 images, each image has 28x28 (features). Each feature simply represents the pixel intensity (white:0 to black:255)

The classification task

- ➔ Learn a model to recognize digits
- ➔ Create a test set and set it aside before inspecting the data closely.
- ➔ This dataset is **already shuffled**.

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

Binary Classification

Training a Binary classifier

- We simplify the problem and we want to identify only one digit.
  - E.g., the ‘5’
- The 5-detector is an example of a binary classifier capable of distinguishing between juste two classes: 5 and not-5.

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

- Next step: pick a classifier and train it.
  - Stochastic Gradient Descent (SGD)
    - Capable of handing large dataset efficiently
    - Suited for Online learning.

```
from sklearn.linear_model import SGDClassifier

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

Performance measures

- Evaluating a classifier is often trickier than evaluating a regressor
- Many performance measures

Measuring accuracy using cross-validation

- K-fold cross-validation:
  - splitting the training set into k folds
  - Then, training the model k times, holding out a different fold each time for the evaluation.
- 3-fold cross-validation on MNIST:
  - [0.95035, 0.96035, 0.9604]
  - Above 95% accuracy (ratio of correct predictions) on all cv folds !!!
  - Amazing ! Do you believe it ?
- Look at a dummy classifier (just classify every image in the most frequent class:
  - [0.90965, 0.90965, 0.90965]

➔ Accuracy is generally not the preferred performance measure for classifier, especially when dealing with **skewed** datasets (unbalanced).

Confusion Matrices

- Counting the number of times instances class A are classified as class B, for all A/B pairs.

		Predicted class	
		Non-5	5
Actual class	Non-5	53892	687
	5	1891	3530

		Predicted class	
		Negative	Positive
Actual class	Negative	TN	FP
	Positive	FN	TP

Precision = TP / (TP + FP)

Recall = TP / (TP+FN)

Precision / Recall: intuitions

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

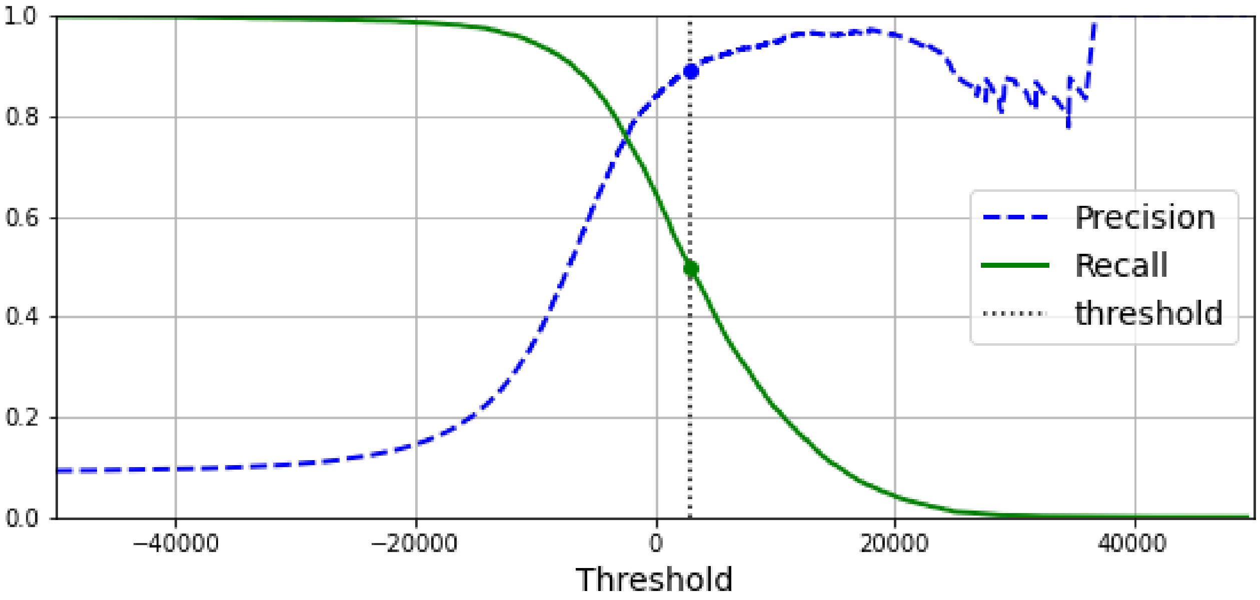
$$\text{Recall} = \frac{TP}{TP + FN}$$

F measure to combine precision and recall

- It is often convenient to combine precision and recall into a single metric
- F_1 score is the harmonic mean (more weight to low values) of precision and recall
  - $F_1 = 2 \times \frac{precision \times recall}{precision + recall} = \frac{TP}{TP + \frac{FN+FP}{2}}$
- This measure favors classifiers with similar precision and recall.
