Evaluating Classifiers

Machine Learning

PHYS 453

Dr. Daugherity

ABILENE CHRISTIAN UNIVERSITY

Big Picture

Typical steps in ML project:

- **Getting Data** – in the real world cleaning the data can be really difficult
- **Preprocessing** – getting data ready to use, only part we've discussed so far is test/train split
- **Training** – choosing models and parameters, "fitting" them to data
- **Evaluating** – reporting how well it works
- **Predicting** – now that you've got a working thing, go out and use it!

https://www.youtube.com/watch?v=--szrOHtR6U

Evaluating Classifiers

- How can I measure how well a classifier works?
- Where do I look for ways to improve performance?
- Note: remember that we evaluate in two different places:
  - with test/train split to check for under- or over-fitting while we are tuning a model
  - using just the test data to evaluate final model

Sources:

- https://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics
- Binary Classification Metrics paper, on canvas or: https://arxiv.org/pdf/1410.5330
- https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html

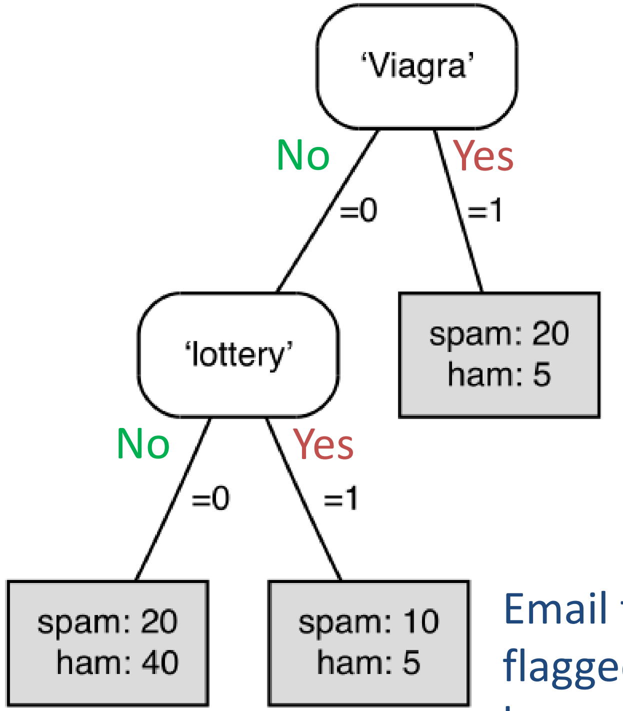
Evaluating Classifiers

CONFUSION MATRIX

Confusion Matrix

		Predicted class	
		P	N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Spam detection



Email that contains word Viagra is flagged as spam. 20 of 25 emails here are actually spam

Email that contains neither is flagged as ham.

Email that contains word lottery is flagged as spam. 10 of 15 emails here are actually spam

		Predicted class	
		P	N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

	SPAM	HAM	
	Predicted ⊕	Predicted ⊖	
Actual ⊕	30	20	50
Actual ⊖	10	40	50
	40	60	100

