3253 Machine Learning

Data Science Fundamentals Certificate

Module 3

CLASSIFICATION

Course Roadmap

Module / Week	Title
1	Introduction to Machine Learning
2	End to End Machine Learning Project
3	Classification
4	Clustering & Un-Supervised Learning
5	Training Models & Feature Selection
6	Support Vector Machines
7	Decision Trees, Ensemble Learning & Random Forests
8	Dimensionality Reduction
9	Introduction to TensorFlow and Neural Networks
10	Training Deep NNs
11	Distributing TensorFlow and Other Architectures
12	External Speakers and Students Presentations

Introduction

Week 1 we mentioned that the most common supervised learning tasks are regression (predicting values) and classification (predicting classes).

Week 2 we explored a regression task, predicting housing values, using various algorithms such as Linear Regression, Decision Trees, and Random Forests.

Now we will turn our attention to classification systems.

Module 3: Learning Outcomes

- Binary Classification
- Performance Measures
- Multiclass Classification
- Error Analysis
- Multi label Classification

Required/Recommended Readings

Required:

- Hands-On Machine Learning with Scikit-Learn and TensorFlow - *Aurélien Géron* **(Chapter 3)**

Recommended:

- C. M. Bishop Pattern Recognition and Machine Learning
  (Chapter 3 Linear Models for Regression)

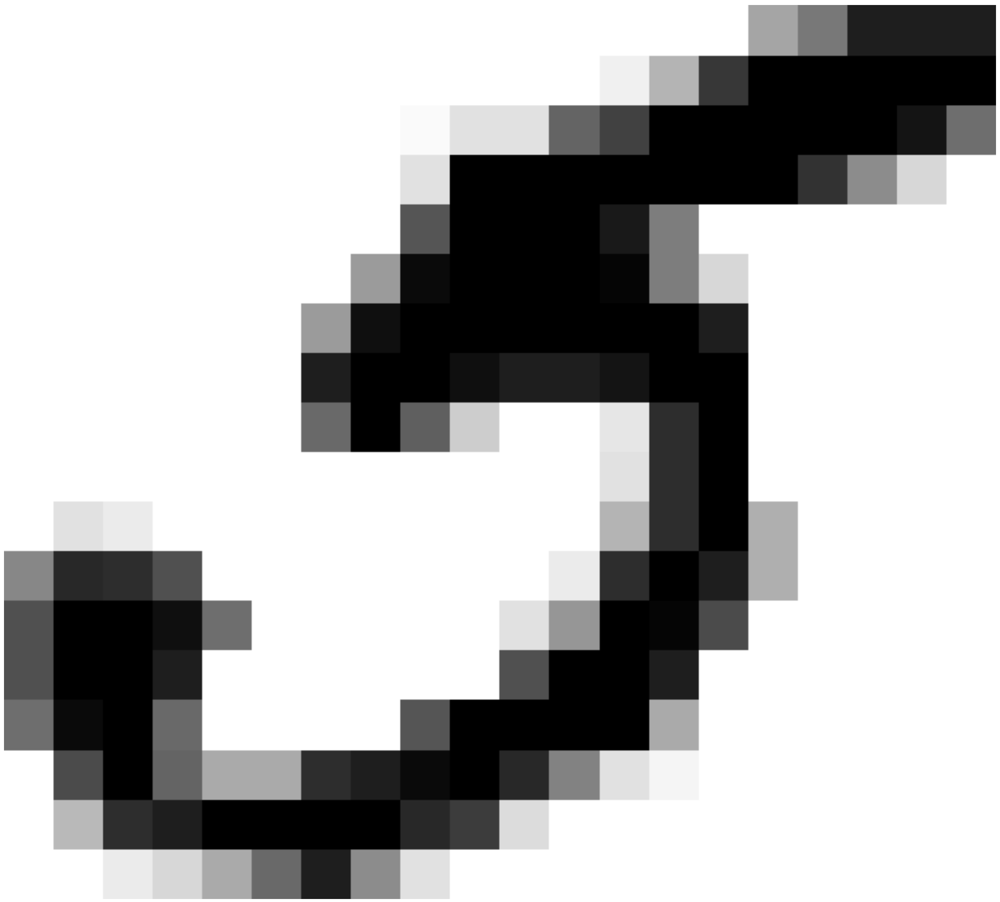
Hand Written Digit Recognition

MNIST dataset, which is 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.

This set has been studied so much that it is often called the “Hello World” of Machine Learning:

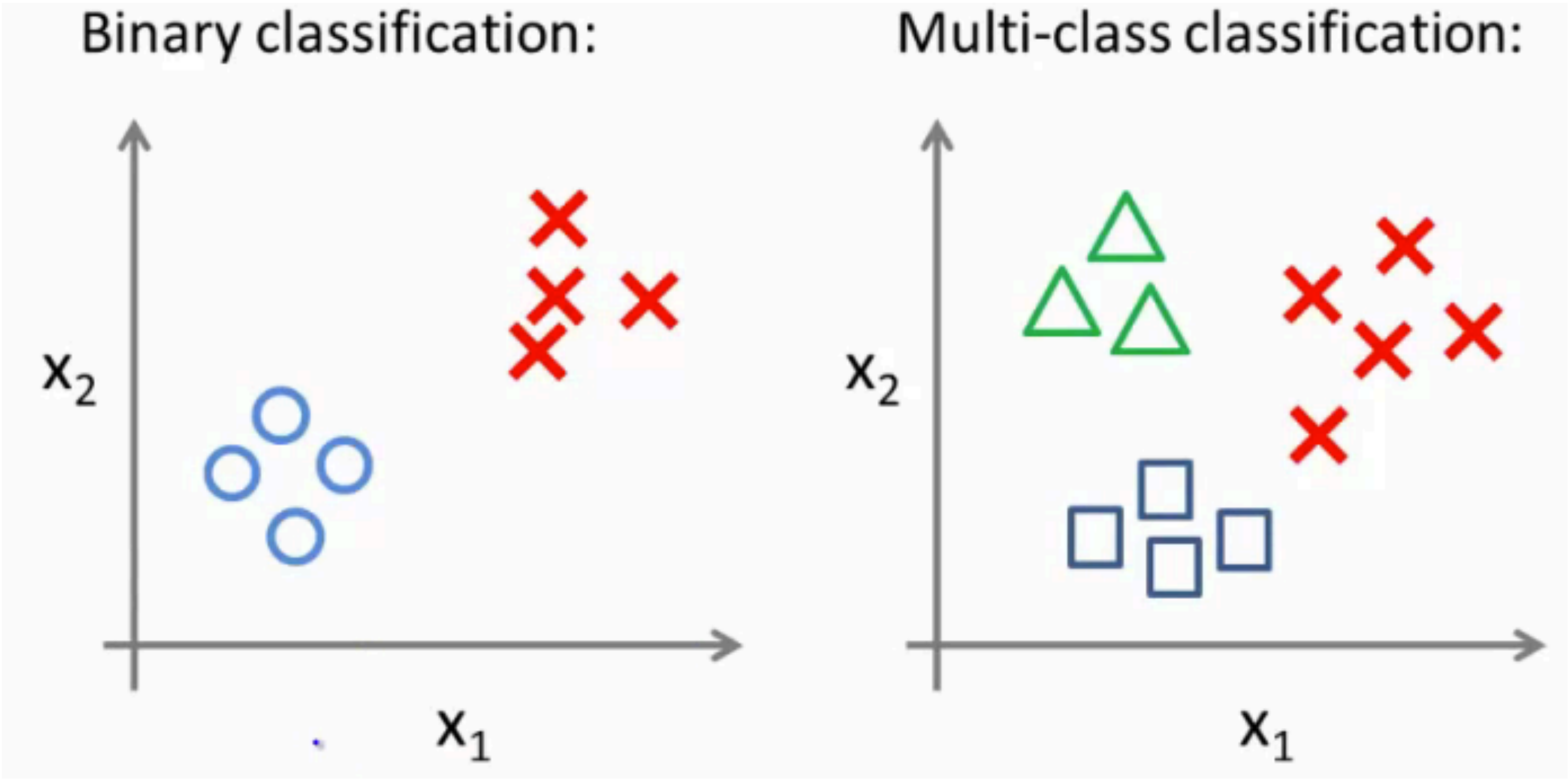
Hand Written Digit Recognition



There are 70,000 images, and each image has 784 features.

This is because each image is 28×28 pixels, and each feature simply represents one pixel's intensity, from 0 (white) to 255 (black).

Binary Classification



5, Not-5 Classification

Split data

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

Shuffle data

```
shuffle_index = np.random.permutation(60000)
X_train, y_train = X_train[shuffle_index], y_train[shuffle_index]
```

Set Labels

```
y_train_5 = (y_train == 5)  # True for all 5s, False for all other digits.
y_test_5 = (y_test == 5)
```

5, Not-5 Classification

Train Classifier

```
from sklearn.linear_model import SGDClassifier

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

Evaluate Performance

```
>>> from sklearn.model_selection import cross_val_score
>>> cross_val_score(sgd_clf, X_train, y_train_5, cv=3, scoring="accuracy")
array([ 0.9502 ,  0.96565,  0.96495])
```

Never 5 Classifier

Set Output to 0 (Not 5)

```
from sklearn.base import BaseEstimator

class Never5Classifier(BaseEstimator):
    def fit(self, X, y=None):
        pass
    def predict(self, X):
        return np.zeros((len(X), 1), dtype=bool)
```

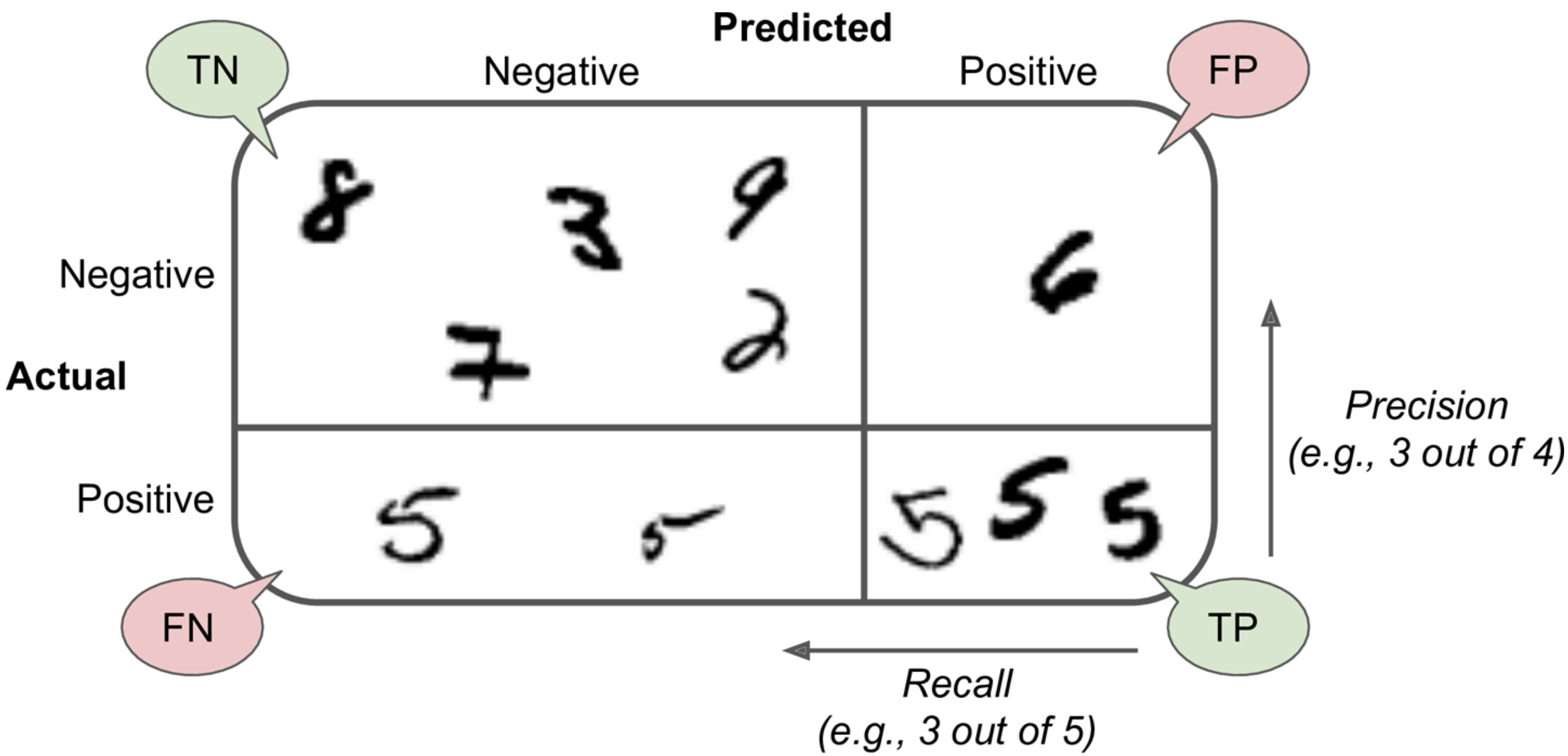
Evaluate Performance