- Increasing precision reduces recall and vice versa: aka precision/ recall trade-off

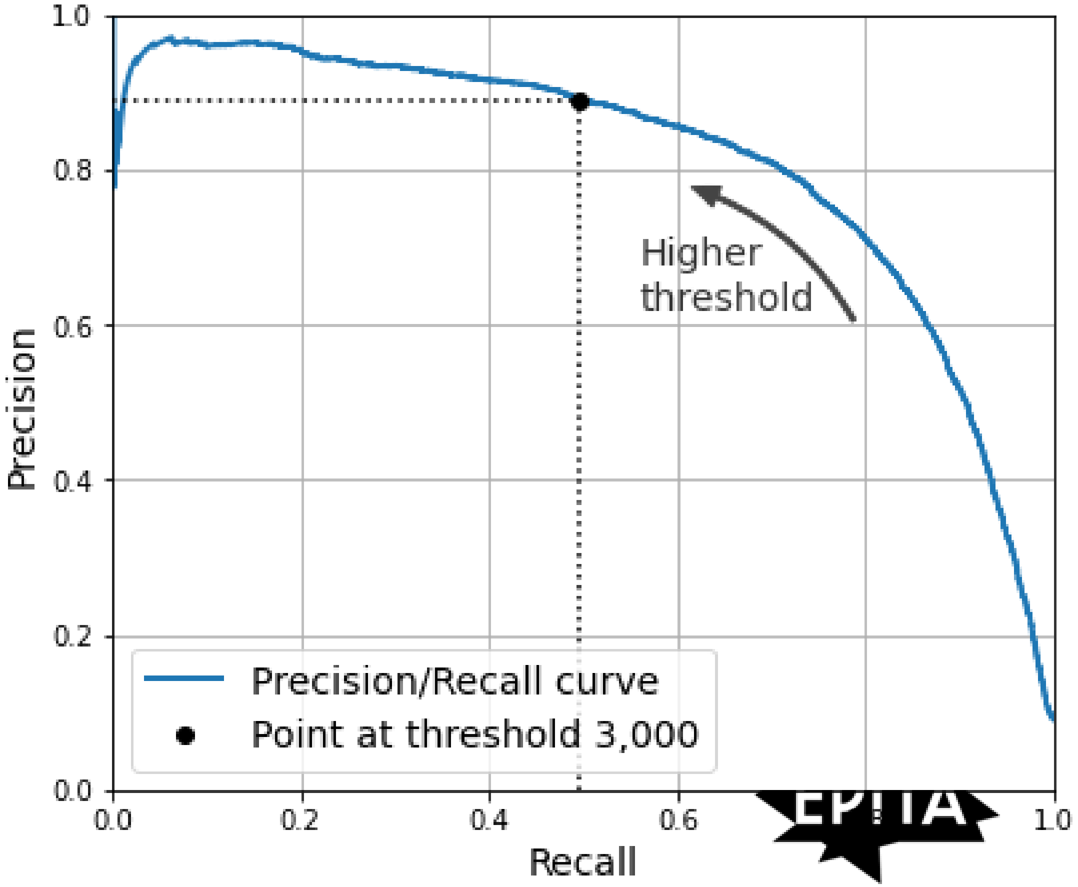
The precision/recall trade-off

- SGD classifier (and not only) computes a score based on a **decision function**.
    - **If** this score is **greater** than a **threshold**, it assigns the instance to the **positive class**; **otherwise** it assigns it to the **negative** class.