Chapter 2.1

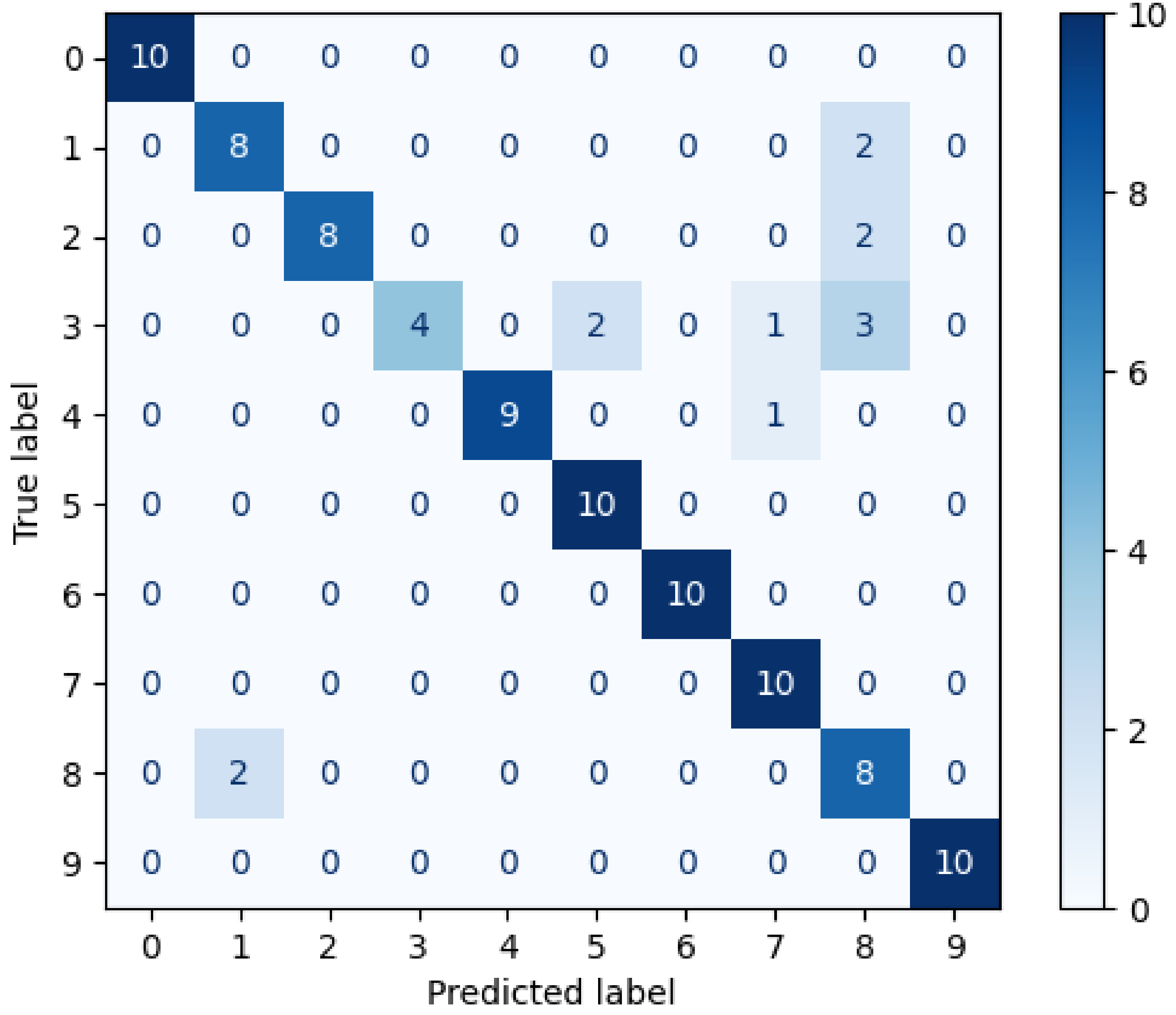
Difficulty: 2

There are 20 dogs(+) and 10 cats(-). A binary classifier correctly predicts 5 dogs and incorrectly predicts 5 cats. Fill in the following contingency for this binary classifier matrix.

