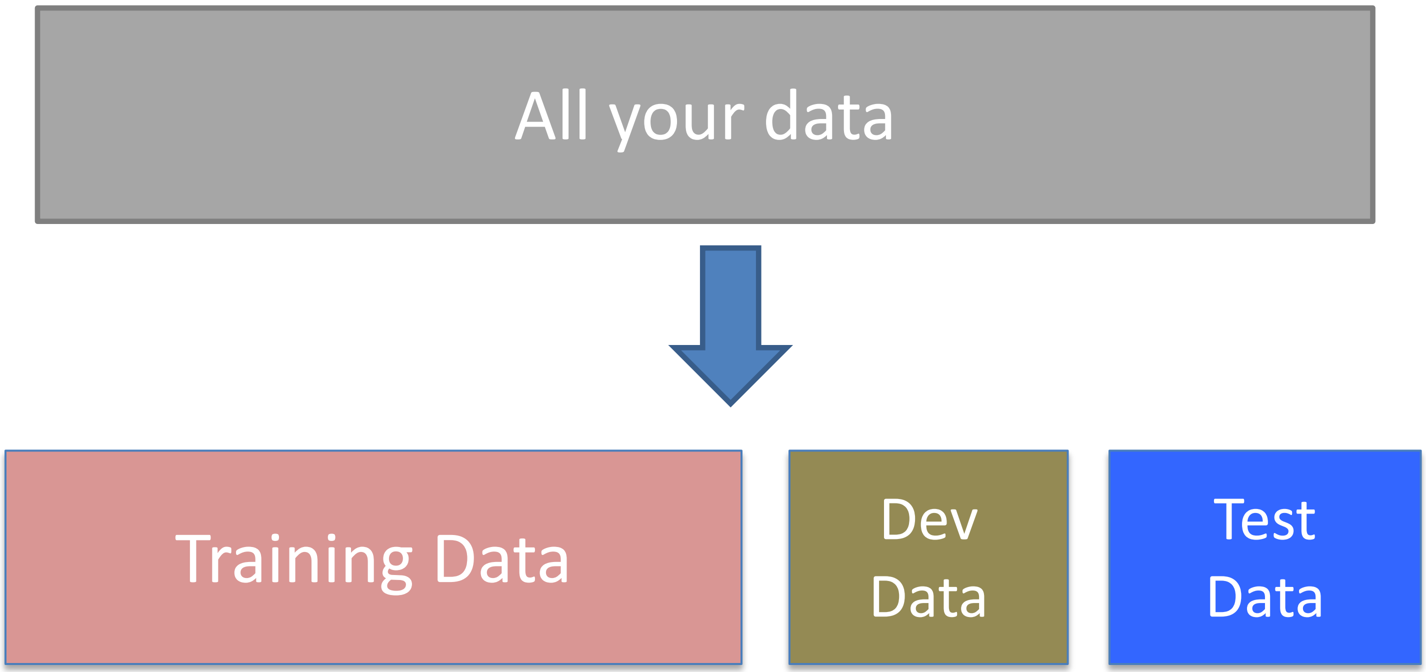
CMSC 471: Machine Learning

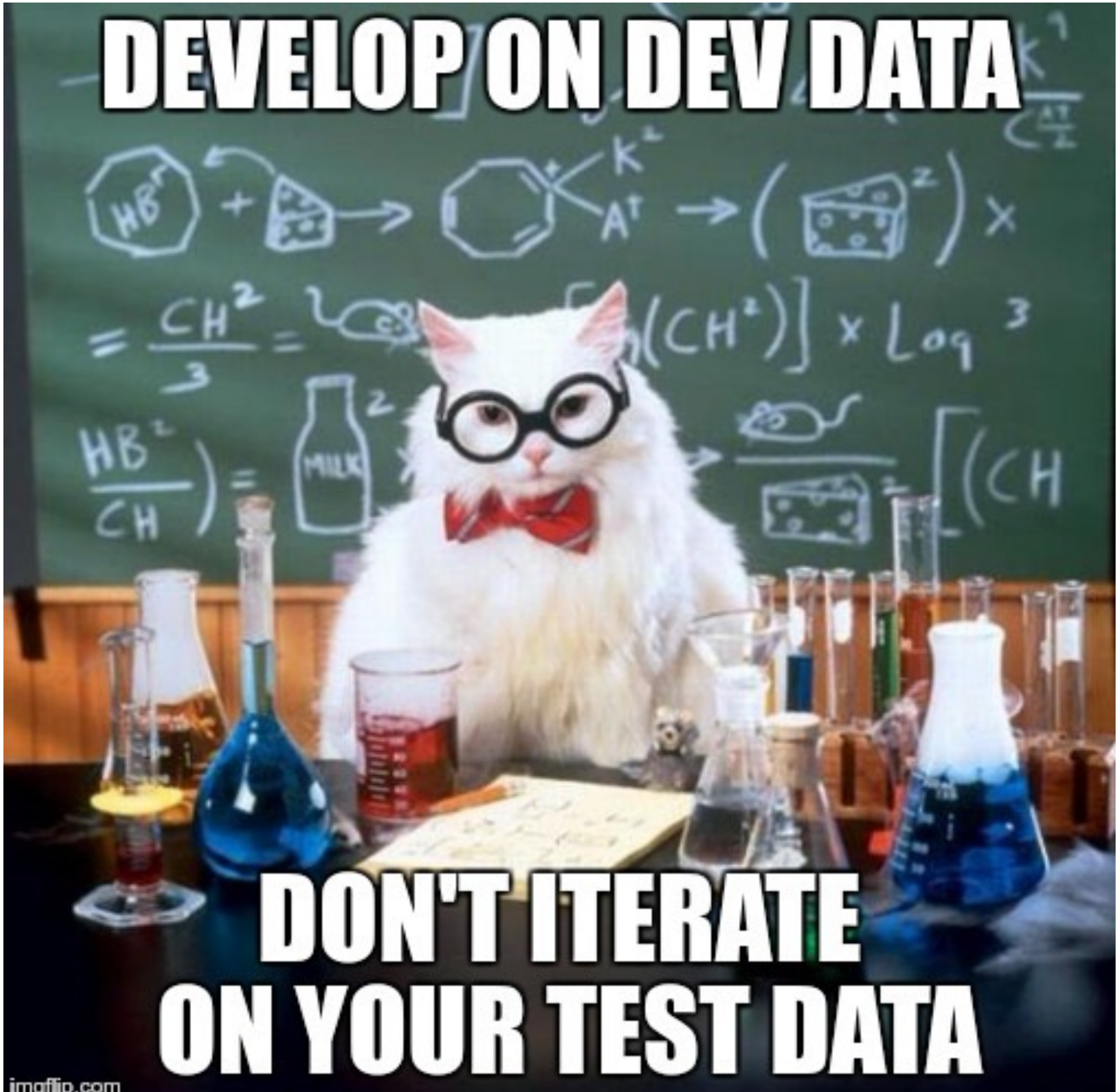
KMA Solaiman – ksolaima@umbc.edu

Some slides courtesy Tim Finin and Frank Ferraro

Experimenting with Machine Learning Models



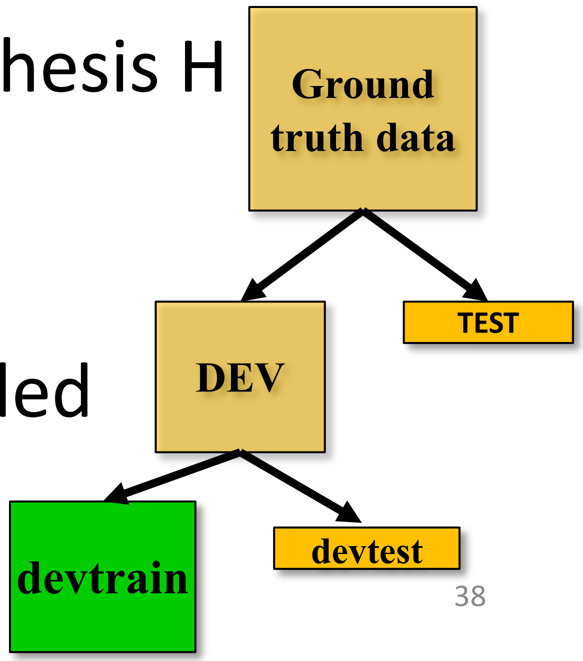
Rule #1


DEVELOP ON DEV DATA
DON'T ITERATE
ON YOUR TEST DATA
imgflip.com

Evaluation methodology (3)