```
>>> never_5_clf = Never5Classifier()
>>> cross_val_score(never_5_clf, X_train, y_train_5, cv=3, scoring="accuracy")
array([ 0.909  ,  0.90715,  0.9128 ])
```

Binary Classification

		True Condition	
	Total Population	Condition Positive	Condition Negative
Test Outcome	Test Outcome Positive	True Positive	False Positve (Type I error)
	Test Outcome Negative	False Negative (Type II error)	True Negative

Hand Written Digit Recognition



Hand Written Digit Recognition

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

the accuracy of the positive predictions

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

the ratio of positive instances that are correctly detected by the classifier

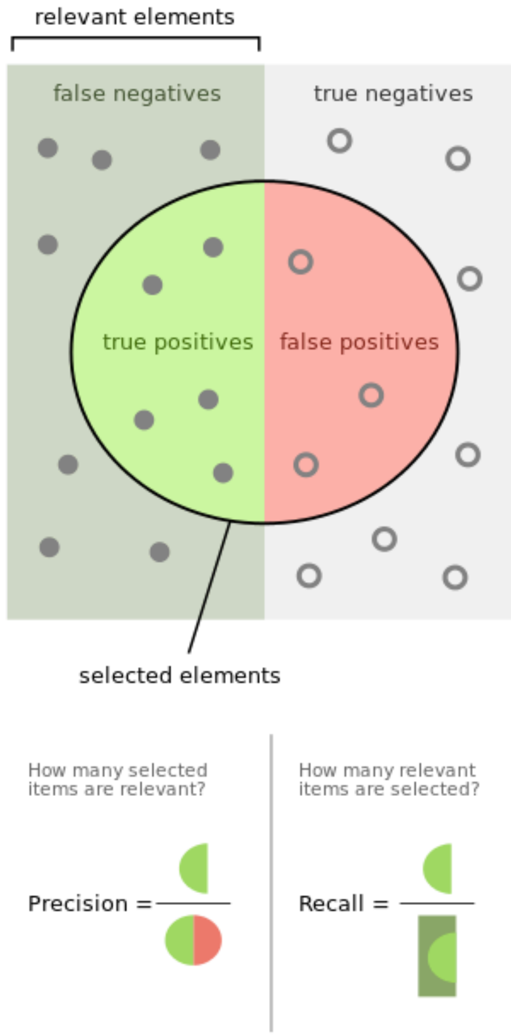
$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{TP}{TP + \frac{FN + FP}{2}}$$

harmonic mean of precision and recall

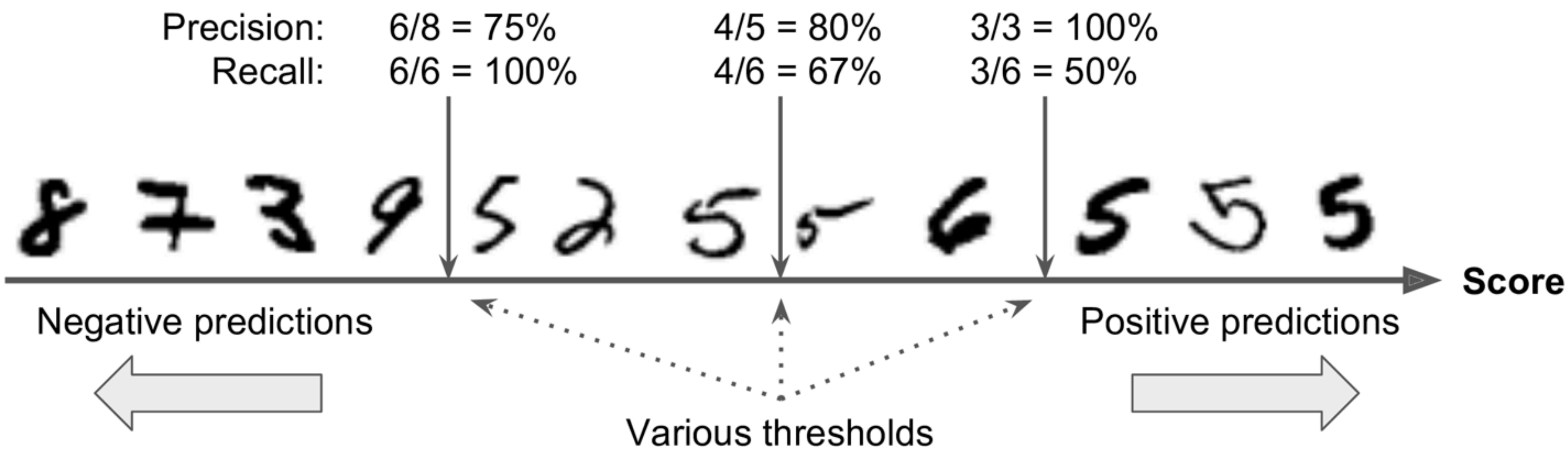
Hand Written Digit Recognition

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

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



Precision Recall Tradeoff

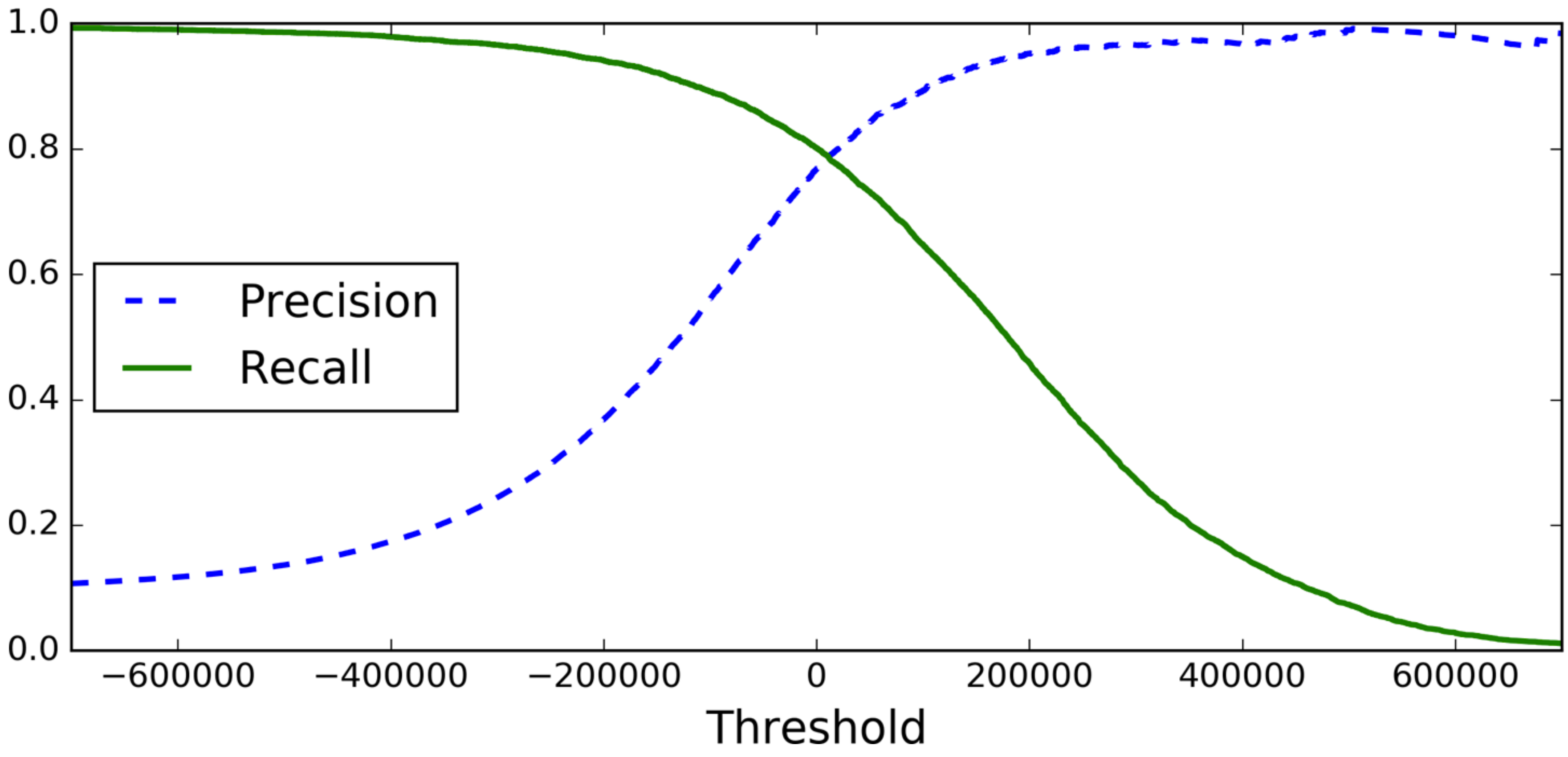


For each instance, the classifier computes a score based on a decision function.

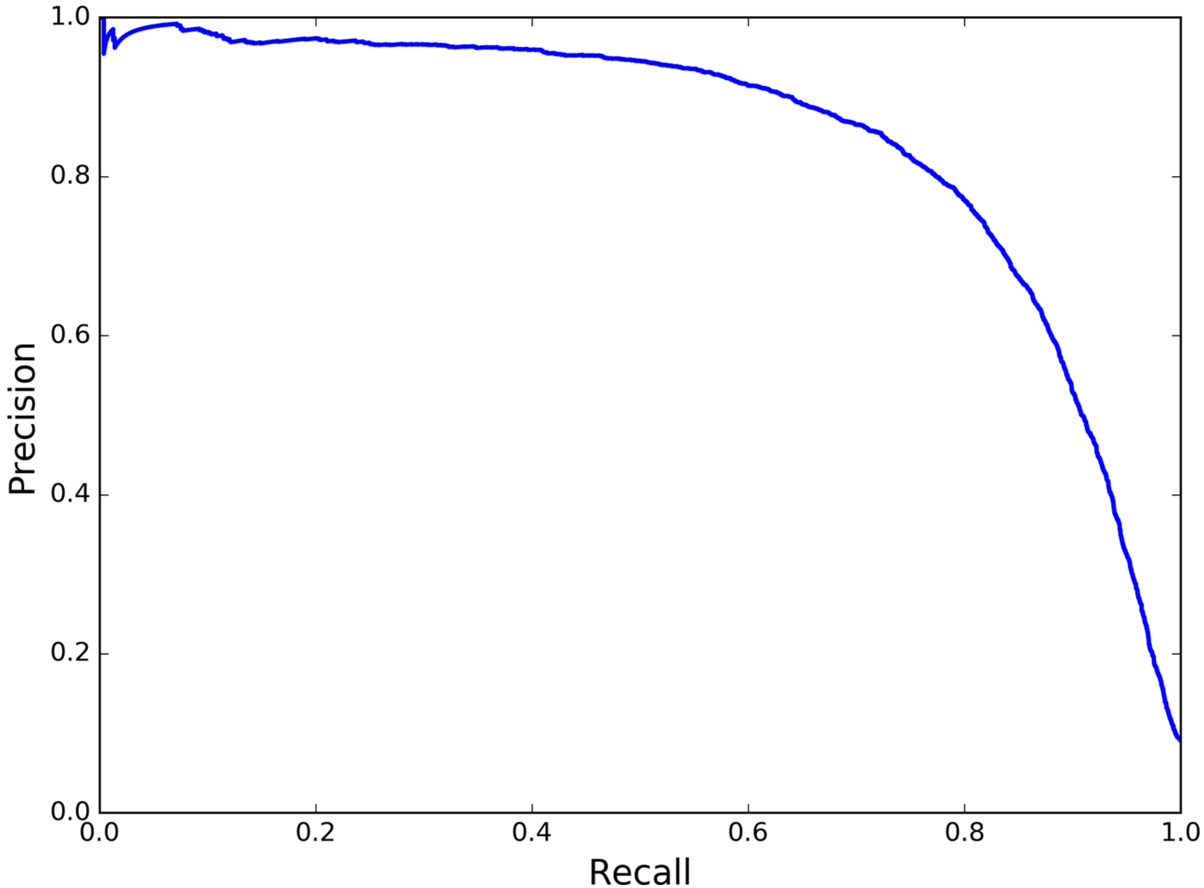
If that score is greater than a threshold, it assigns the instance to the positive class, or else it assigns it to the negative class.

Precision Recall Tradeoff

