



# **3253 - Analytic Techniques and Machine Learning**

## **Module 3: Classification**



# Course Plan

Module Titles
Module 1 – Introduction to Machine Learning
Module 2 – End to End Machine Learning Project
<b>Module 3 – Current Focus: Classification</b>
Module 4 – Clustering and Unsupervised Learning
Module 5 – Training Models and Feature Selection
Module 6 – Support Vector Machines
Module 7 – Decision Trees and Ensemble Learning
Module 8 – Dimensionality Reduction
Module 9 – Introduction to TensorFlow
Module 10 – Introduction to Deep Learning and Deep Neural Networks
Module 11 – Distributing TensorFlow, CNNs and RNNs
Module 12 – Final Assignment and Presentations (no content)



# Learning Outcomes for this Module

- Develop experience with binary classification
- Use performance measures to evaluate classifiers
- Extend the techniques to multiclass classification



# Topics for this Module

- **3.1** The MNIST dataset
- **3.2** Binary classification
- **3.3** Precision and recall
- **3.4** ROC curves
- **3.5** Multi-class classification
- **3.6** Evaluating classifiers
- **3.7** Resources and Wrap-up



## Module 3 – Section 1

# The MNIST Dataset

# Hand Written Digit Recognition

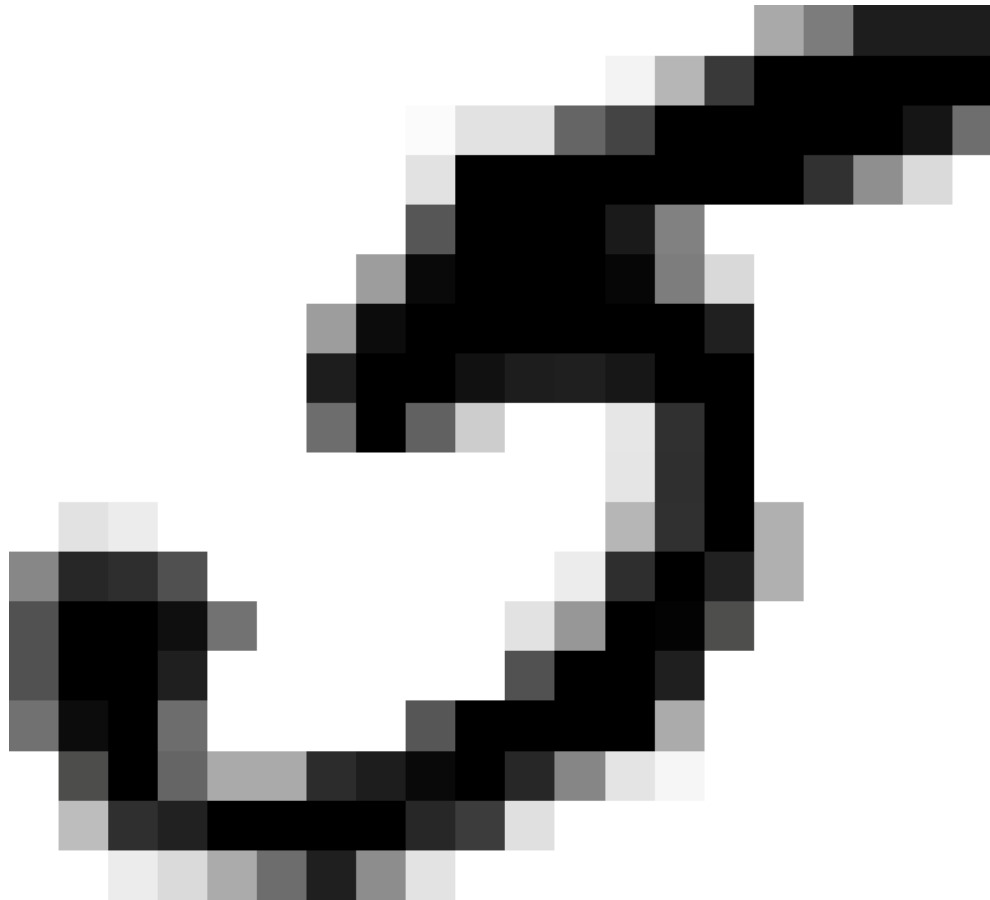


MNIST dataset is a set of 70,000 small images of digits handwritten by high school students and employees of the US Census Bureau.

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 (cont'd)



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

This is because each image is 28x28 pixels, and each feature simply represents one pixel's intensity, from 0 (white) to 255 (black).



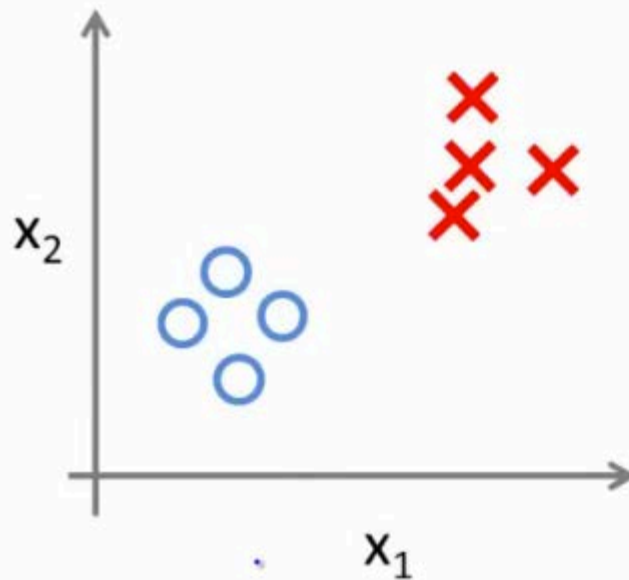
## Module 3 – Section 2

# Binary Classification

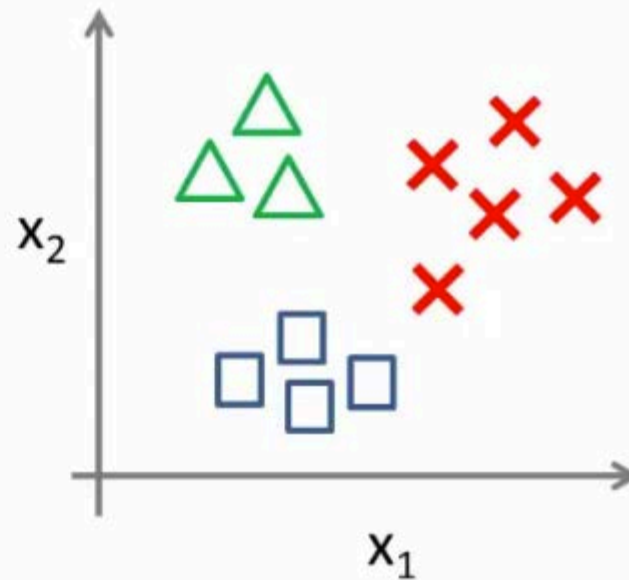


# Binary Classification

Binary classification:



Multi-class 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 (cont'd)

## 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

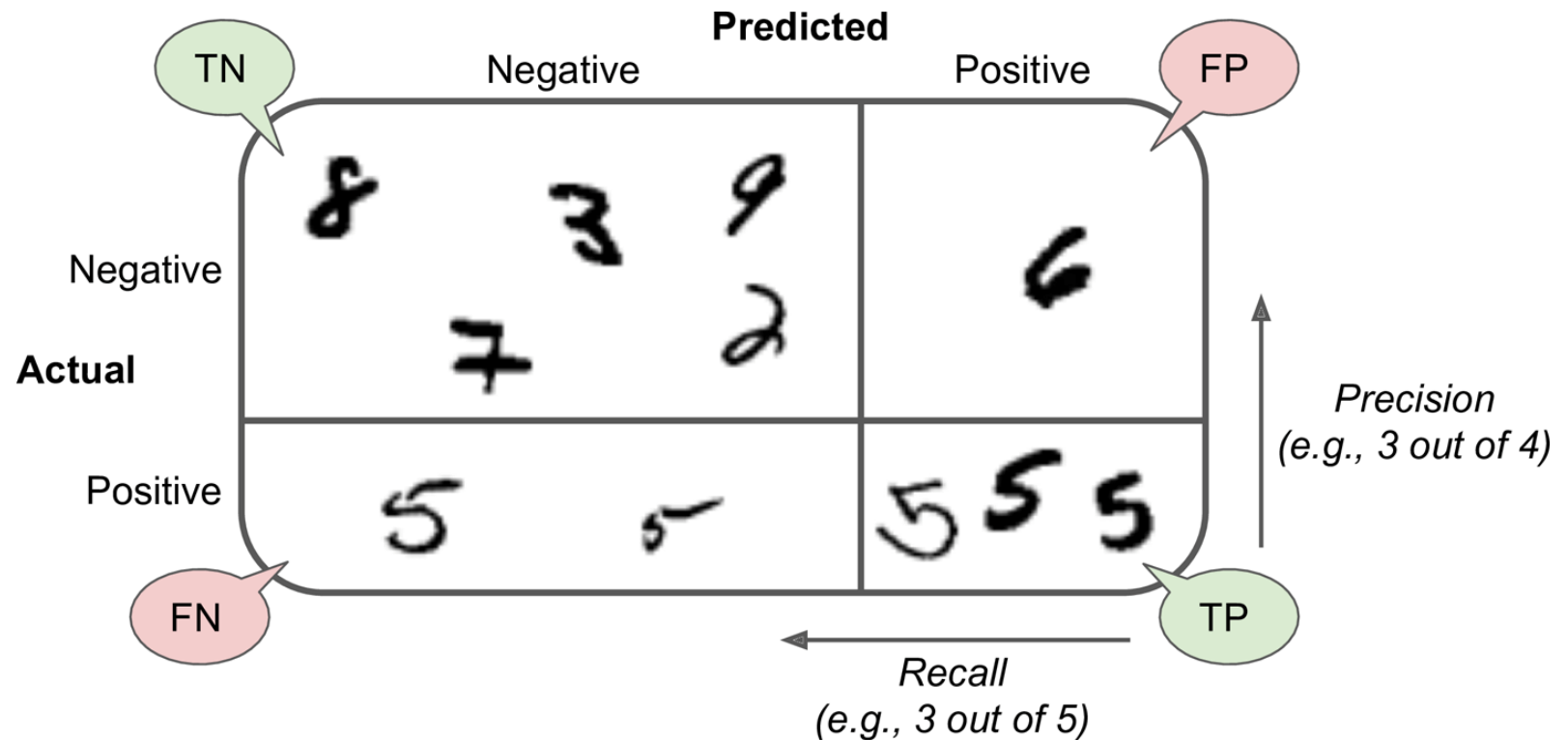
		True Condition	
		Condition Positive	Condition Negative
Test Outcome	Total Population		
	Test Outcome Positive	True Positive	False Positive (Type I error)
	Test Outcome Negative	False Negative (Type II error)	True Negative