Precision / recall curve

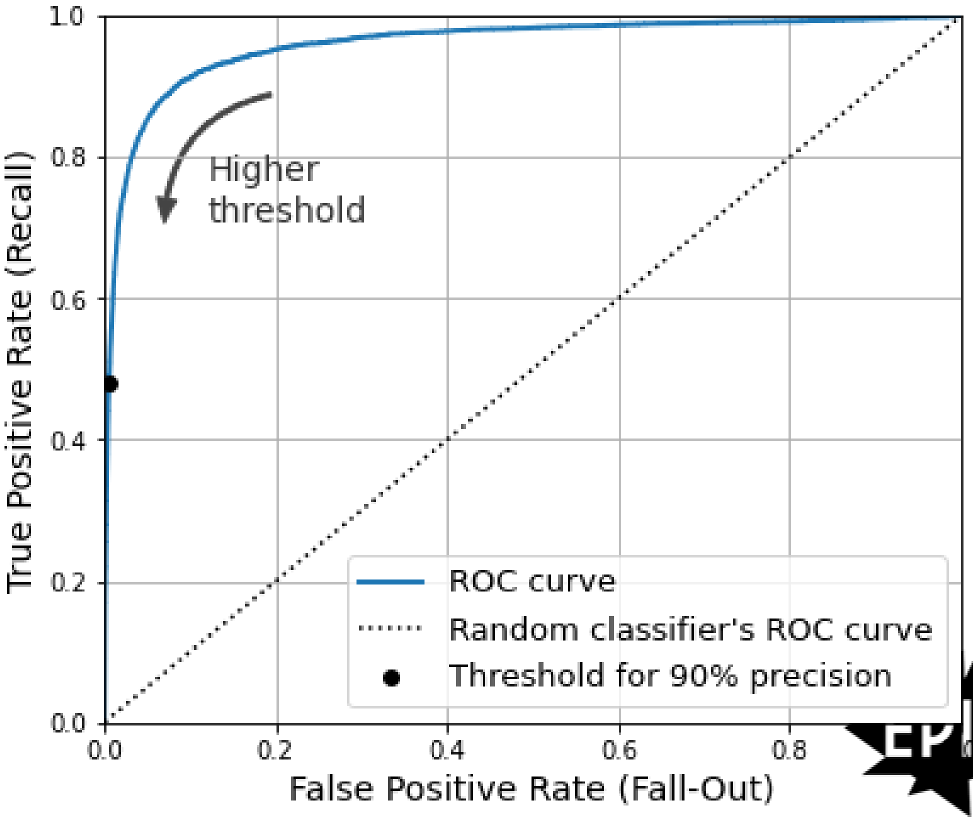
- This plot can be used to set the threshold
- E.g., one wants a 90% precisions
  - Search for the lowest threshold that gives at least 90% precision.
- Easy to create a classifier with desired precision
  - What about the recall?



The ROC Curve

- The **receiver operating characteristic** (ROC) is a common tool used with binary classifier.
    - Very similar to precision/recall curve
    - The ROC curve plots the **TP rate** (recall) vs the **FP rate** (also called the *fall-out*)
    - FPR= ratio of negative instances that are incorrectly classified as positive.
        - FPR = 1 – TNR
    - TNR: ratio of negative instances that are correctly classified as negative.
    - TNR: also called specificity
- ROC curve plots sensitivity (recall) versus 1-specificity.

- Again a trade-off
- One way to compare classifier is to measure the area under the curve (AUC).
- A perfect classifier will have a ROC AUC equal to 1.