Common variation on methodology:

1. Collect set of examples with correct classifications
2. Randomly divide it into two disjoint sets: ***development*** & ***test****;* further divide development into ***devtrain*** & ***devtest***
3. Apply ML to *devtrain,* creating hypothesis H
4. Measure performance of H w.r.t. *devtest* data
5. Modify approach, repeat 3-4 as needed
6. Final test on *test* data

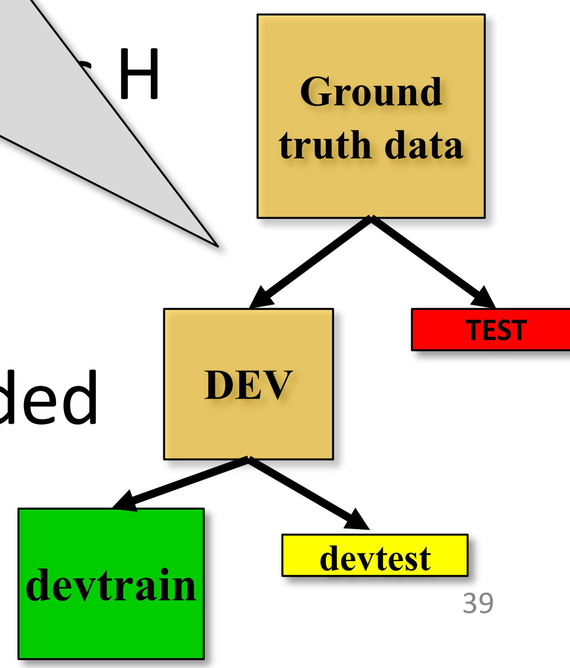


Evaluation methodology (4)

Co

- Only **devtest** data used for evaluation during system **development**
- When all development has ended, **test** data used for **final evaluation**
- Ensures final system not influenced by test data
- If more development needed, get new dataset!

1. … classifications
2. … sets: … development
3. … H
4. … *devtest* data
5. Modify approach, repeat 3-4 as needed
6. Final test on *test* data



Zoo evaluation

train_and_test(learner, data, start, end) uses data[start:end] for test and rest for train

```
>>> dtl = DecisionTreeLearner
>>> train_and_test(dtl(), zoo, 0, 10)
1.0
>>> train_and_test(dtl(), zoo, 90, 100)
0.80000000000000004
>>> train_and_test(dtl(), zoo, 90, 101)
0.81818181818181823
>>> train_and_test(dtl(), zoo, 80, 90)
0.90000000000000002
```