## Module 3 – Section 3

# Precision and Recall

# Hand Written Digit Recognition



# Hand Written Digit Recognition (cont'd)

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

the accuracy of the positive predictions

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

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

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

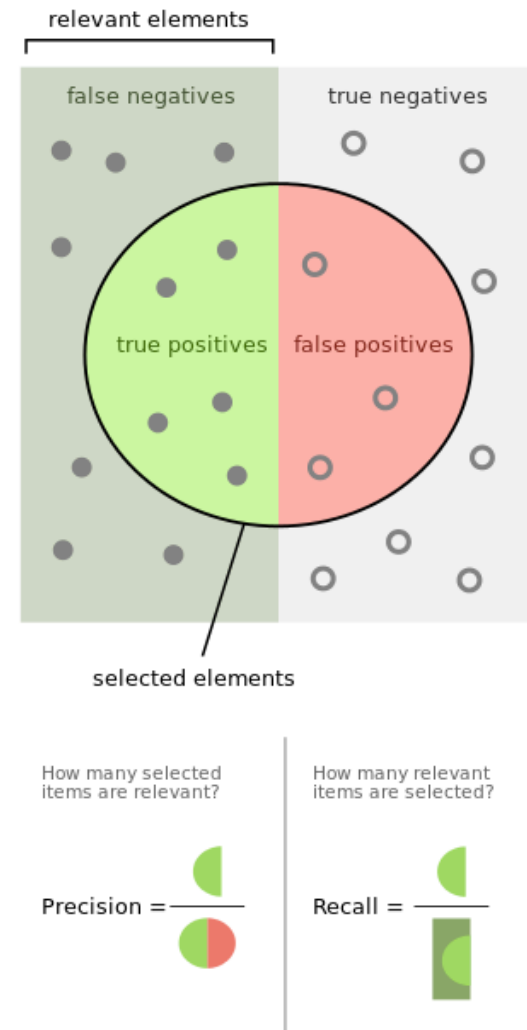
harmonic mean of precision and recall



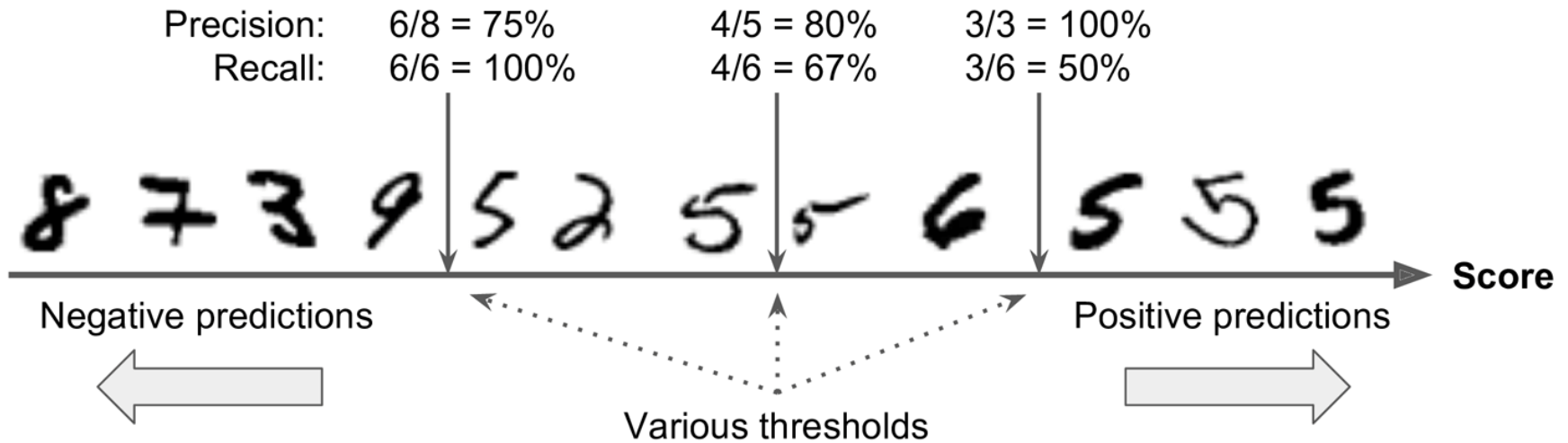
# Hand Written Digit Recognition (cont'd)

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

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



# Precision Recall Tradeoff

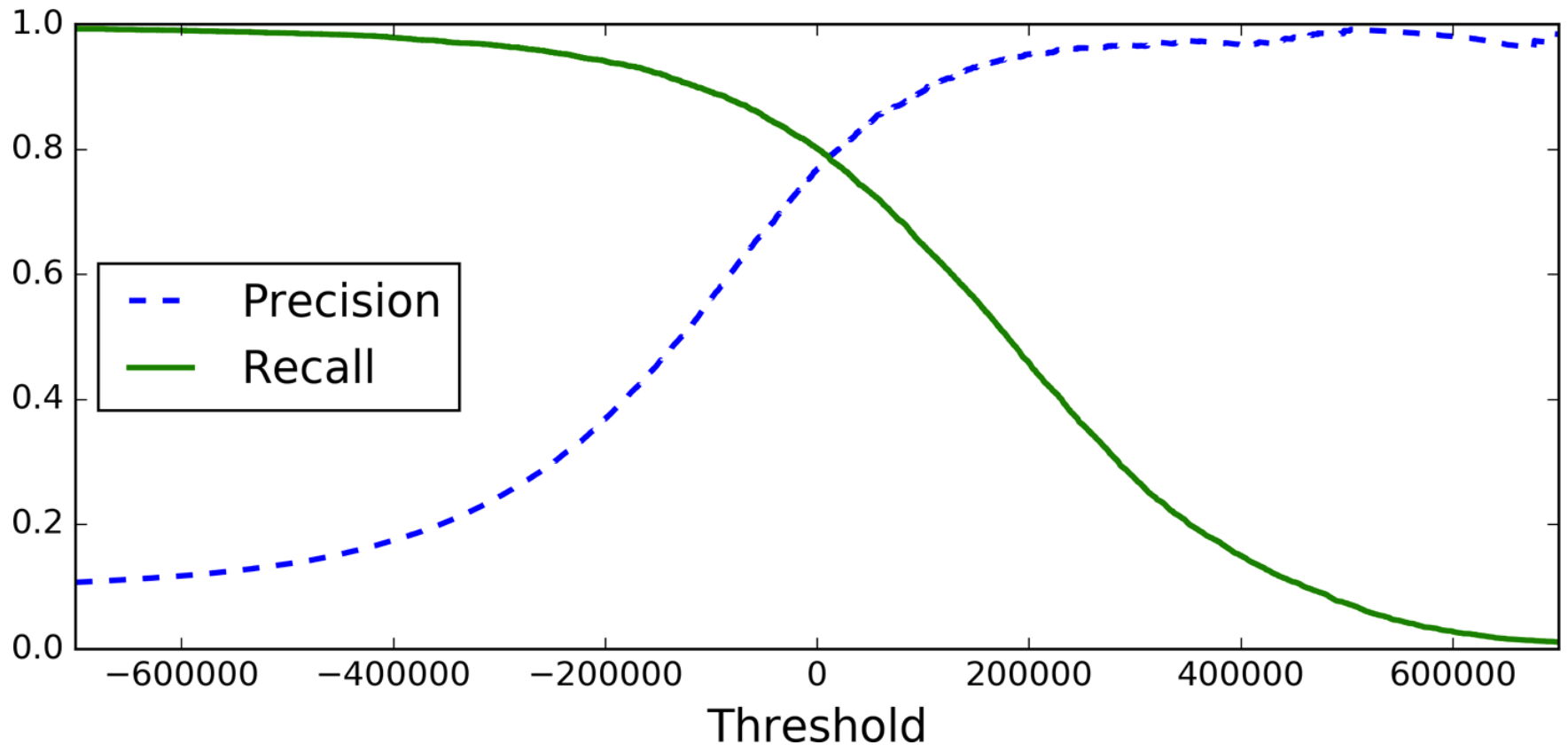


For each instance, the classifier computes a score based on a decision function.