- A purely random:
    - ROC AUC =0.5

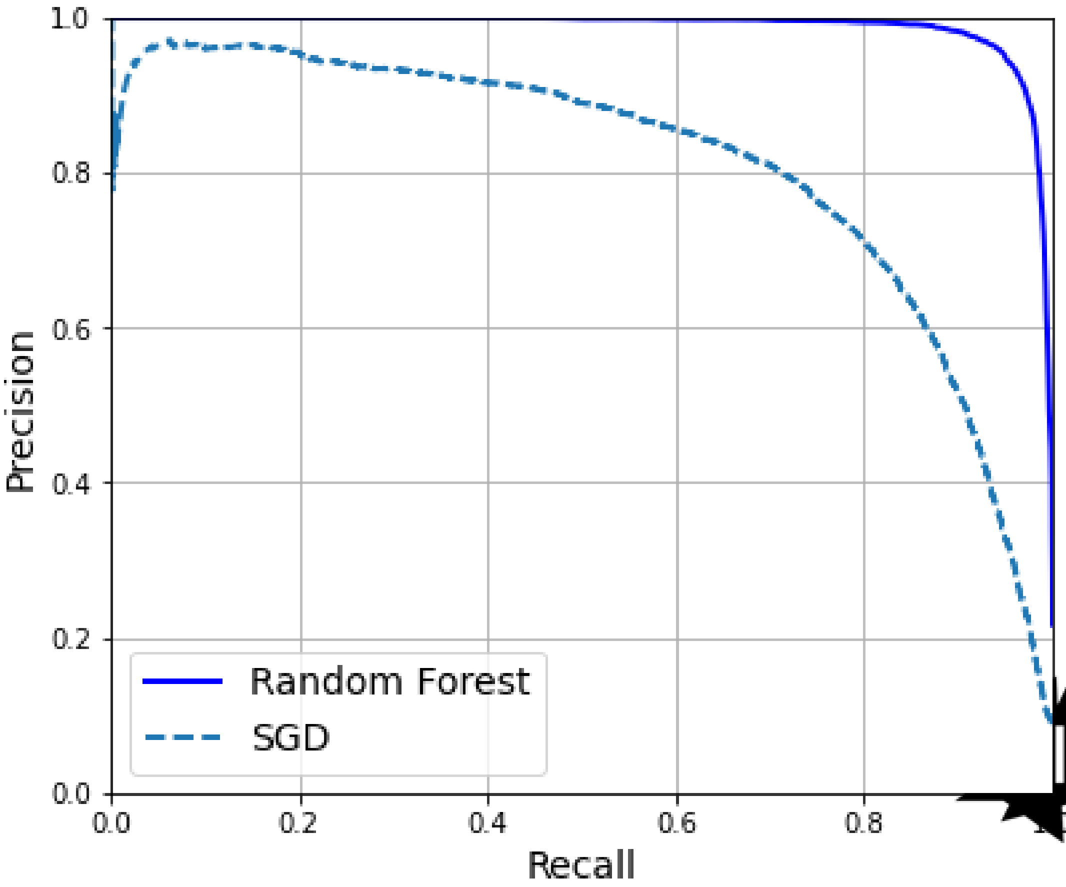


ROC curve or PR curve ?

- They are similar so: **how to decide which one to use ?**
- Prefer PR curve whenever
  - **the positive class is rare**
  - Or you **care more about the false positives than the false negatives**.
- Otherwise, use the ROC curve (and ROC AUC score).
- Example:
  - Considering the previous ROC curve you may think that the classifier is really good but this is mostly because there are few positives (5s) compared to the negatives (non-5s). In constrast, PR curve makes it clear that the classifier has room for improvement.

Considering another classifier.

- Random Forest
- ROC AUC : 0.99 (vs 0.96 for SGD)



Multiclass Classification

To distinguish between more than two classes

To distinghish between more than two classes

- Aka: *multinomial* classifiers
- Some classifiers are able to handle multiple classes **natively** (e.g., Logistic reg., Random Forest, GaussianNB)
- Others are **strictly binary classifiers** (e.g., SGD, SVC)

➔ There are **various strategies** to perform multiclass classification with **multiple binary classifiers**.

   ➔ *One-Versus-All* (OVA/OVR)
   ➔ *One-Versus-One* (OVO)

One-versus-the-rest / one-versus-all (OVR/OVA)

- Create a system that can classify the instances into k classes by training k binary classifiers.
  - One classifier for each class.
  - To classify a new instance:
    - take the decision for each classifier
    - Select the class whose classifier outputs the **highest** score.
- MNIST: train 10 binary classifiers (one for each digit)
  - 0-detector, 1-detector, …, 9-detector

One-versus-One

- Train a binary classifier for every pair of labels.
  - One to distinguish 0s and 1s, another to distinguish 0s and 2s …
  - Need to train 45 binary classifiers
  - If N classes ➔ (N x (N-1) / 2) classifiers.
- To classify an image:
  - Run the image through all the classifiers
  - See which class wins the most duels.