	Predicted +	Predicted -	
Actual +			
Actual -			

Multiclass Example: identify hand-written digits 0-9

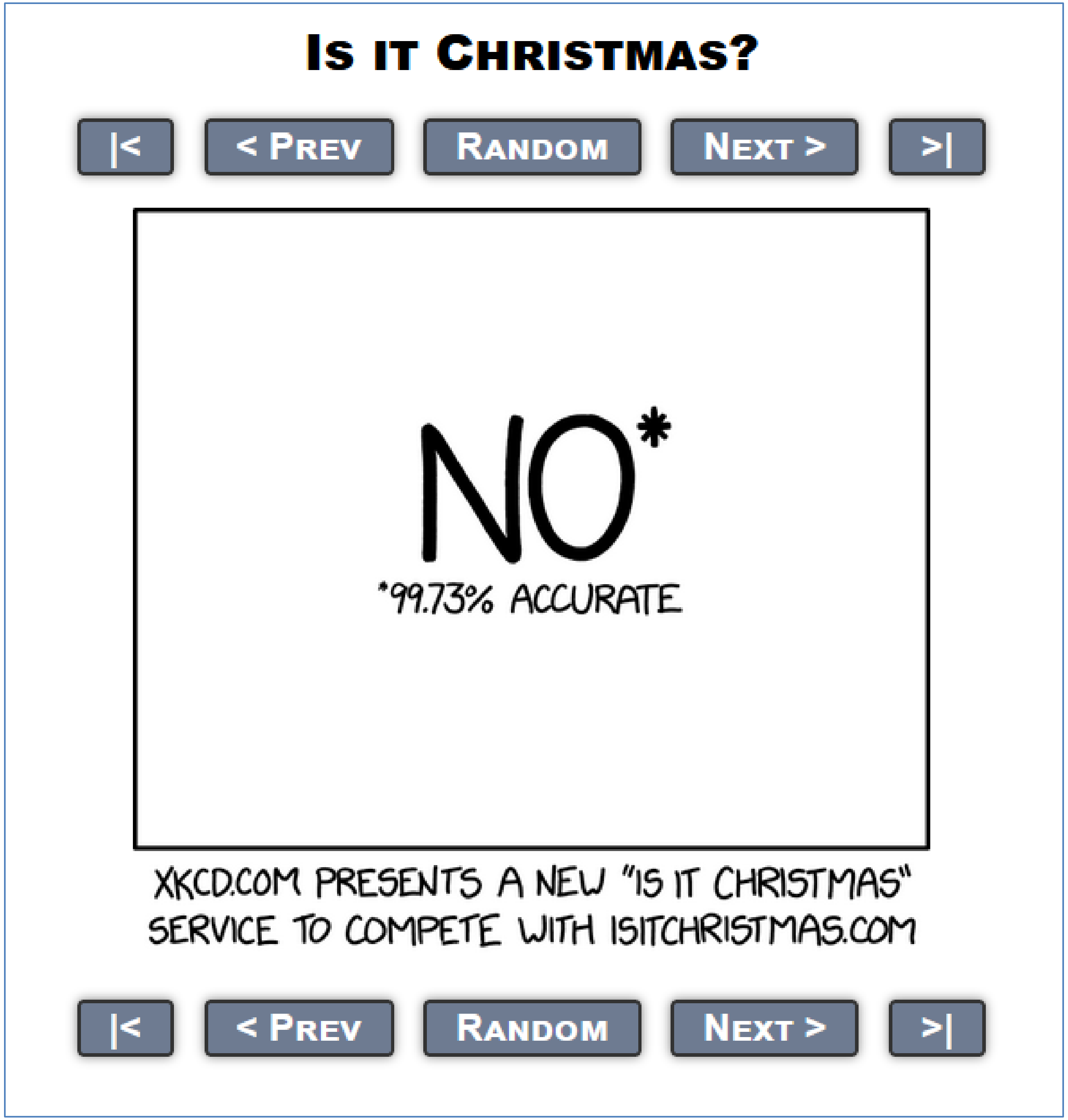
Shows the biggest source of error is calling everything an 8 (especially 3's)



Evaluating Classifiers

CLASSIFIER METRICS

https://xkcd.com/2236/



		Predicted class	
		P	N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Accuracy Metrics

$$ERR = \frac{FP + FN}{FP + FN + TP + TN} = 1 - ACC$$

Error %

$$ACC = \frac{TP + TN}{FP + FN + TP + TN} = 1 - ERR$$

Accuracy %

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN}$$

False Positive Rate = (# of FP) / (# actually N)
"what percentage of the real N did I miss?"

$$TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$

True Positive Rate = (# of TP) / (# actually P)
"what percentage of the real P did I get?"

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

PRECISION = the ability of the classifier not to label as positive a sample that is negative.
Fraction of pos guesses that are right.

$$REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$

RECALL (aka TPR) = the ability of the classifier to find all the positive samples
Fraction of all actual pos we guessed as pos.

$$F_1 = 2 \cdot \frac{PRE \cdot REC}{PRE + REC}$$

F1 Score = combines both into a single number. 1 is perfect.

Metrics

- A system with high recall but low precision returns many results, but most of its predicted labels are incorrect when compared to the training labels.
- A system with high precision but low recall is just the opposite, returning very few results, but most of its predicted labels are correct when compared to the training labels.
- Can plot curve showing precision/recall trade-off

https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html#sphx-glr-auto-examples-model-selection-plot-precision-recall-py

Challenge: gotta find them all!

	Predicted ⊕	Predicted ⊖	
Actual ⊕	30	20	50
Actual ⊖	10	40	50
	40	60	100

		Predicted class	
		P	N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

$$ERR = \frac{FP + FN}{FP + FN + TP + TN} = 1 - ACC$$

$$ACC = \frac{TP + TN}{FP + FN + TP + TN} = 1 - ERR$$

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN}$$

$$TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$

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

$$REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$

$$F_1 = 2 \cdot \frac{PRE \cdot REC}{PRE + REC}$$

Evaluating Classifiers

SKLEARN TOOLS

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

sklearn.metrics.confusion_matrix

`sklearn.metrics.confusion_matrix`(*y_true, y_pred, *, labels=None, sample_weight=None, normalize=None*) [source]

Compute confusion matrix to evaluate the accuracy of a classification.