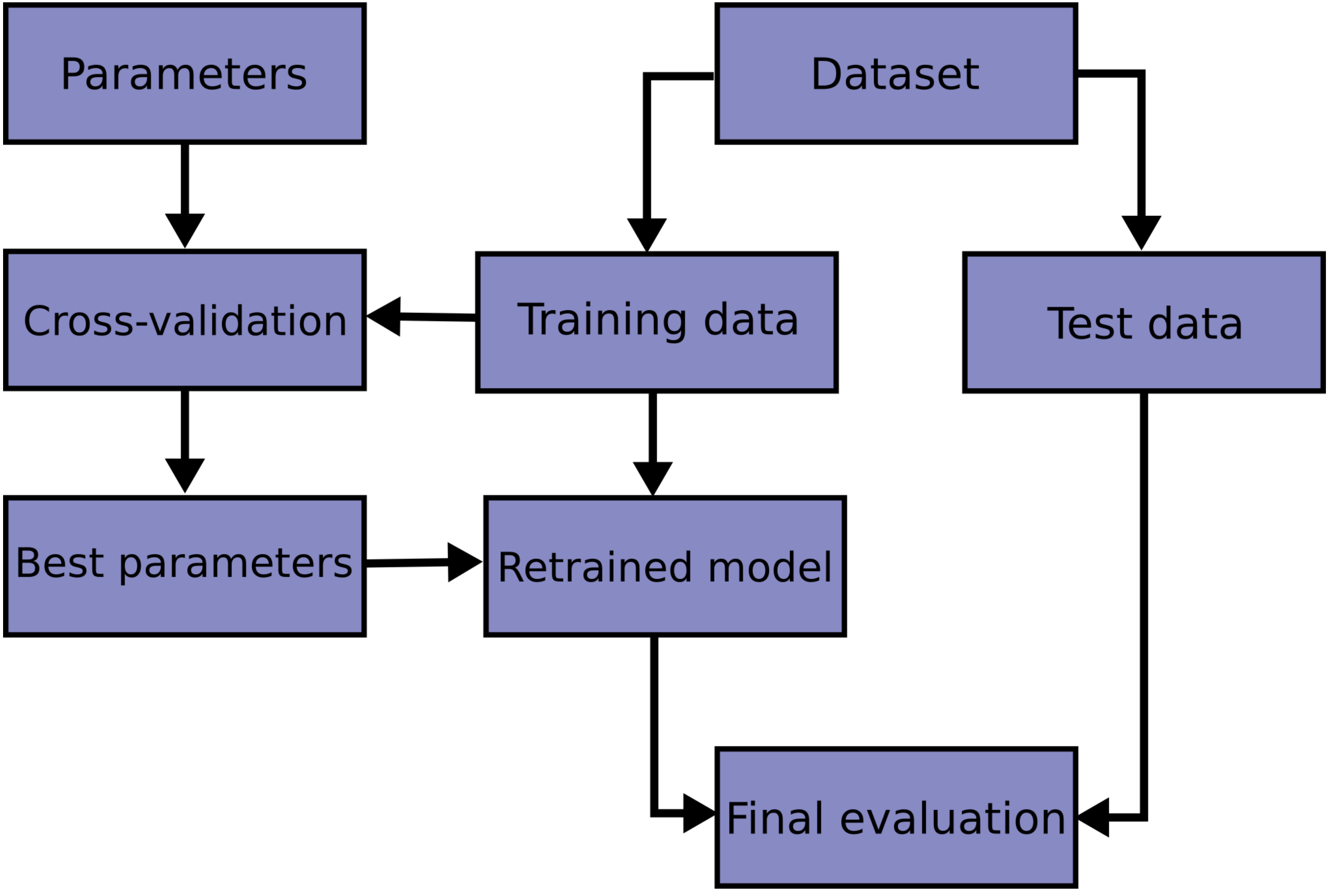
Zoo evaluation

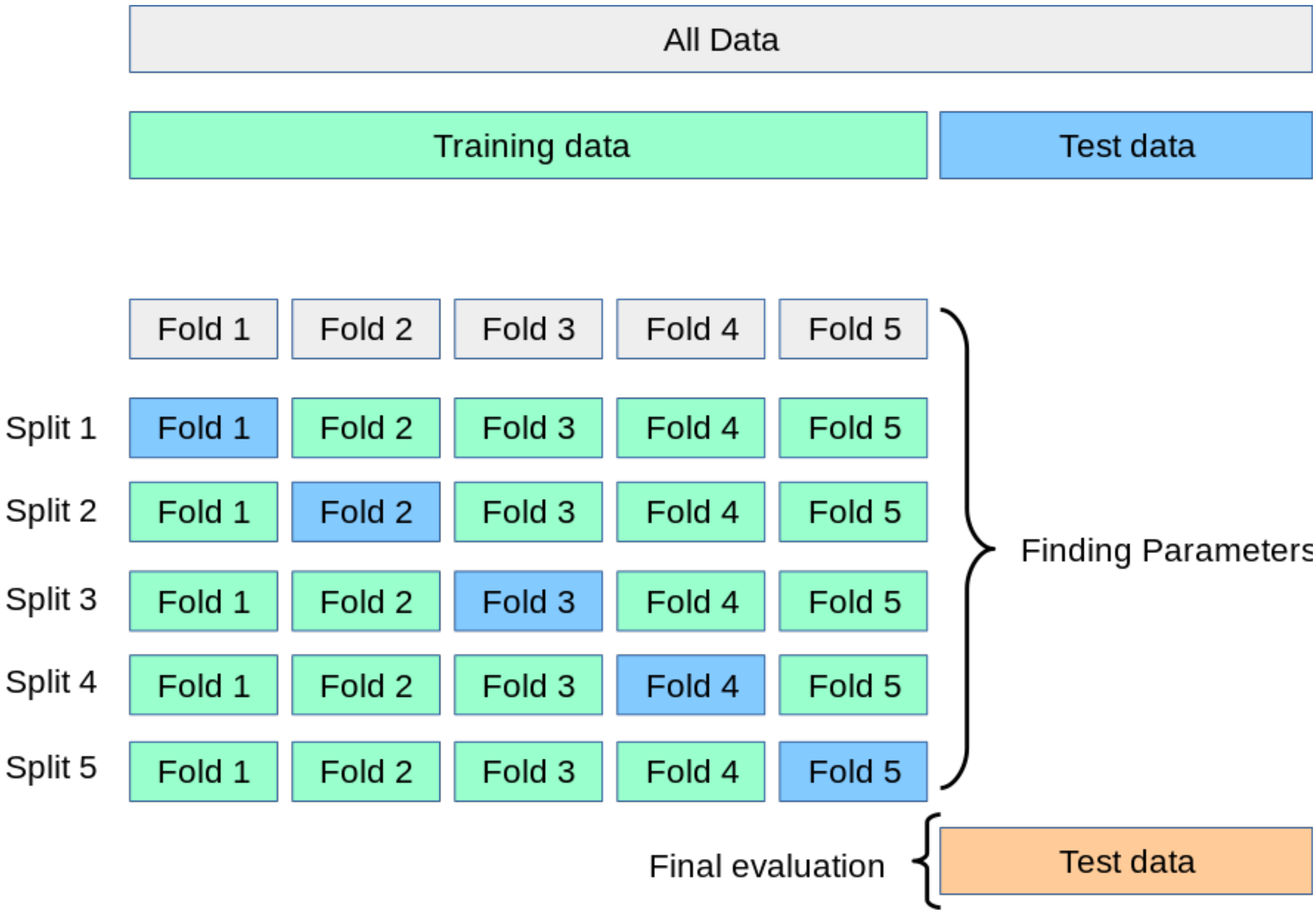
train_and_test(learner, data, start, end) uses data[start:end] for test and rest for train

- We hold out 10 data items for test; train on the other 91; show the accuracy on the test data
- Doing this **four times for different test subsets** shows accuracy from 80% to 100%
- **What's the true accuracy of our approach?**

K-fold Cross Validation

- **Problems:**
  - getting *ground truth* data expensive
  - need different test data for each test
  - experiments needed to find right *feature space* & parameters for ML algorithms
- **Goal:** minimize training+test data needed
- **Idea:** split training data into K subsets; use K-1 for *training* and one for *development testing*
- Repeat K times and average performance
- Common K values are 5 and 10


Parameters
Dataset
Cross-validation
Training data
Test data
Best parameters
Retrained model
Final evaluation


All Data
Training data
Test data
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 1
Split 2
Split 3
Split 4
Split 5
Finding Parameters
Final evaluation
Test data

Zoo evaluation

- AIMA code has a cross_validation function that runs K-fold cross validation
- cross_validation(learner, data, K, N) does N iterations, each time randomly selecting 1/K data points for test, leaving rest for train

```
>>> cross_validation(dtl(), zoo, 10, 20)
0.9550000000000007
```

- This is a very common approach to evaluating the accuracy of a model during development
- Best practice is still to hold out a final test data set

Leave one out Cross Validation

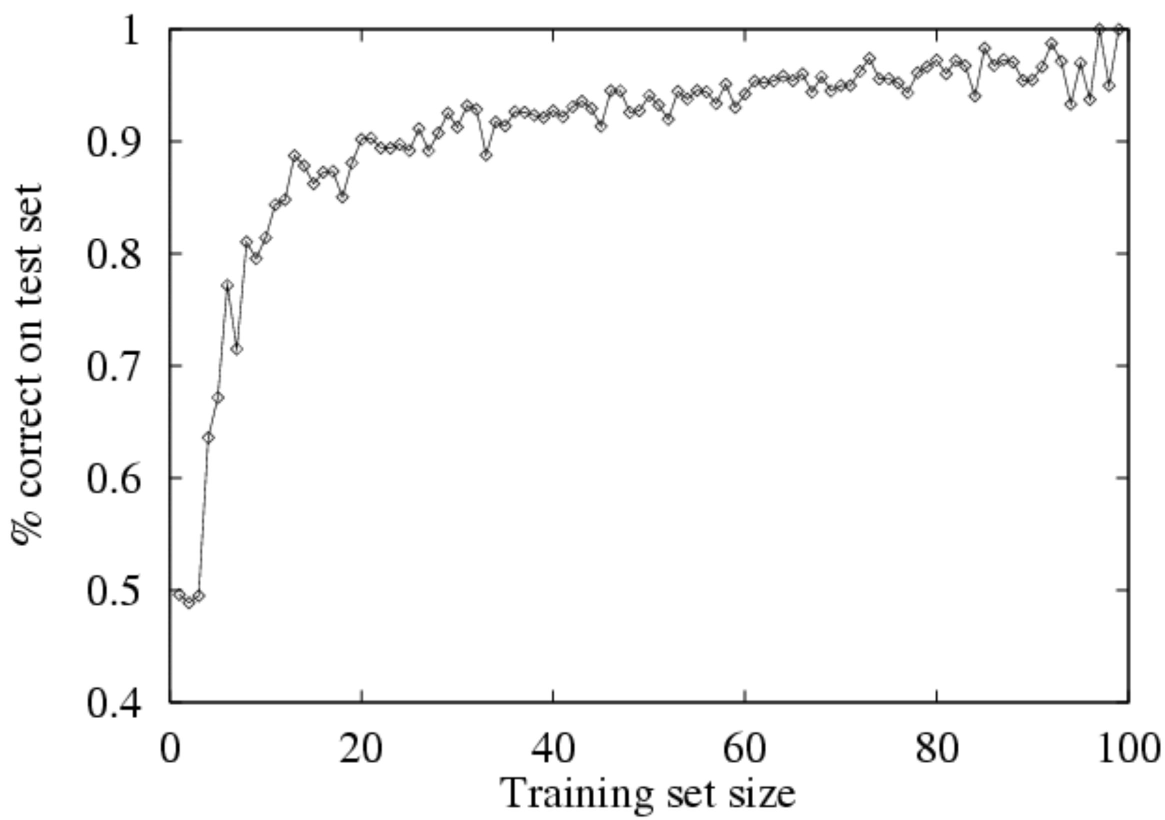
- AIMA code also has a *leave1out* function that runs a different set of experiments to estimate accuracy of the model
- *leave1out(learner, data)* does len(data) trials, each using one element for test, rest for train

```
>>> leave1out(dtl(), zoo)
0.97029702970297027
```

- K-fold cross validation can be too pessimistic, since it only trains with 80% or 90% of the data
- The leave one out evaluation is an alternative