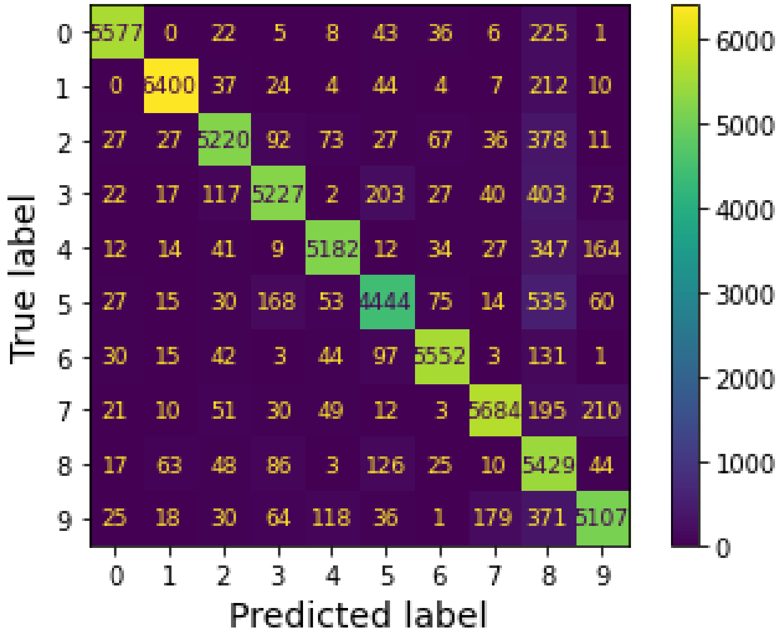
+ : each classifier only needs to be trained on the part of the training set containing the two classes it must distinguish.

  - Some algorithms (e.g., SVM) scale poorly with the size of the training set ➔ OvO is preferred because it is faster to train many classifiers on small training set than few classifiers on large training sets.

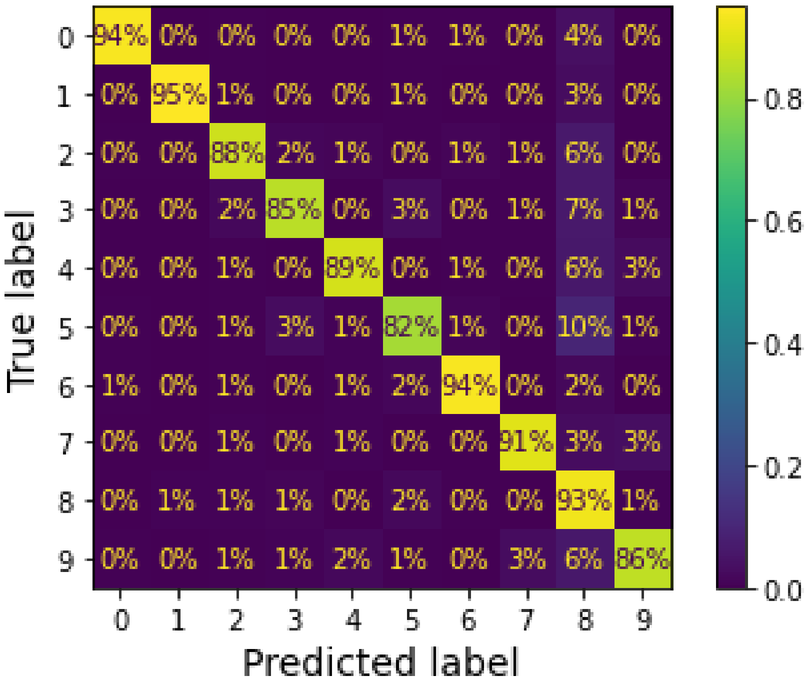
- Scikit-Learn detect when a binary classification algorithm is used for a multiclass classification task.
- To force Scikit-Learn to use OvO or OVR:
  - OneVsOneClassifier or OneVsRestClassifier classes
  - Simply create an instance and pass a classifier to its constructor.