```python
y_scores = cross_val_predict(sgd_clf, X_train, y_train_5, cv=3,
                             method="decision_function")
```

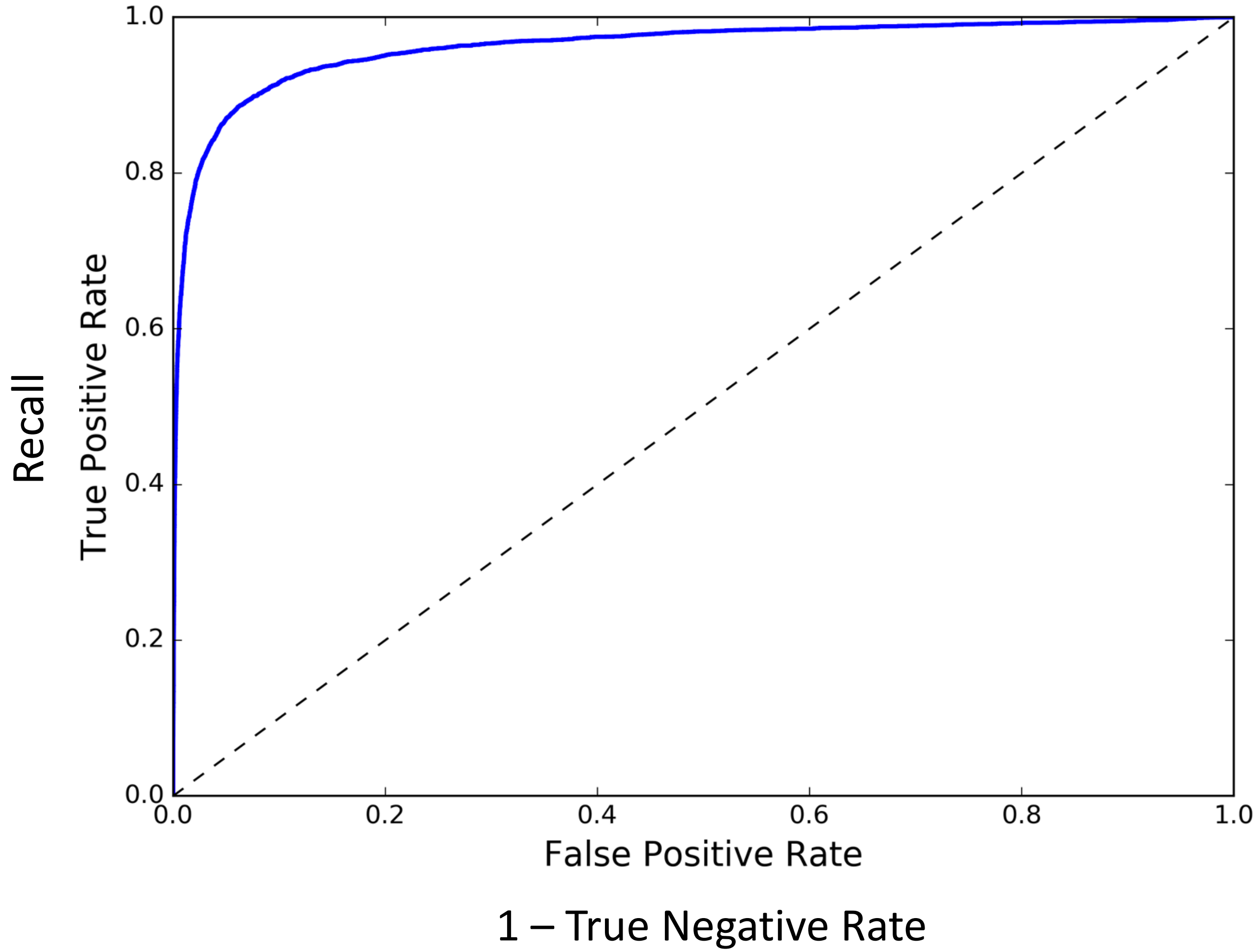


Precision Recall Tradeoff



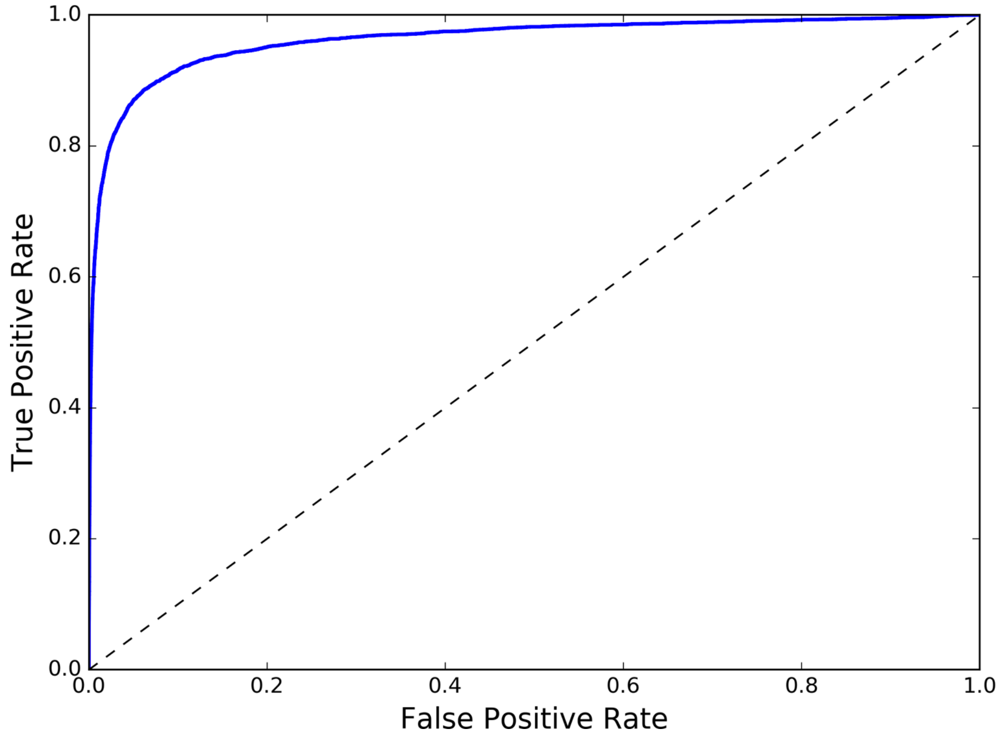
If someone says “let’s reach 99% precision,” you should ask, “at what recall?”

ROC Curve



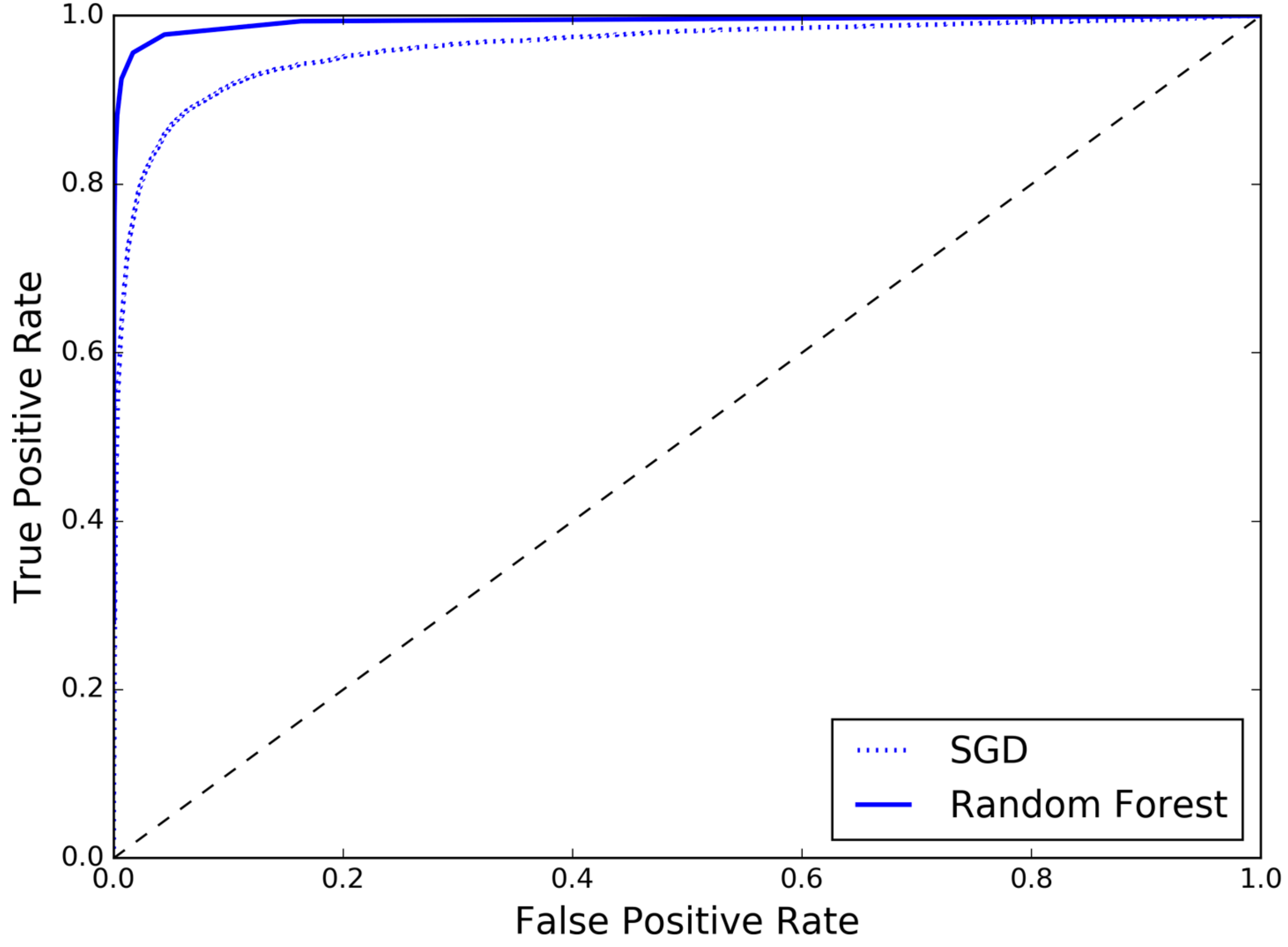
Recall
True Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate
1 – True Negative Rate

ROC Curve