By definition a confusion matrix C is such that $C_{i,j}$ is equal to the number of observations known to be in group i and predicted to be in group j.

Thus in binary classification, the count of true negatives is $C_{0,0}$, false negatives is $C_{1,0}$, true positives is $C_{1,1}$ and false positives is $C_{0,1}$.

Read more in the User Guide.

Parameters:

y_true : *array-like of shape (n_samples,)*
Ground truth (correct) target values.

y_pred : *array-like of shape (n_samples,)*
Estimated targets as returned by a classifier.

labels : *array-like of shape (n_classes), default=None*
List of labels to index the matrix. This may be used to reorder or select a subset of labels. If `None` is given, those that appear at least once in `y_true` or `y_pred` are used in sorted order.

sample_weight : *array-like of shape (n_samples,), default=None*
Sample weights.

New in version 0.18.

normalize : *{'true', 'pred', 'all'}, default=None*
Normalizes confusion matrix over the true (rows), predicted (columns) conditions or all the population. If None, confusion matrix will not be normalized.

```
>>> from sklearn.metrics import confusion_matrix
>>> y_true = [2, 0, 2, 2, 0, 1]
>>> y_pred = [0, 0, 2, 2, 0, 2]
>>> confusion_matrix(y_true, y_pred)
array([[2, 0, 0],
       [0, 0, 1],
       [1, 0, 2]])
```

```
>>> y_true = ["cat", "ant", "cat", "cat", "ant", "bird"]
>>> y_pred = ["ant", "ant", "cat", "cat", "ant", "cat"]
>>> confusion_matrix(y_true, y_pred, labels=["ant", "bird", "cat"])
array([[2, 0, 0],
       [0, 0, 1],
       [1, 0, 2]])
```

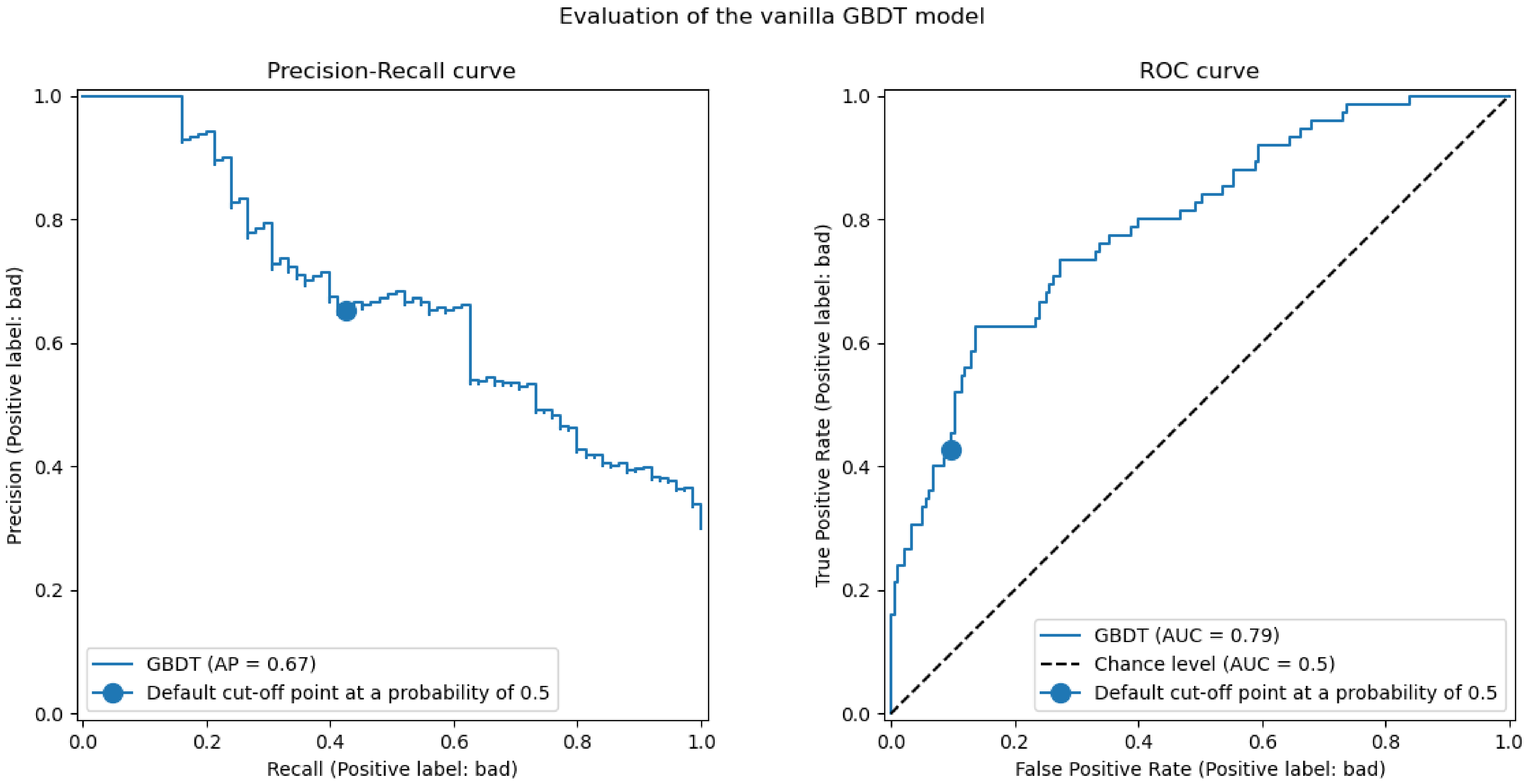
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html#sklearn.metrics.classification_report

```
>>> from sklearn.metrics import classification_report
>>> y_true = [0, 1, 2, 2, 2]
>>> y_pred = [0, 0, 2, 2, 1]
>>> target_names = ['class 0', 'class 1', 'class 2']
>>> print(classification_report(y_true, y_pred, target_names=target_names))
              precision    recall  f1-score   support

     class 0       0.50      1.00      0.67         1
     class 1       0.00      0.00      0.00         1
     class 2       1.00      0.67      0.80         3

    accuracy                           0.60         5
   macro avg       0.50      0.56      0.49         5
weighted avg       0.70      0.60      0.61         5
```