    - `ovr_clf = OneVsRestClassifier(SVC(random_state=42))`
    - `ovr_clf.fit(X_train[:2000], y[train:2000])`
    - `over_clr.predict([some_digits])`

Error Analysis

```
from sklearn.metrics import ConfusionMatrixDisplay

y_train_pred = cross_val_predict(sgd_clf, X_train_scaled, y_train, cv=3)
plt.rc('font', size=9)  # extra code – make the text smaller
ConfusionMatrixDisplay.from_predictions(y_train, y_train_pred)
plt.show()
```



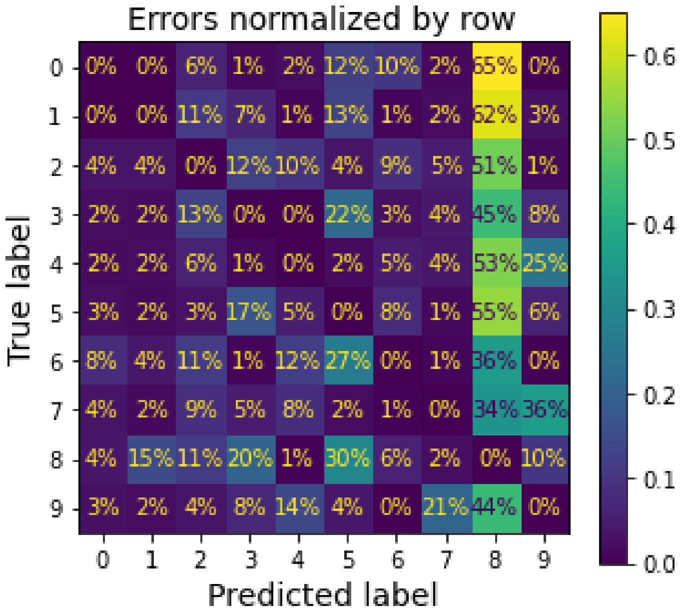
```python
plt.rc('font', size=10)  # extra code
ConfusionMatrixDisplay.from_predictions(y_train, y_train_pred,
                                        normalize="true", values_format=".0%")
plt.show()
```

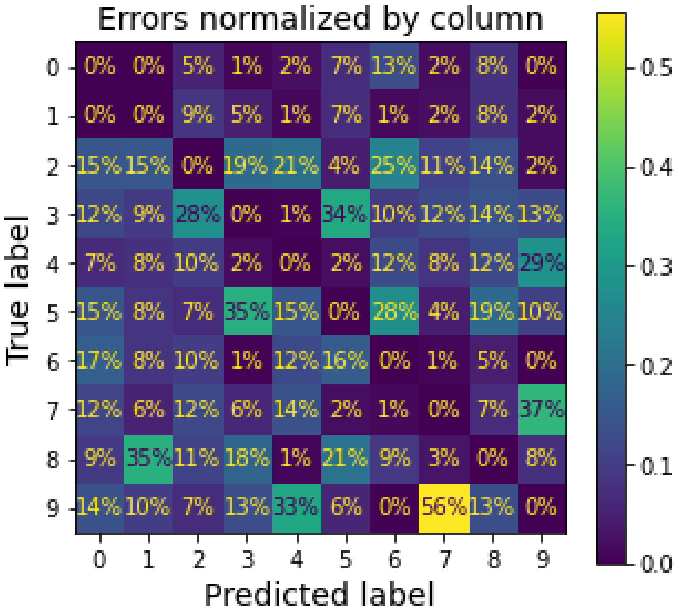


82% of ’5’s are correctly classified

10% of instances classified as ‘5’s are actually ‘8’s

- (0,8) left means 65% of errors the model made on ’0’s were misclassified as ‘8’s.
- (9,7) right means 56% of misclassified ‘7’s are actually ’9’s.





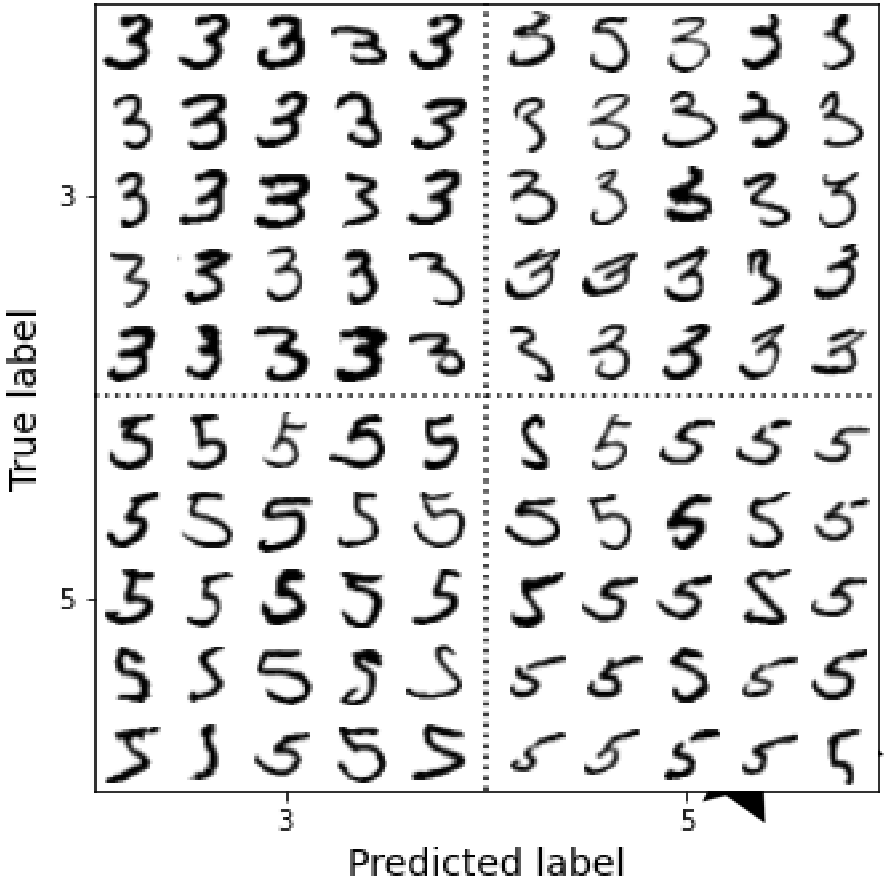
EPITA

Go further

- The classifier is quite sensitive to:
  - Image shifting and rotation.