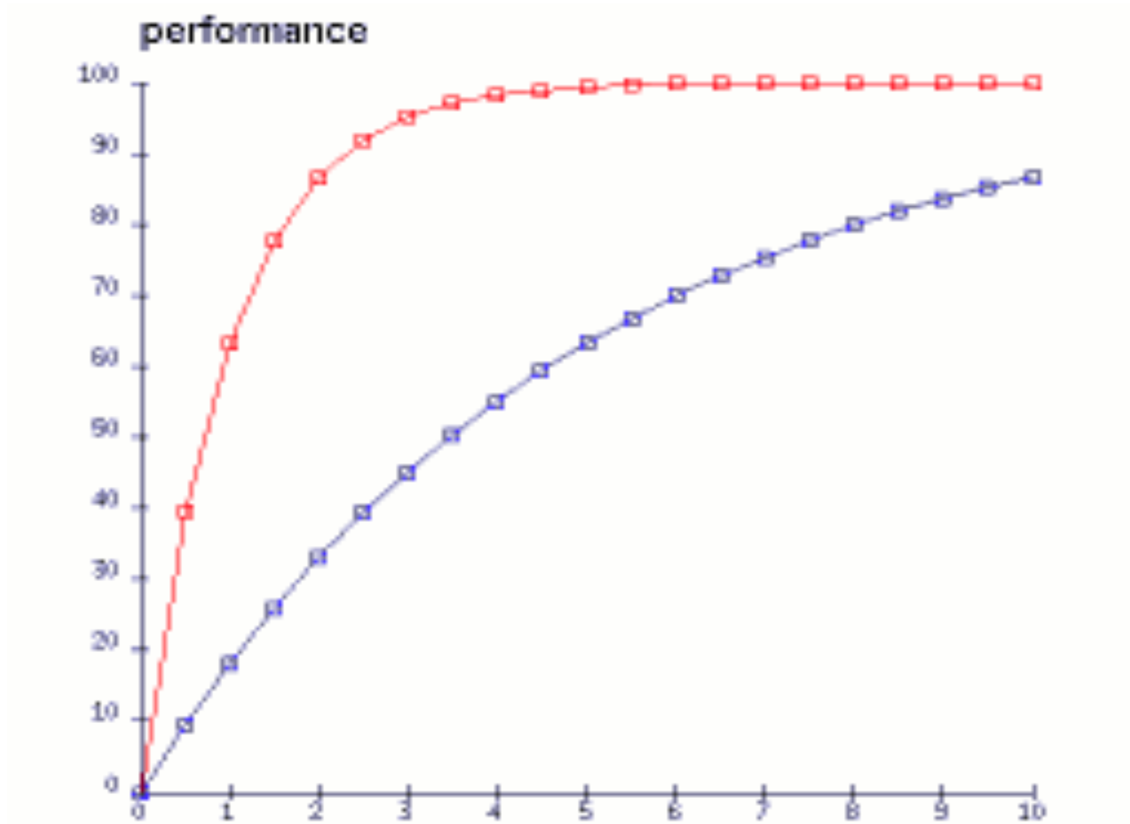
Learning curve (1)

A learning curve shows accuracy on test set as a function of training set size or (for neural networks) running time



Learning curve

- When evaluating ML algorithms, steeper learning curves are better
- They represents faster learning with less data



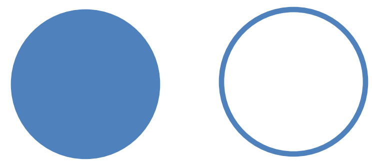
Training set size

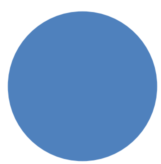
Here the system with the red curve is better since it requires less data to achieve desired accuracy

EVALUATION METRICS

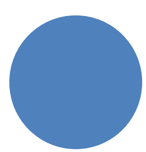
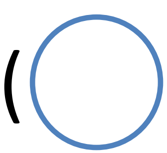
Classification Evaluation: the 2-by-2 contingency table

Let's assume there are two classes/labels



Assume  is the "positive" label

Given X, our classifier predicts either label

p(|X) vs. p(|X)

Classification Evaluation: the 2-by-2 contingency table

	What is the actual label?	
What label does our system predict? (↓)	**Actually Correct**	**Actually Incorrect**
Selected/ Guessed		
Not selected/ not guessed		

Classes/Choices

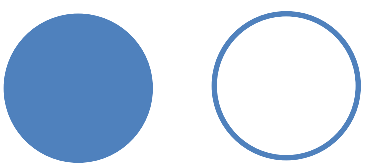
Classification Evaluation: the 2-by-2 contingency table

	What is the actual label?	
What label does our system predict? (↓)	**Actually Correct**	**Actually Incorrect**
Selected/ Guessed	True Positive (TP) *Actual* *Guessed*	
Not selected/ not guessed		

Classes/Choices

Classification Evaluation: the 2-by-2 contingency table

	What is the actual label?	
What label does our system predict? (↓)	**Actually Correct**	**Actually Incorrect**
Selected/ Guessed	True Positive (TP) *Actual* *Guessed*	False Positive (FP) *Actual* *Guessed*
Not selected/ not guessed		



Classes/Choices

Classification Evaluation: the 2-by-2 contingency table

	What is the actual label?	
What label does our system predict? (↓)	**Actually Correct**	**Actually Incorrect**
Selected/ Guessed	True Positive (TP) *Actual* *Guessed*	False Positive (FP) *Actual* *Guessed*
Not selected/ not guessed	False Negative (FN) *Actual* *Guessed*	

Classes/Choices

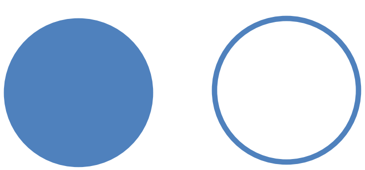
Classification Evaluation: the 2-by-2 contingency table

	What is the actual label?	
What label does our system predict? (↓)	**Actually Correct**	**Actually Incorrect**
Selected/ Guessed	True Positive (TP) *Actual* *Guessed*	False Positive (FP) *Actual* *Guessed*
Not selected/ not guessed	False Negative (FN) *Actual* *Guessed*	True Negative (TN) *Actual* *Guessed*

Classes/Choices

Classification Evaluation: the 2-by-2 contingency table

	What is the actual label?	
What label does our system predict? (↓)	**Actually Correct**	**Actually Incorrect**
Selected/ Guessed	True Positive *Actual* (TP) *Guessed*	False Positive *Actual* (FP) *Guessed*
Not selected/ not guessed	False Negative *Actual* (FN) *Guessed*	True Negative *Actual* (TN) *Guessed*

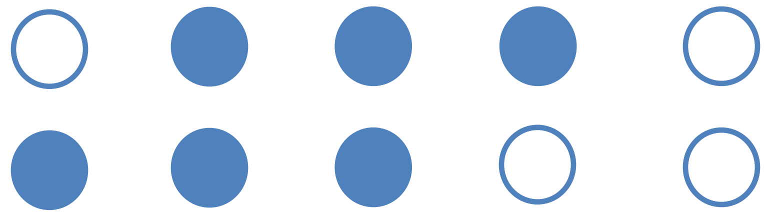

Classes/Choices

Construct this table by *counting* the number of TPs, FPs, FNs, TNs

Contingency Table Example

Predicted:

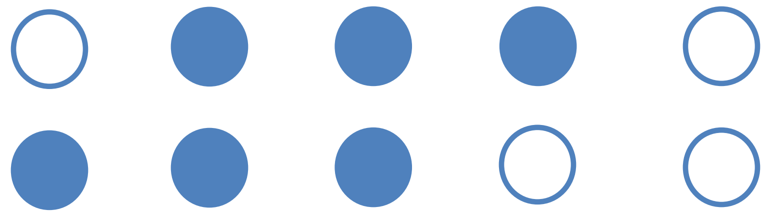
Actual:



Contingency Table Example

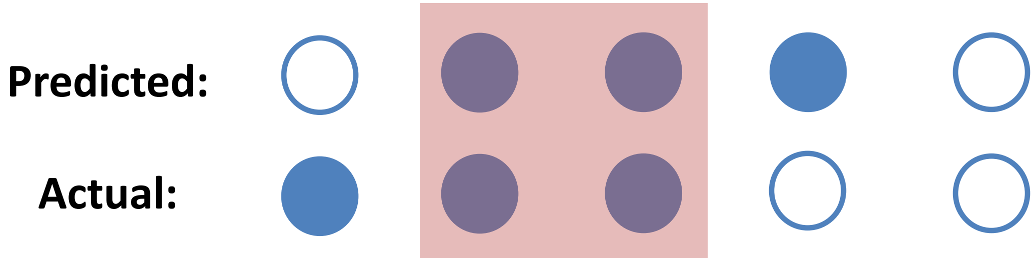
Predicted:

Actual:



	What is the actual label?	
What label does our system predict? (↓)	**Actually Correct**	**Actually Incorrect**
Selected/ Guessed	True Positive (TP)	False Positive (FP)
Not selected/ not guessed	False Negative (FN)	True Negative (TN)