- 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.
- A purely random classifier will have a ROC AUC equal to 0.5.
- Scikit-Learn provides a function to compute the ROC AUC:

Comparing ROC Curves



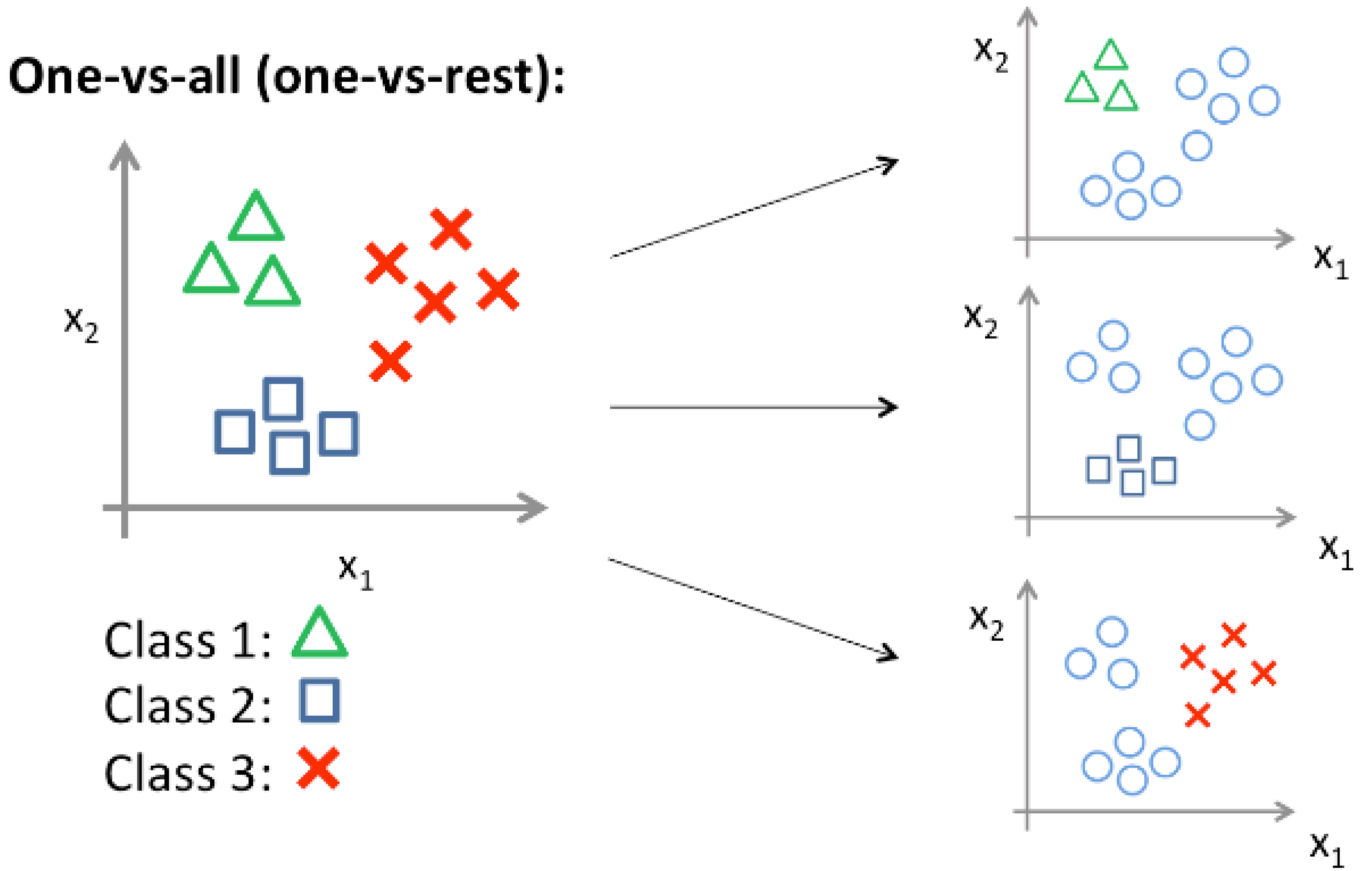
Multi-class Classification

Distinguish between two or more classes

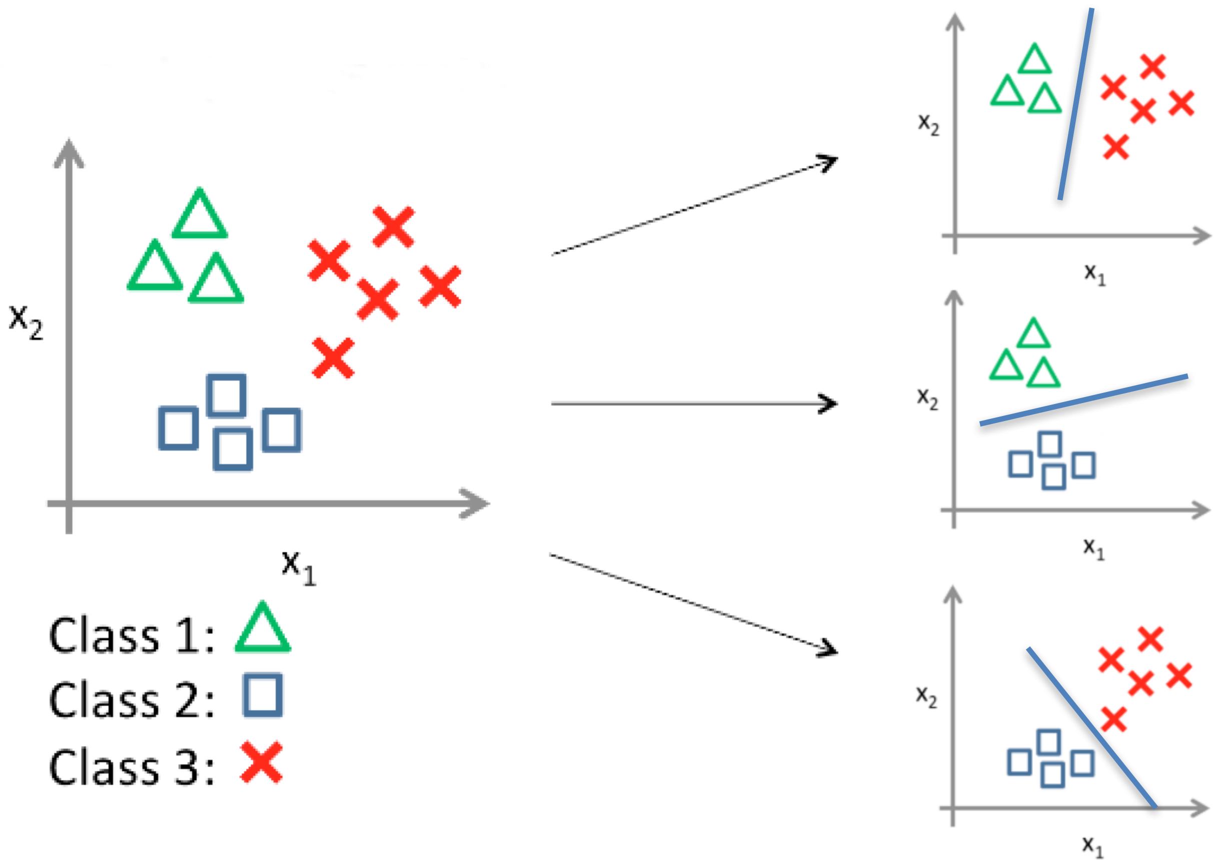
One way to create a system that can classify n-classes is to train n-binary classifiers, one for each class. Then when you want to classify a new