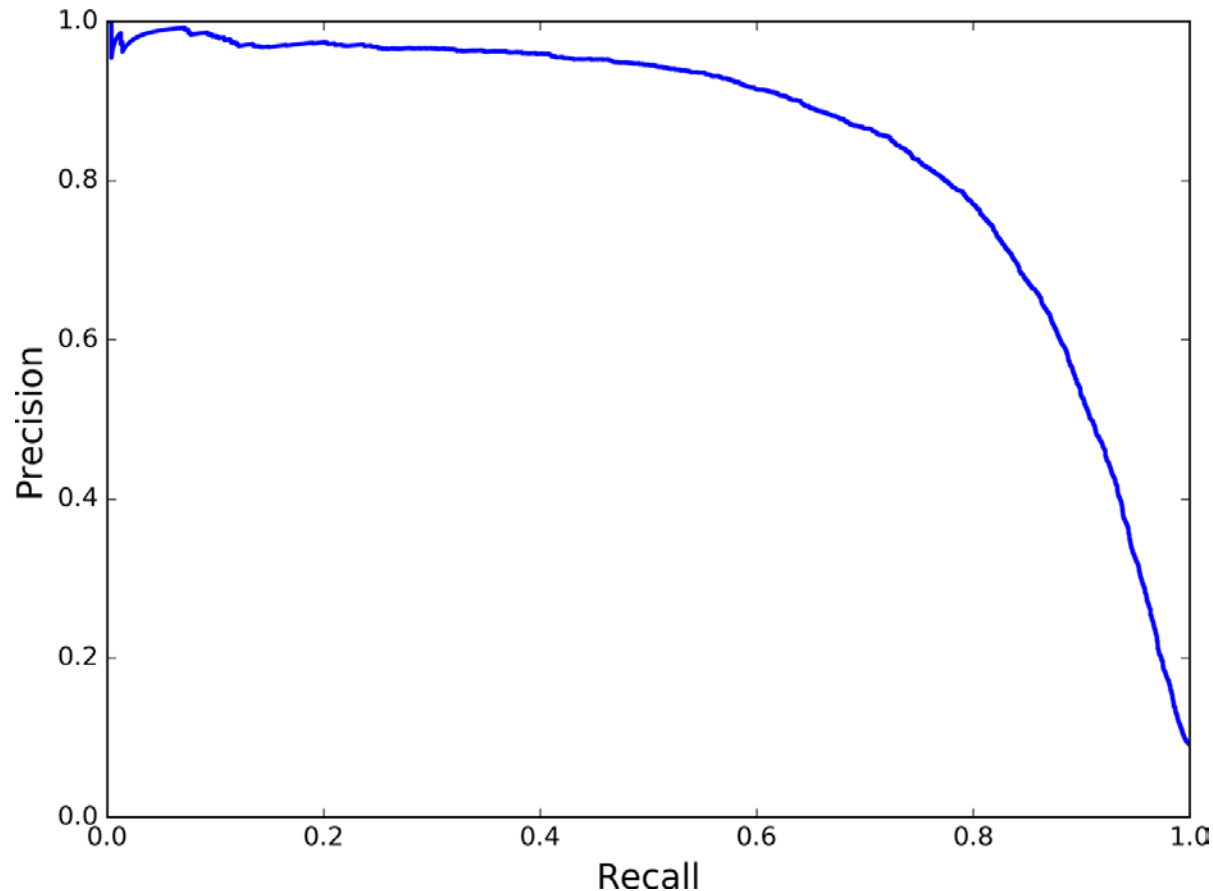
If that score is greater than a threshold, it assigns the instance to the positive class, or else it assigns it to the negative class.