Contingency Table Example

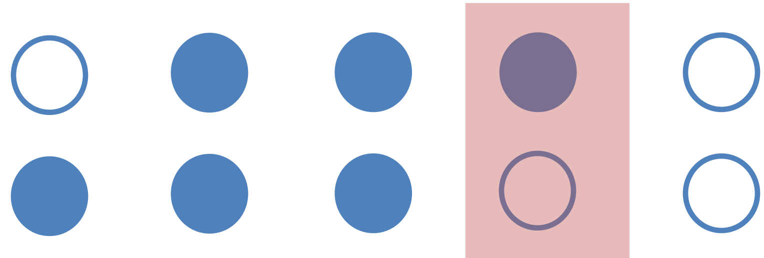


	What is the actual label?	
What label does our system predict? (↓)	**Actually Correct**	**Actually Incorrect**
Selected/ Guessed	True Positive (TP) = 2	False Positive (FP)
Not selected/ not guessed	False Negative (FN)	True Negative (TN)

Contingency Table Example

Predicted:

Actual:

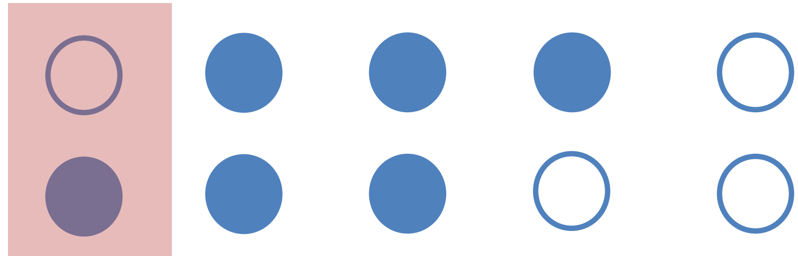


	What is the actual label?	
What label does our system predict? (↓)	**Actually Correct**	**Actually Incorrect**
Selected/ Guessed	True Positive (TP) = 2	False Positive (FP) = 1
Not selected/ not guessed	False Negative (FN)	True Negative (TN)

Contingency Table Example

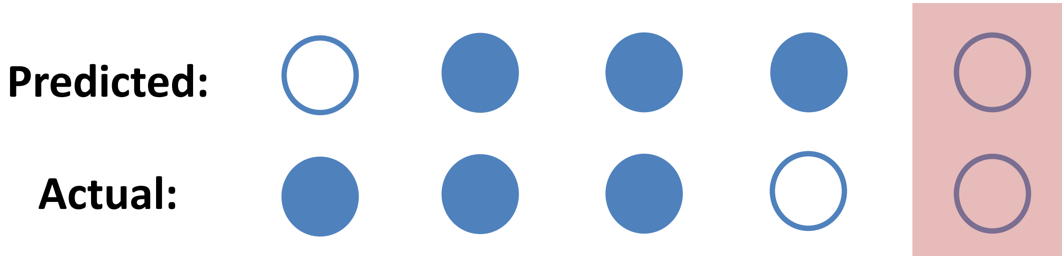
Predicted:

Actual:



	What is the actual label?	
What label does our system predict? (↓)	**Actually Correct**	**Actually Incorrect**
Selected/ Guessed	True Positive (TP) = 2	False Positive (FP) = 1
Not selected/ not guessed	False Negative (FN) = 1	True Negative (TN)

Contingency Table Example

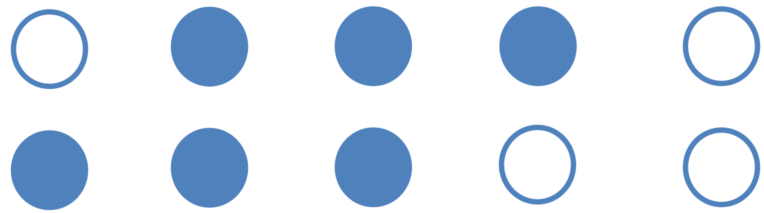


	What is the actual label?	
What label does our system predict? (↓)	**Actually Correct**	**Actually Incorrect**
Selected/ Guessed	True Positive (TP) = 2	False Positive (FP) = 1
Not selected/ not guessed	False Negative (FN) = 1	True Negative (TN) = 1

Contingency Table Example

Predicted:

Actual:



	What is the actual label?	
What label does our system predict? (↓)	**Actually Correct**	**Actually Incorrect**
Selected/ Guessed	True Positive (TP) = 2	False Positive (FP) = 1
Not selected/ not guessed	False Negative (FN) = 1	True Negative (TN) = 1

Classification Evaluation: Accuracy, Precision, and Recall

Accuracy: % of items correct

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

	Actually Correct	**Actually Incorrect**
Selected/Guessed	True Positive (TP)	False Positive (FP)
Not select/not guessed	False Negative (FN)	True Negative (TN)

Classification Evaluation: Accuracy, Precision, and Recall

Accuracy: % of items correct

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

Precision: % of selected items that are correct

$$\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

	Actually Correct	**Actually Incorrect**
Selected/Guessed	True Positive (TP)	False Positive (FP)
Not select/not guessed	False Negative (FN)	True Negative (TN)

Classification Evaluation: Accuracy, Precision, and Recall

Accuracy: % of items correct

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

Precision: % of selected items that are correct

$$\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

Recall: % of correct items that are selected

$$\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

	Actually Correct	Actually Incorrect
Selected/Guessed	True Positive (TP)	False Positive (FP)
Not select/not guessed	False Negative (FN)	True Negative (TN)