➔ Heavy and conscientious preprocessing

➔ Or **data augmentation**.



Multilabel Classification

- Until now, each instance has been assigned to just one class.
- In some cases, you may want your classifier to output multiple classes for each instance.
  - Face recognition: several people in the same picture.
  - News: may have several topics (e.g., diplomacy, sport, politics, business).

➔ A system that outputs **multiple binary tags** is called a **multilabel classification system**.

Ex: Large and odd digit ?

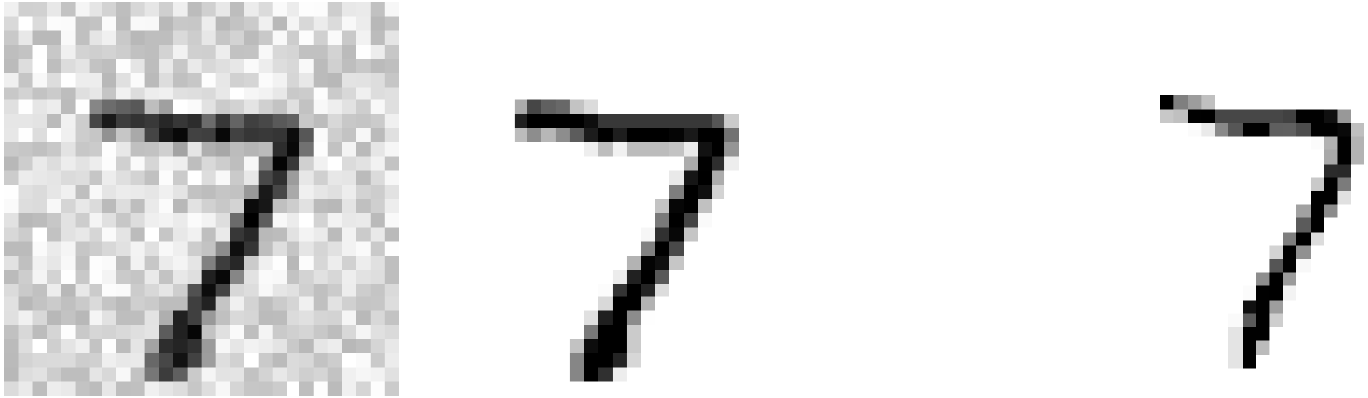
```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier

y_train_large = (y_train >= '7')
y_train_odd = (y_train.astype('int8') % 2 == 1)
y_multilabel = np.c_[y_train_large, y_train_odd]

knn_clf = KNeighborsClassifier()
knn_clf.fit(X_train, y_multilabel)
```

- To go further: have a look at ChainClassifier to capture dependency between labels.

Multioutput Classification

Multi-output Classification

- Multioutput-multiclass Classification or just Multioutput Classification
- A generalization of multilabel classification where each label can be multiclass (i.e., can have more than two possible values).
- Example: image denoising



Summary

Conclusion

- Now, you know:
  - How to select good metrics for classification tasks,
  - Pick the appropriate precision/recall trade-off,
  - Compare classifiers,
  - Build good classification systems on a variety of tasks.
- Next level:
  - Learn how all these machine learning models actually work.

EPITA

The end

- Exercices
- One mini-project next time