# Precision Recall Tradeoff (cont'd)

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



# Precision Recall Tradeoff (cont'd)



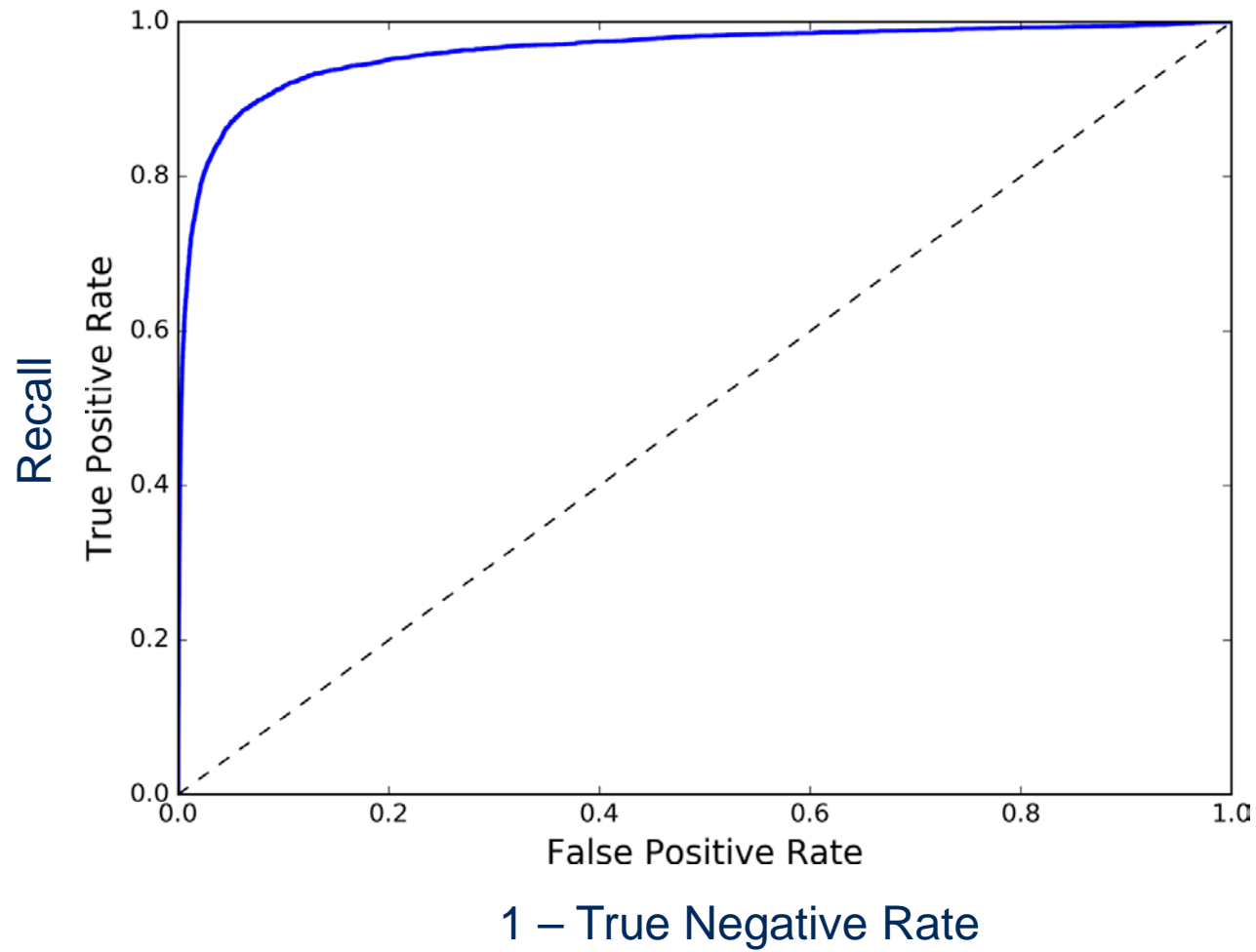
If someone says “let’s reach 99% precision,” you should ask, “at what recall?”