Classification Evaluation: Accuracy, Precision, and Recall

Accuracy: % of items correct

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

Precision: % of selected items that are correct

$$\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

Min: 0 ☹

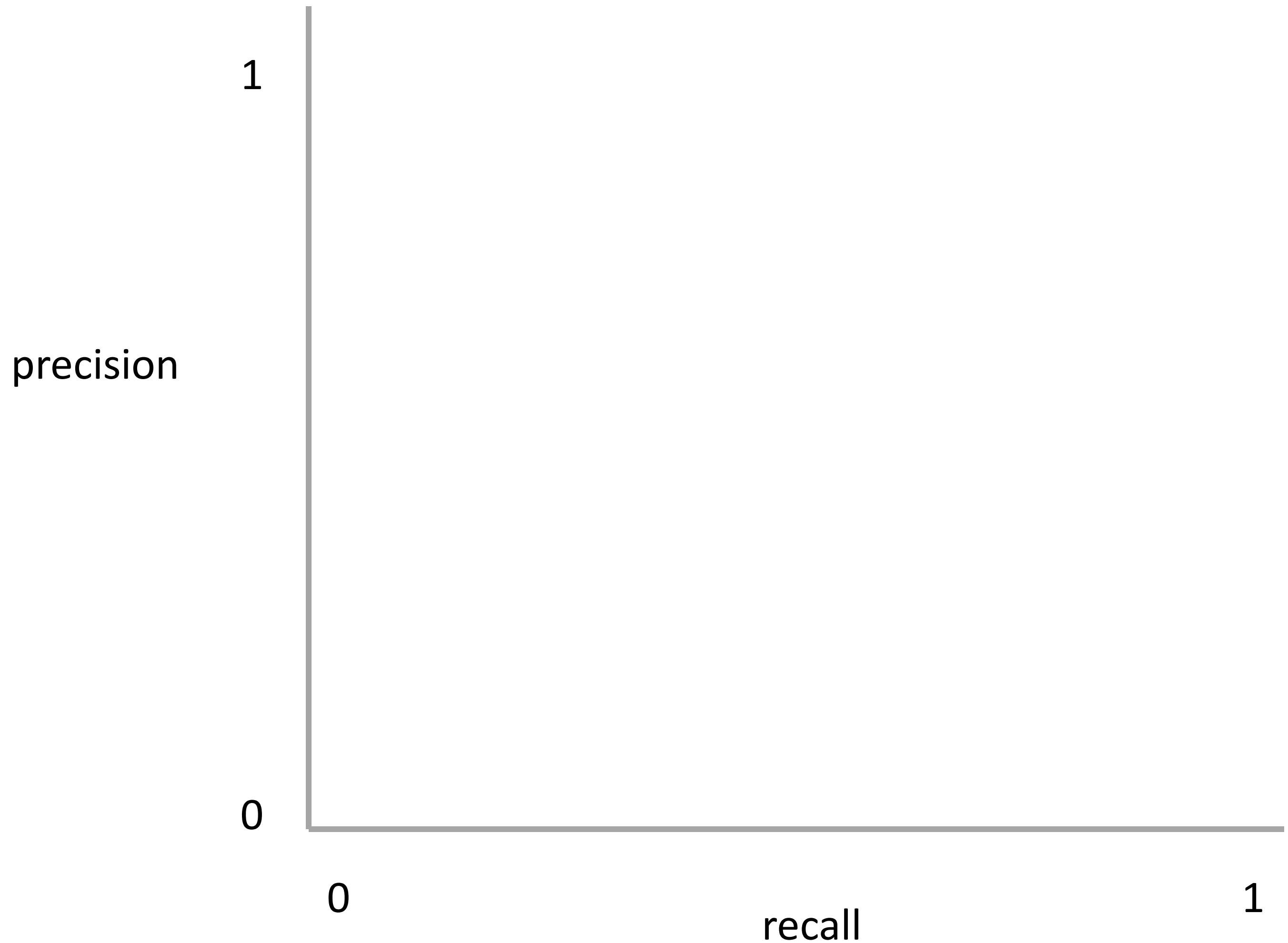
Max: 1 😀

Recall: % of correct items that are selected

$$\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

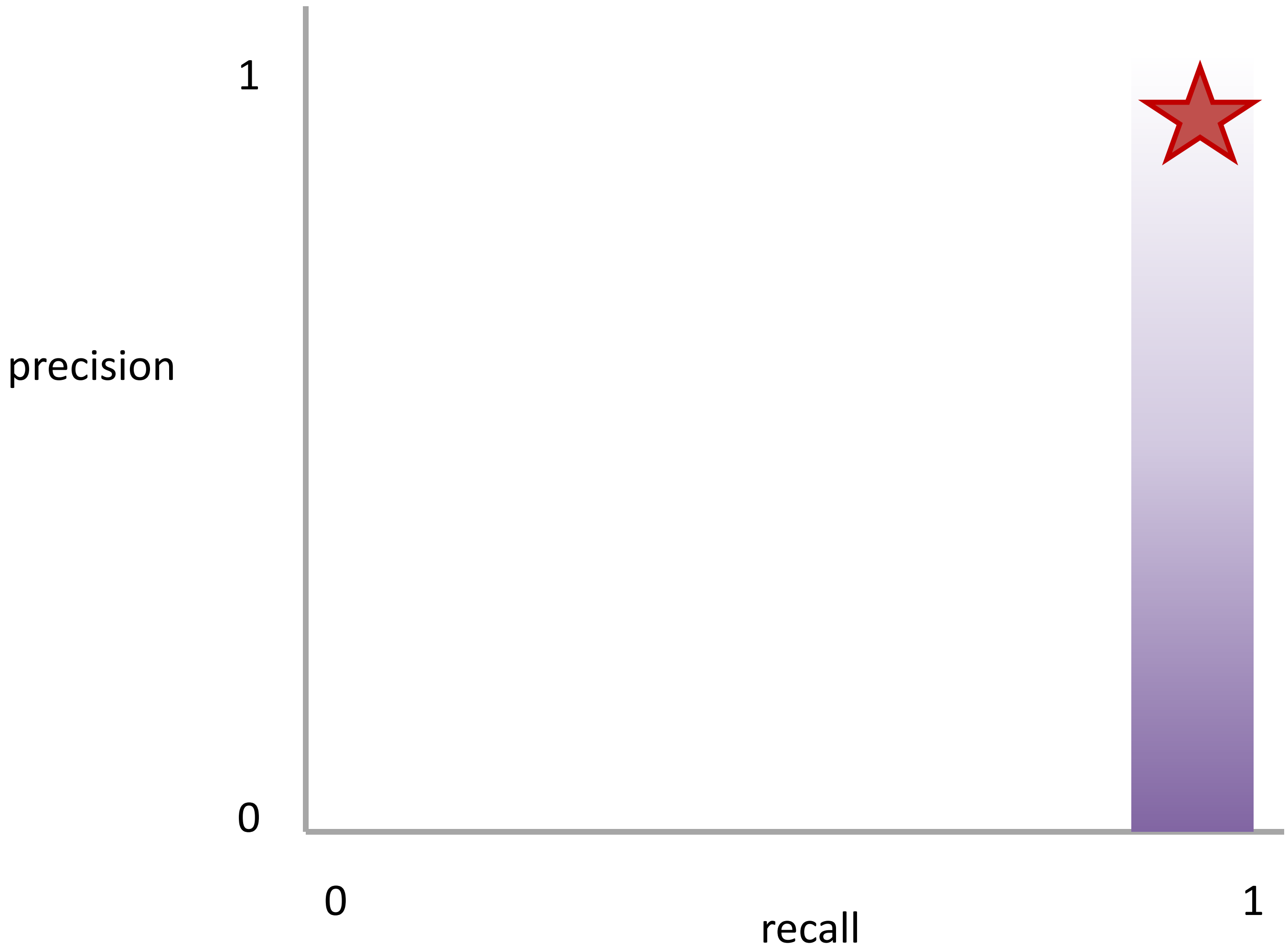
	Actually Correct	Actually Incorrect
Selected/Guessed	True Positive (TP)	False Positive (FP)
Not select/not guessed	False Negative (FN)	True Negative (TN)

Precision and Recall Present a Tradeoff



Q: Where do you want your ideal model?

Precision and Recall Present a Tradeoff

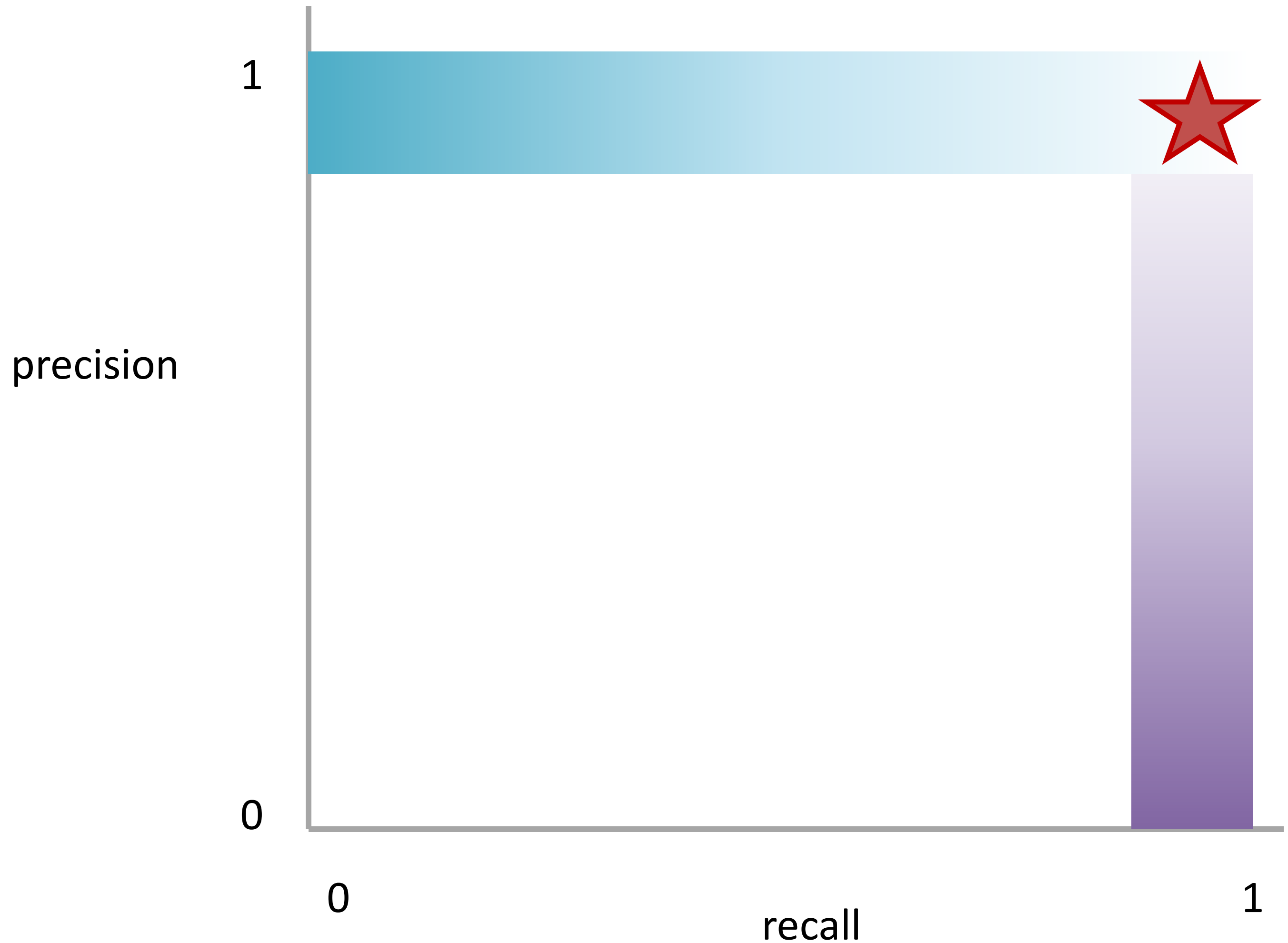


Q: Where do you want your ideal model?