Overall accuracy = 0.60

Note: in the binary case, the precision and recall for the problem are the values for the positive class

- Inherent trade-off precision vs recall (or TPR vs FPR)
- We can adjust a classifier to optimize for one or the other, but never both simultaneously.
- See User Guide section 3.3 for techniques to adjust the threshold



References:

- https://scikit-learn.org/stable/auto_examples/miscellaneous/plot_display_object_visualization.html#sphx-glr-auto-examples-miscellaneous-plot-display-object-visualization-py
- https://scikit-learn.org/stable/auto_examples/model_selection/plot_cost_sensitive_learning.html#sphx-glr-auto-examples-model-selection-plot-cost-sensitive-learning-py
- https://scikit-learn.org/stable/modules/classification_threshold.html

Summary

Know the following:

- Accuracy / error rate
- TP, FP, TN, FN in confusion matrix
- Precision
- Recall
- F1 Score
- REMEMBER THE TEST/TRAIN SPLIT

Don’t need to memorize, just be familiar with the concepts and know how to look up definition as needed…

[1]

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import ConfusionMatrixDisplay
#from sklearn import metrics
```

	Predicted ⊕	*Predicted* ⊖	
Actual ⊕	**30**	**20**	50
Actual ⊖	**10**	**40**	50
	40	60	100

[2]

```
y_pred = np.array([1,1,1,1,0,0,0,0,0,0])
y_true = np.array([1,1,1,0,1,1,0,0,0,0])
```

[3]