instance, you get the decision score from each classifier and select the class with the highest score. (one-versus-all (OvA) strategy)

Another way, is to train a binary classifier for every pair of classes. This is called the one-versus-one (OvO) strategy. If there are N classes, you need to train N × (N – 1) / 2 classifiers.

One-vs-all classifier



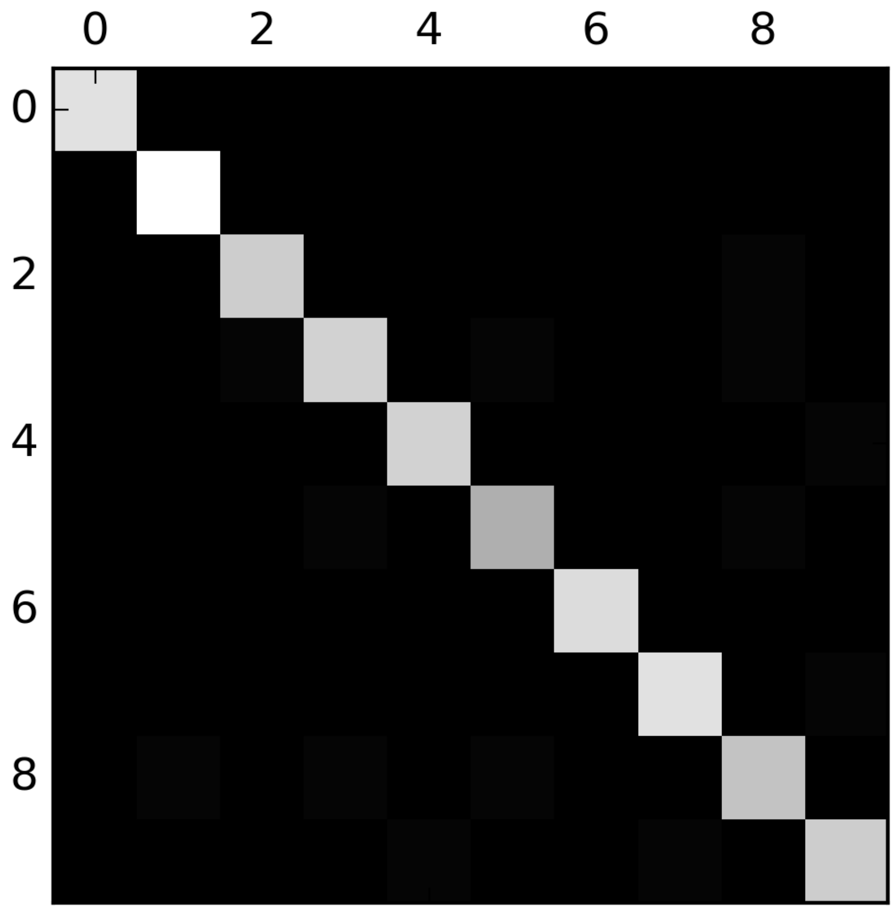
One-vs-One Classifier



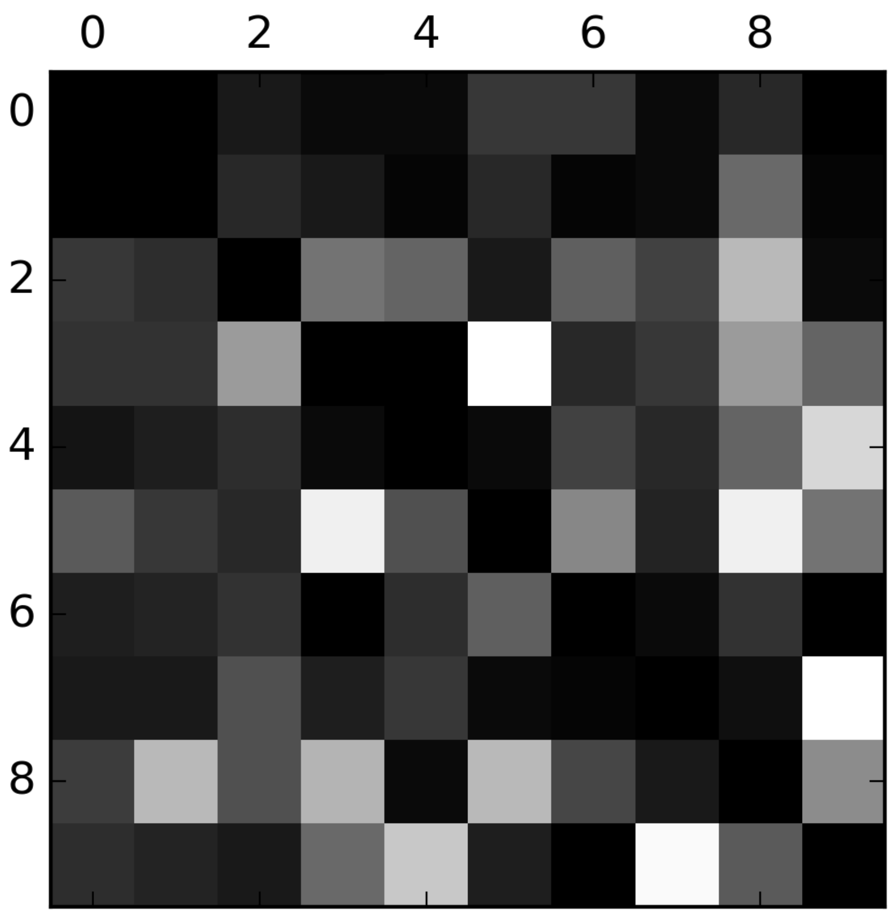
Multi-Dimentional Confusion Matrix