Q: You have a model that always identifies correct instances. Where on this graph is it?

Q: You have a model that only make correct predictions. Where on this graph is it?

Precision and Recall Present a Tradeoff

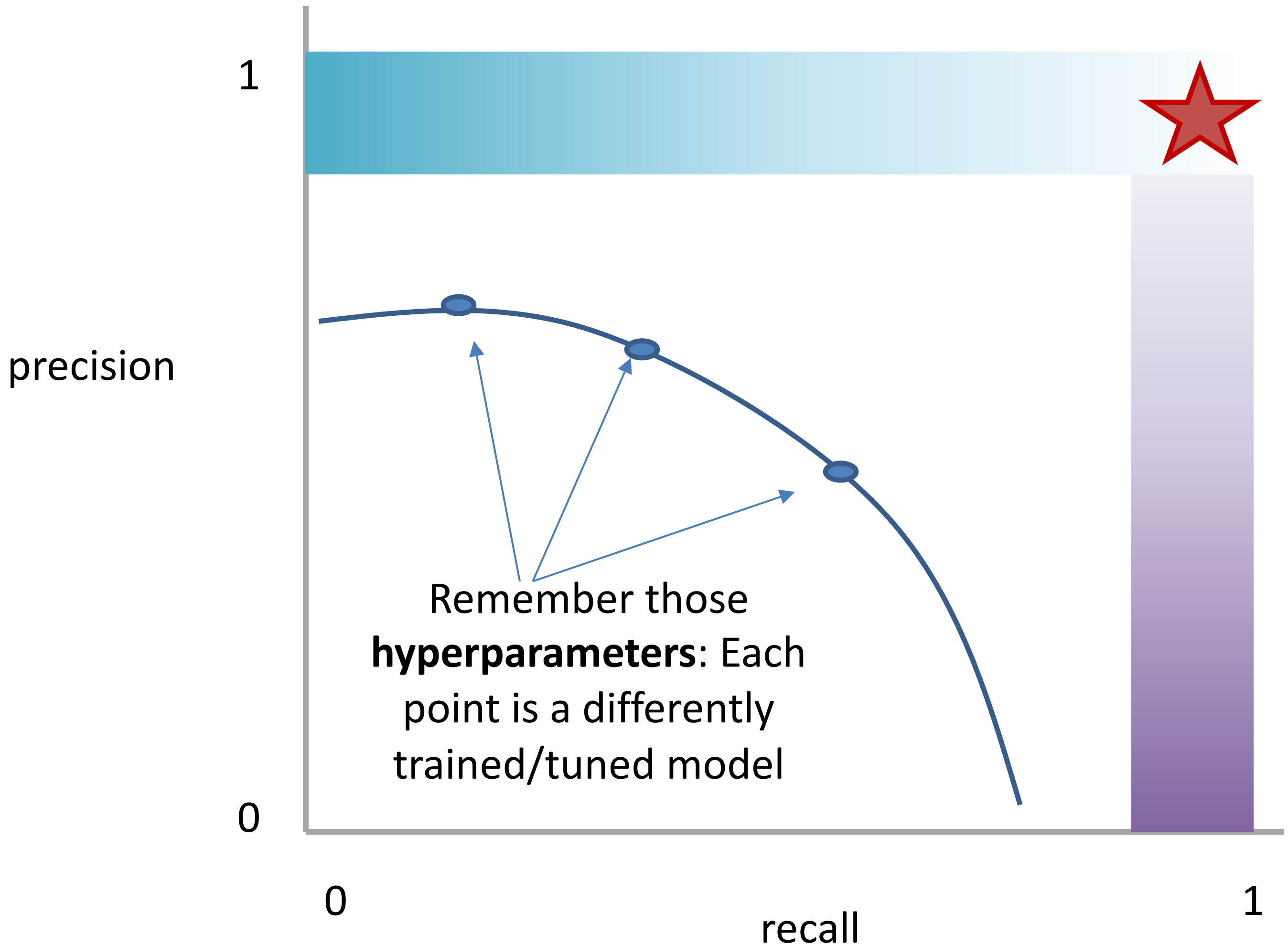


Q: Where do you want your ideal model ?

Q: You have a model that always identifies correct instances. Where on this graph is it?

Q: You have a model that only make correct predictions. Where on this graph is it?

Precision and Recall Present a Tradeoff



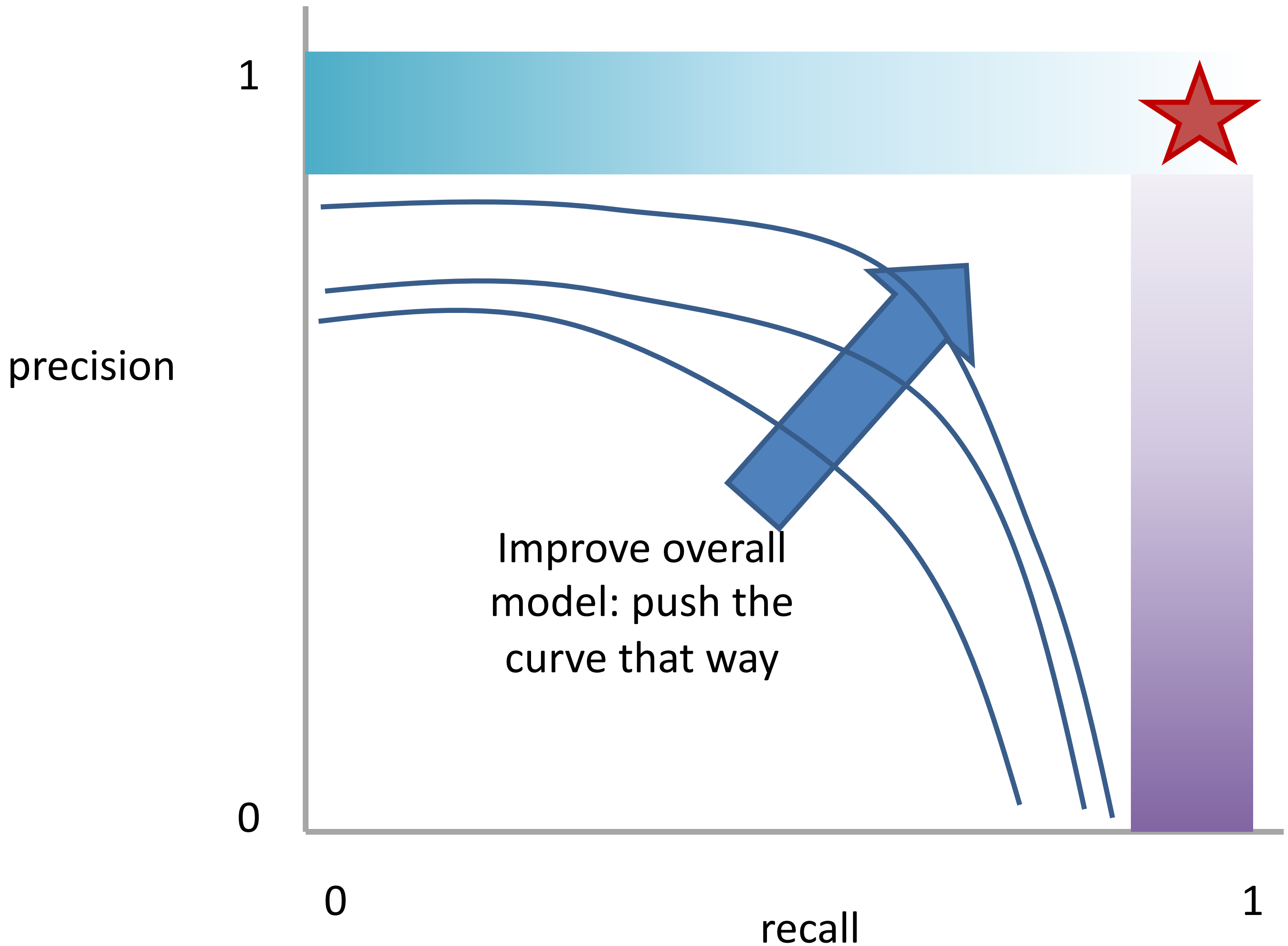
Q: Where do you want your ideal model ?