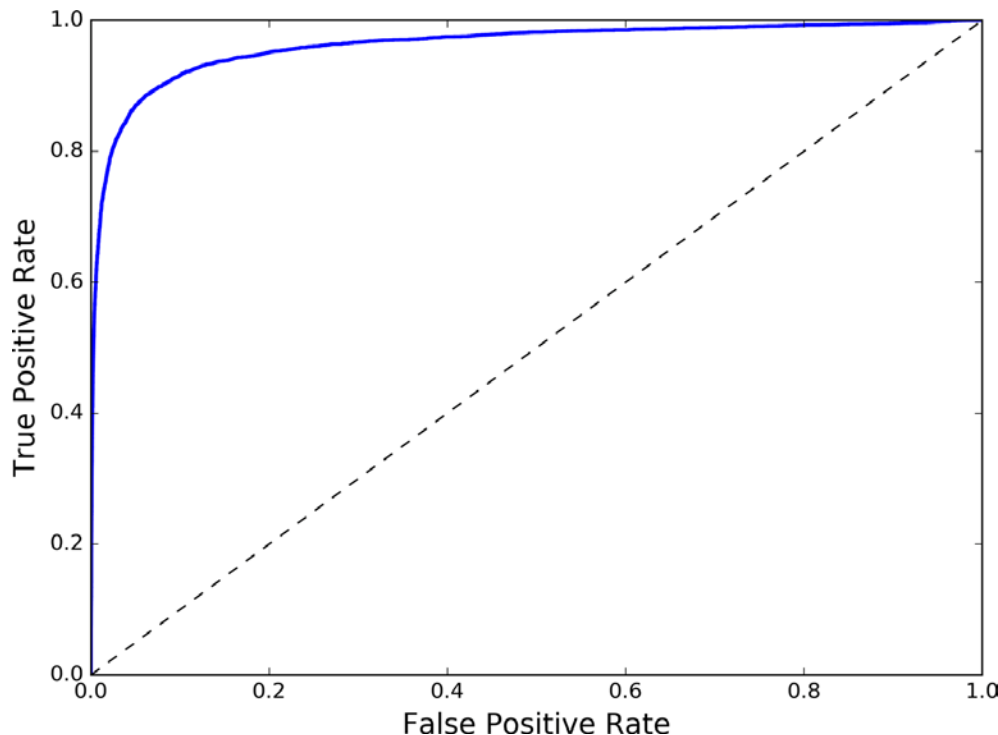
## Module 3 – Section 4

# ROC Curves

# ROC Curve

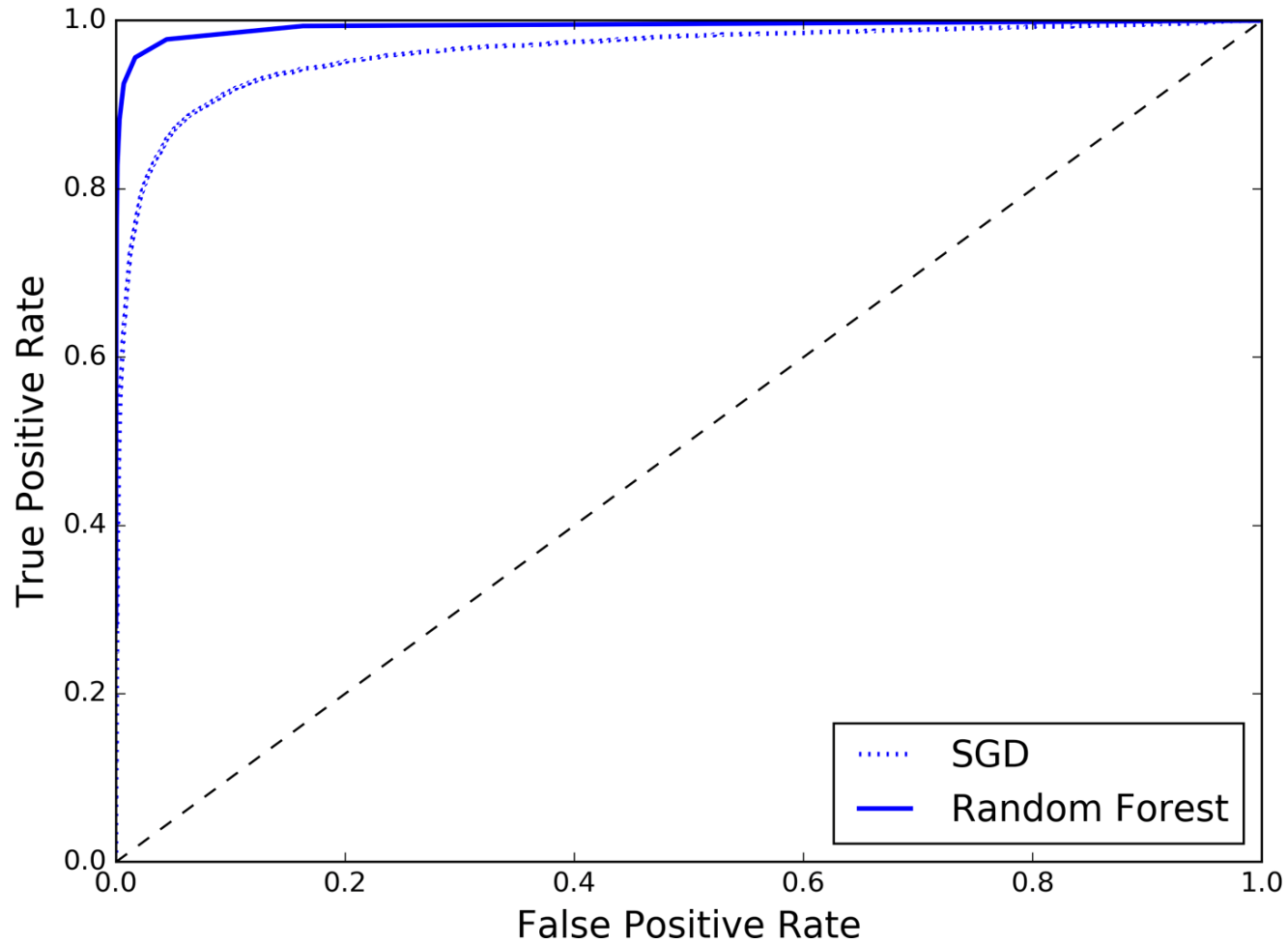


# ROC Curve (cont'd)



- 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







## Module 3 – Section 5

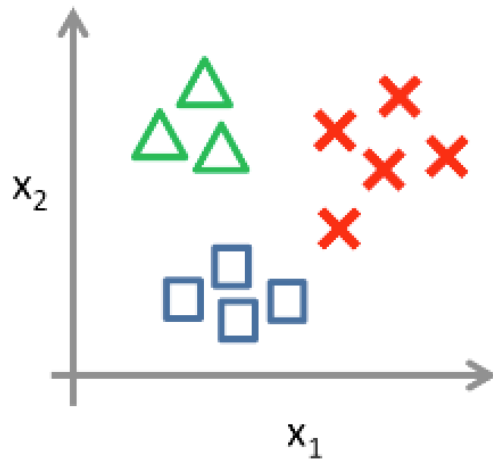
# Multi-class Classification


# 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  $N \times (N - 1) / 2$  classifiers.


# One-vs-all Classifier

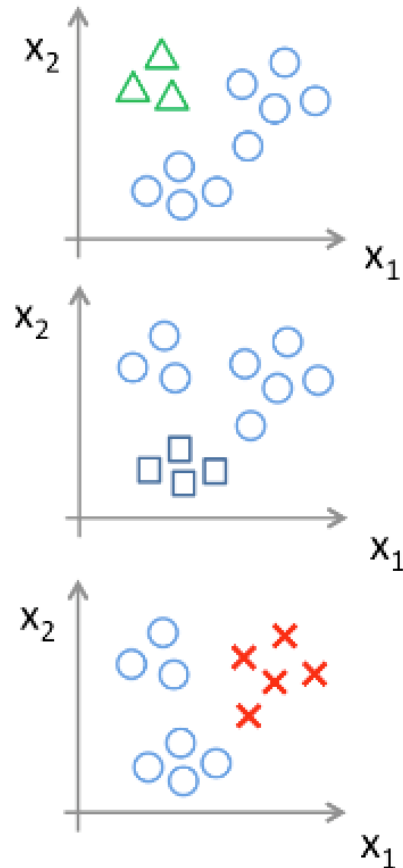
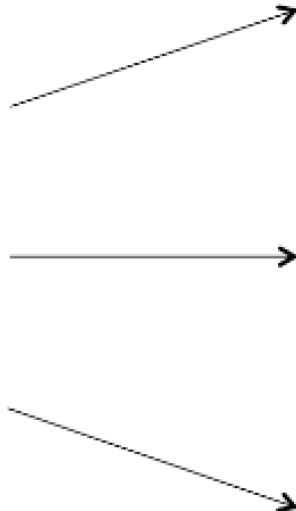
One-vs-all (one-vs-rest):