```
print(confusion_matrix(y_true,y_pred)) # the default sorts entries, so it shows 0 then 1
```

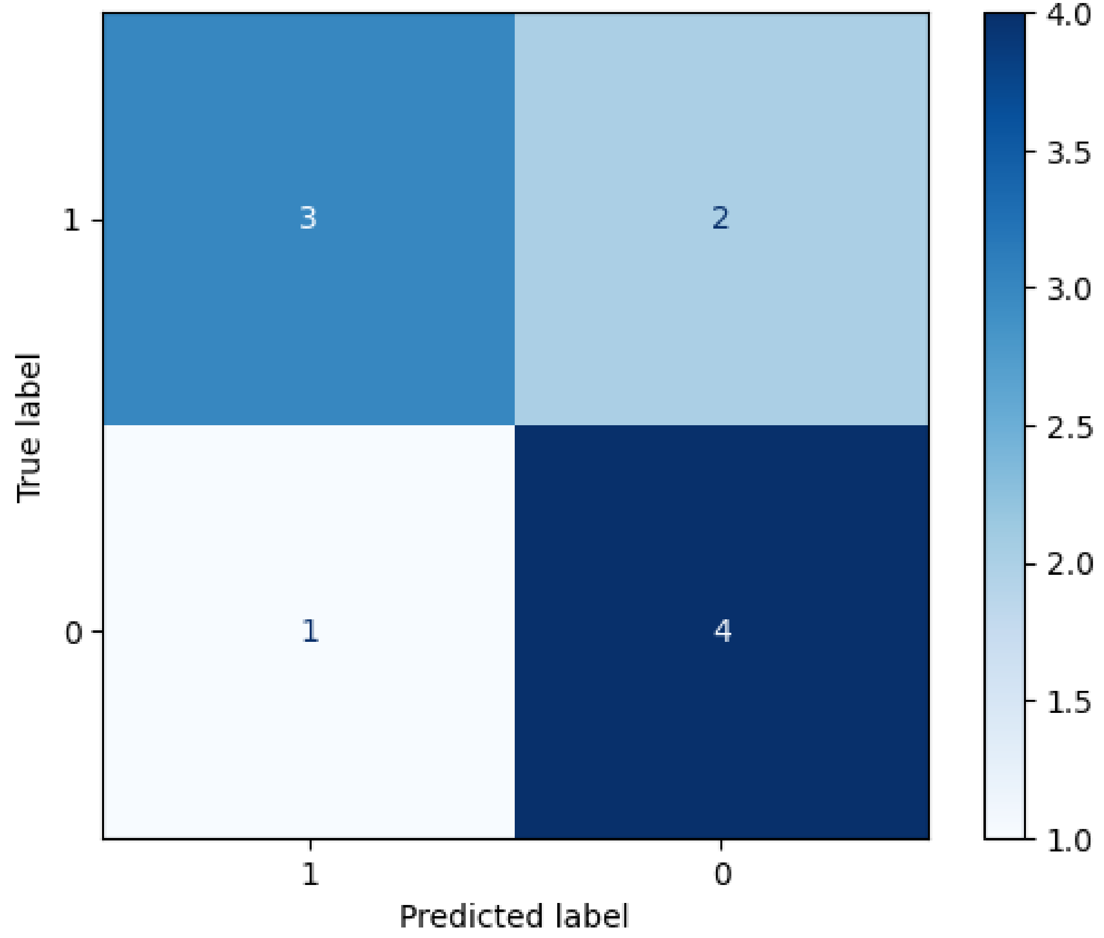
```
[[4 1]
 [2 3]]
```

[4]

```
print(confusion_matrix(y_true,y_pred,labels=[1,0]))  # to reverse this like ppt slide add labels argument
```

```
[[3 2]
 [1 4]]
```

```
1 ConfusionMatrixDisplay.from_predictions(y_true, y_pred, labels=[1,0],cmap='Blues')
2 plt.show()
```



	Predicted ⊕	Predicted ⊖	
Actual ⊕	**30**	**20**	50
Actual ⊖	**10**	**40**	50
	40	60	100

```
[7]  print(classification_report(y_true, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.67      0.80      0.73         5
           1       0.75      0.60      0.67         5

    accuracy                           0.70        10
   macro avg       0.71      0.70      0.70        10
weighted avg       0.71      0.70      0.70        10
```

```
from sklearn.metrics import precision_score
precision_score(y_true,y_pred)
```

```
0.75
```

	Predicted ⊕	*Predicted* ⊖	
Actual ⊕	**30**	**20**	50
Actual ⊖	**10**	**40**	50
	40	60	100

- In binary case, metrics like precision, recall, and f1-score are defined for positive case only
- Averages can be used for multiclass cases

Evaluating Classifiers

MORE PROBLEMS

Chapter 2.1

Difficulty: 2

Given that:

Total = 100
False Negatives = 10
Precision = 4/5
Recall = 6/7
Can you complete the contingency matrix.

	Predicted +	Predicted -	
Actual +			
Actual -			

Chapter 2.1

Difficulty: 4

Two binary classifiers are used to predicted whether a patent has a life threatening diseases or not. Decide whether Classifier A or B would be better at reducing casualties.

A

10	10	20
40	9940	9980
50	9950	10000

B

13	7	20
87	9813	9980
100	9900	10000

Two contingence matrices.