```
>>> y_train_pred = cross_val_predict(sgd_clf, X_train_scaled, y_train, cv=3)
>>> conf_mx = confusion_matrix(y_train, y_train_pred)
>>> conf_mx
array([[5725,    3,   24,    9,   10,   49,   50,   10,   39,    4],
       [   2, 6493,   43,   25,    7,   40,    5,   10,  109,    8],
       [  51,   41, 5321,  104,   89,   26,   87,   60,  166,   13],
       [  47,   46,  141, 5342,    1,  231,   40,   50,  141,   92],
       [  19,   29,   41,   10, 5366,    9,   56,   37,   86,  189],
       [  73,   45,   36,  193,   64, 4582,  111,   30,  193,   94],
       [  29,   34,   44,    2,   42,   85, 5627,   10,   45,    0],
       [  25,   24,   74,   32,   54,   12,    6, 5787,   15,  236],
       [  52,  161,   73,  156,   10,  163,   61,   25, 5027,  123],
       [  43,   35,   26,   92,  178,   28,    2,  223,   82, 5240]])
```

Multi-Dimentional Confusion Matrix

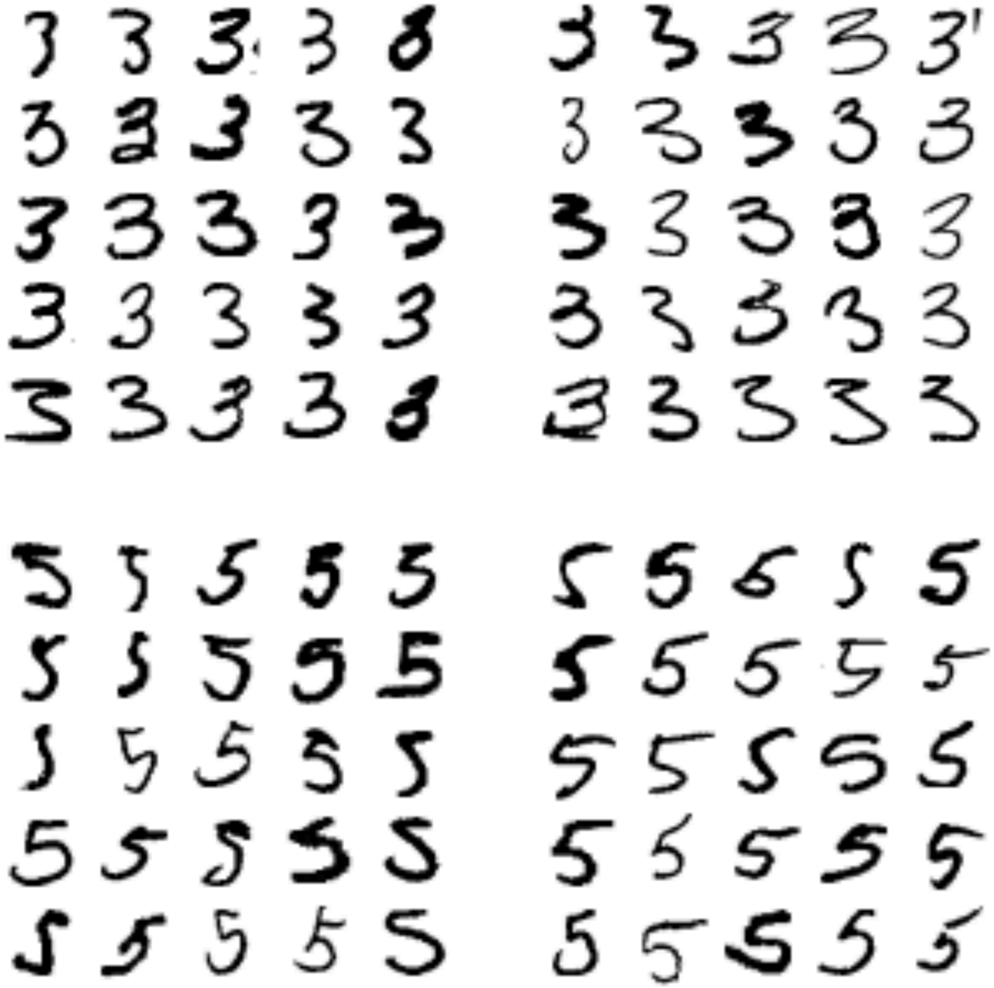


Multi-Dimentional Error Analysis



divide each value in the confusion matrix by the number of images in the corresponding class

Multi-Dimentional Error Analysis



Questions?

Homework

- Complete the notebook in the assignments section for this week
- Submit your solution here
  - https://goo.gl/forms/F5ytppo5KWnCqkt62
  - Rename your notebook to
    - W3_LastName_UTORid.ipynb
    - Example W3_Benitez_q212131.ipynb

Next Class

- Unsupervised methods
- Clustering