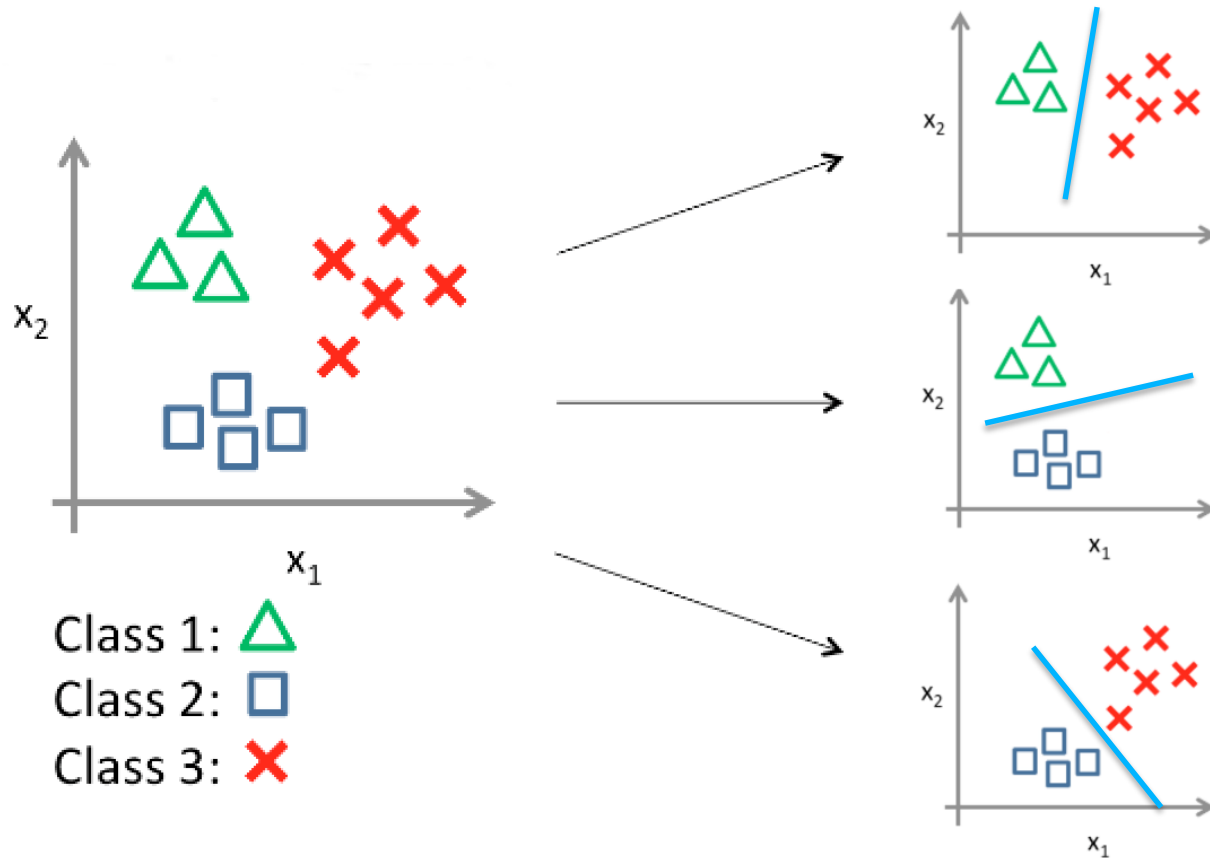
Class 1: 

Class 2: 

Class 3: 



# One-vs-one Classifier





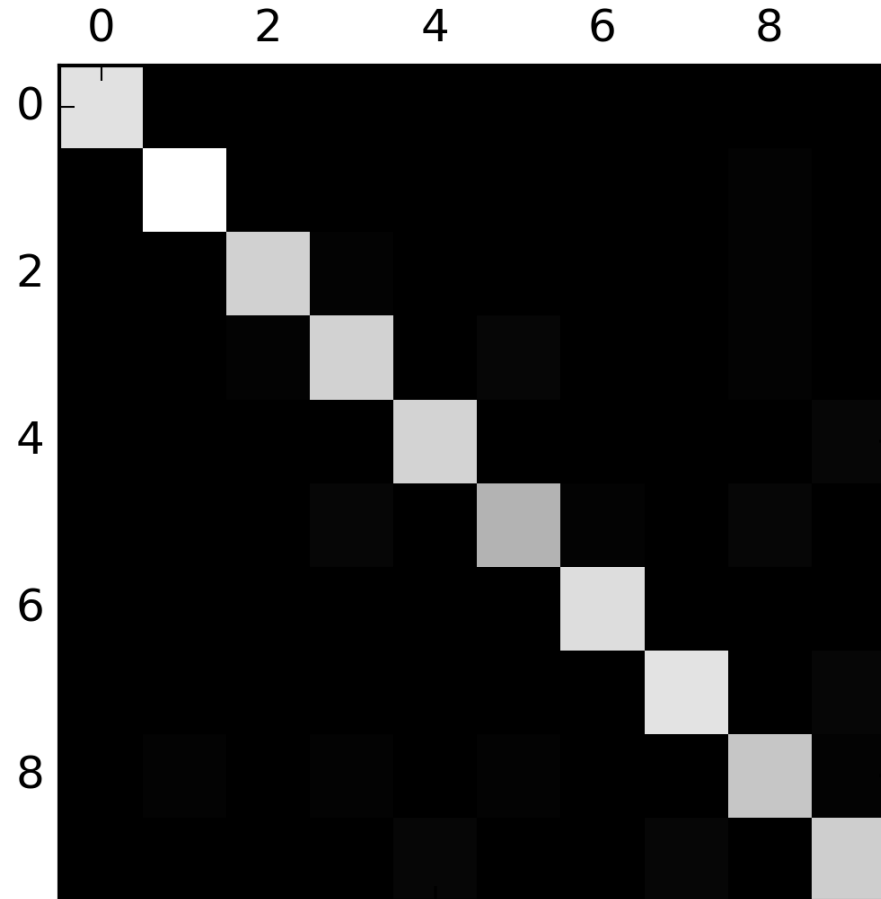
## Module 3 – Section 6

# Evaluating Classifiers

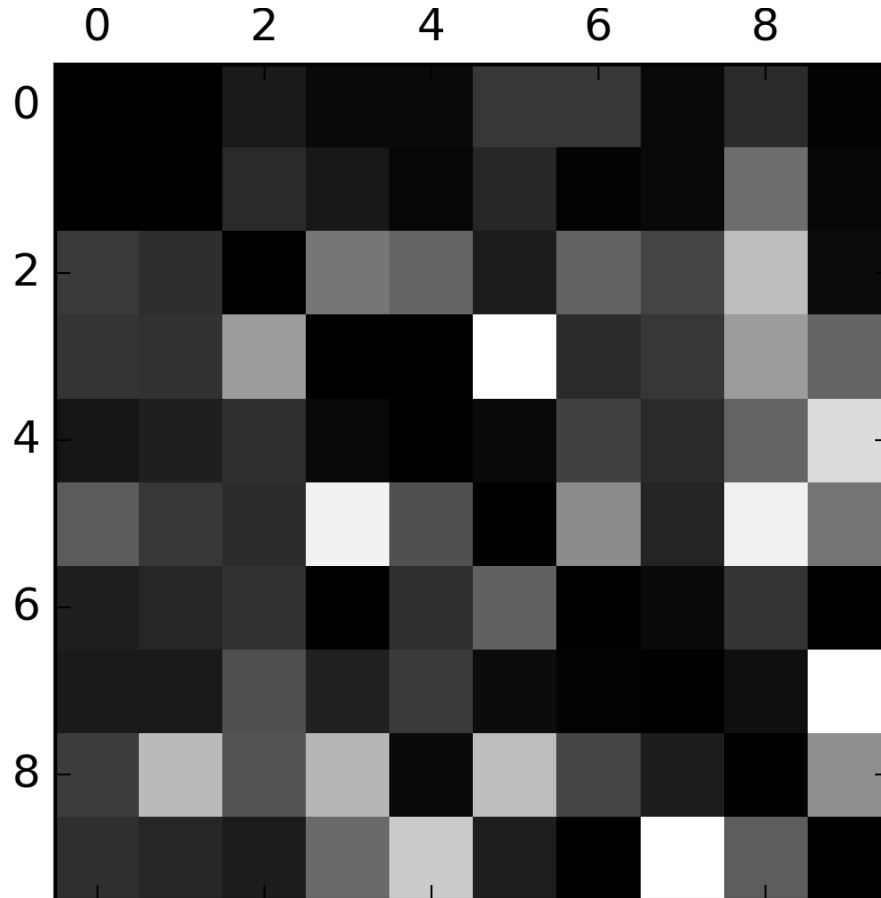
# Multi-Dimensional Confusion Matrix

```
>>> 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-Dimensional Confusion Matrix (cont'd)



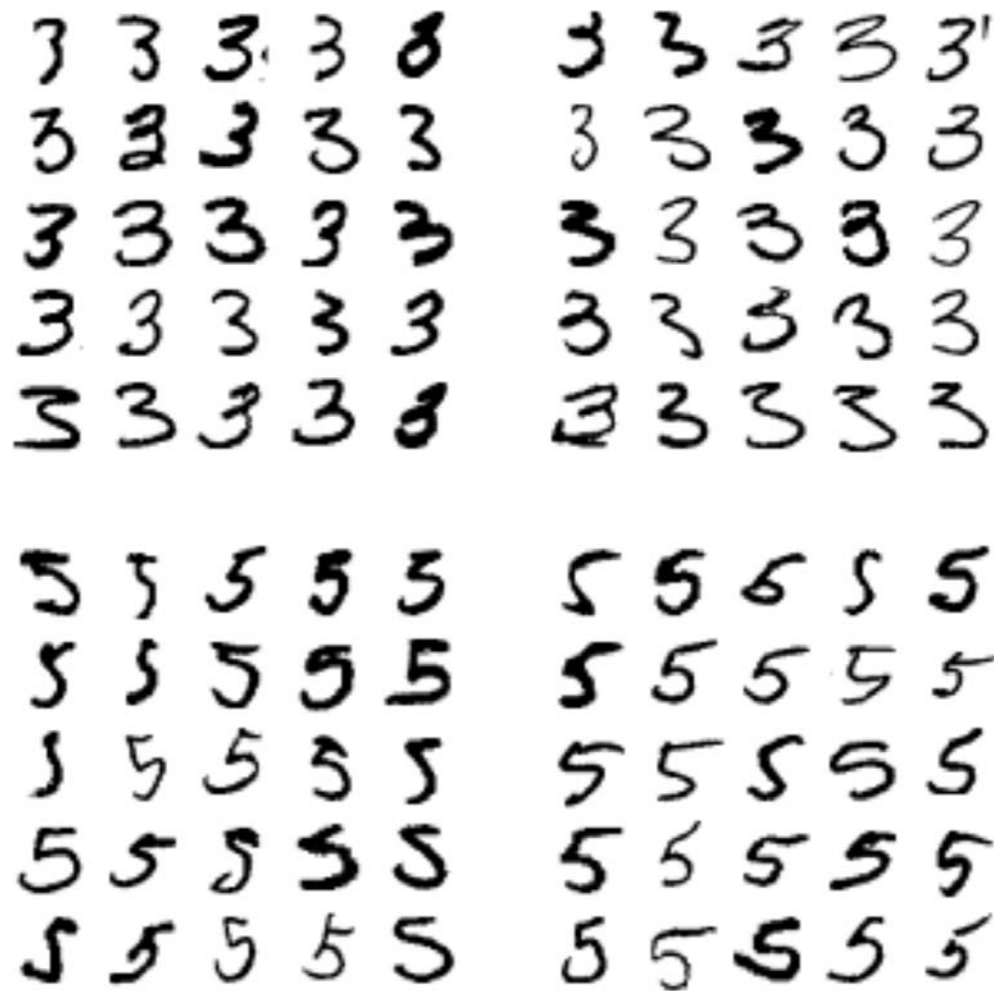
# Multi-Dimensional Error Analysis



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



# Multi-Dimensional Error Analysis (cont'd)





## Module 3 – Section 7

# Resources and Wrap-up

# Homework

- Complete the notebook in the assignments section for this week
- Submit your solution [here](#)
- Make sure you rename your notebook to
  - W3\_UTORid.ipynb
  - Example: W3\_adfasd01.ipynb

# Next Class

- Unsupervised methods
- Clustering

# Follow us on social

Join the conversation with us online:

 [facebook.com/uoftscs](https://facebook.com/uoftscs)

 [@uoftscs](https://twitter.com/uoftscs)

 [linkedin.com/company/university-of-toronto-school-of-continuing-studies](https://linkedin.com/company/university-of-toronto-school-of-continuing-studies)

 [@uoftscs](https://instagram.com/uoftscs)



**Any questions?**



# Thank You

Thank you for choosing the University of Toronto  
School of Continuing Studies