Q: You have a model that always identifies correct instances. Where on this graph is it?

Q: You have a model that only make correct predictions. Where on this graph is it?

Idea: measure the tradeoff between precision and recall

Precision and Recall Present a Tradeoff



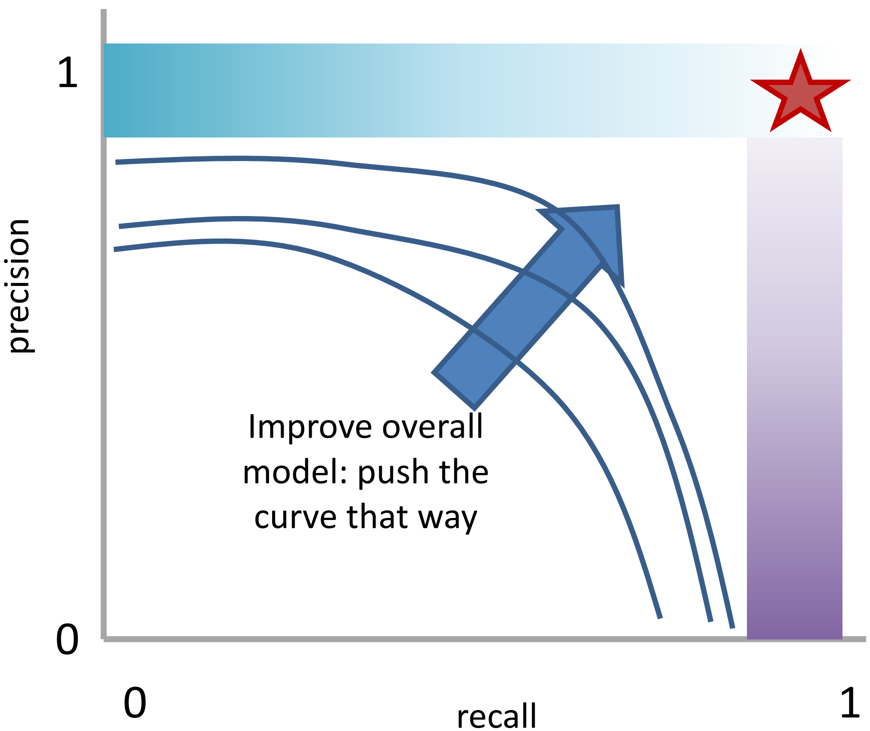
Q: Where do you want your ideal model?

Q: You have a model that always identifies correct instances. Where on this graph is it?

Q: You have a model that only make correct predictions. Where on this graph is it?

Idea: measure the tradeoff between precision and recall

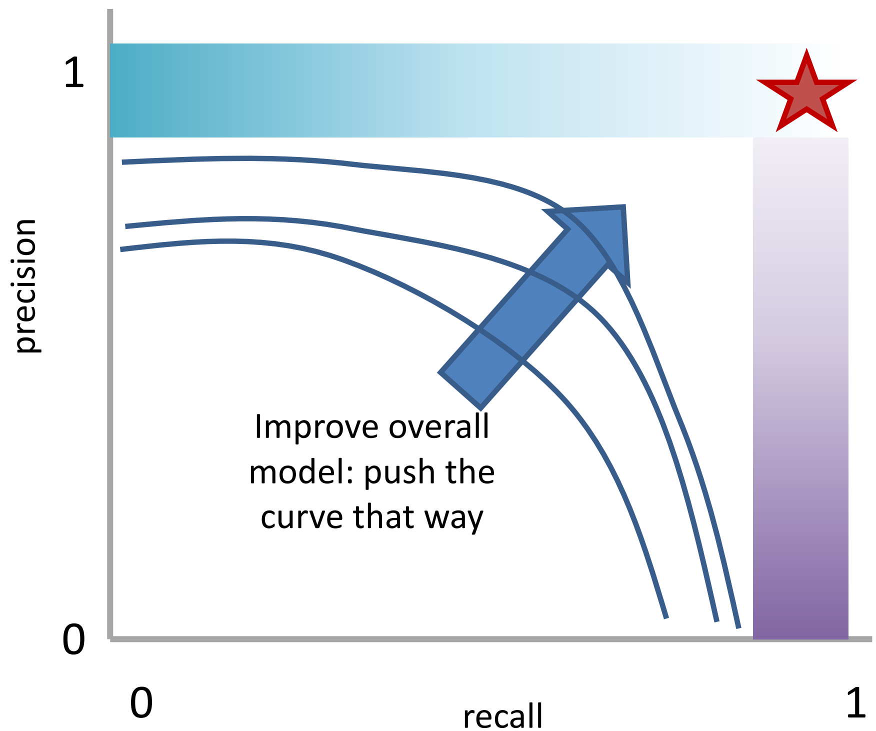
Measure this Tradeoff: Area Under the Curve (AUC)



AUC measures the area under this tradeoff curve

Min AUC: 0 ☹️
Max AUC: 1 😀

Measure this Tradeoff: Area Under the Curve (AUC)



Min AUC: 0 ☹
Max AUC: 1 😀

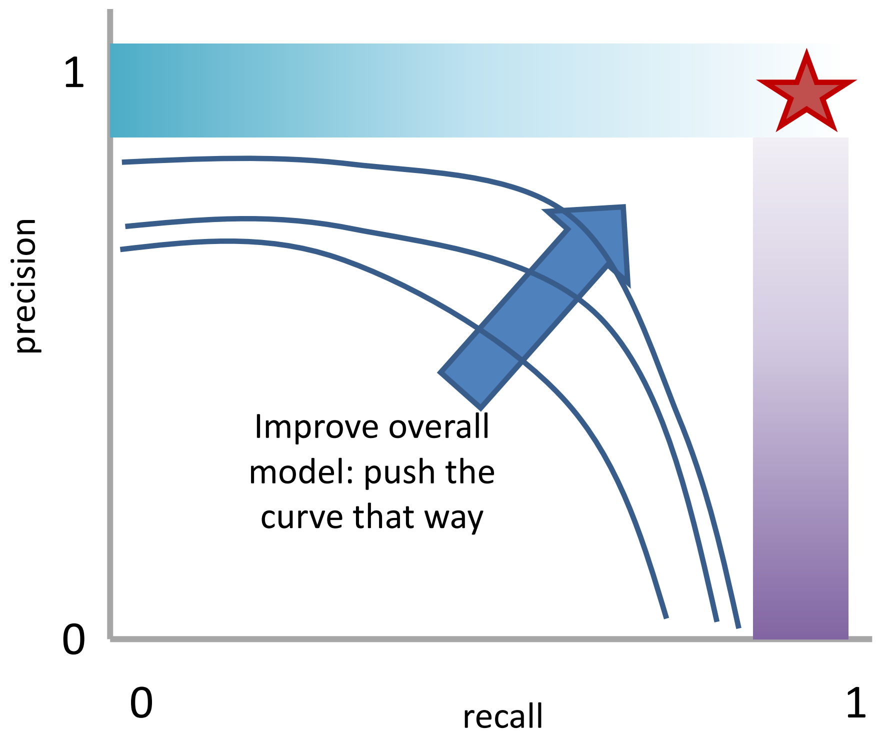
AUC measures the area under this tradeoff curve

1. Computing the curve

   You need true labels & predicted labels with some score/confidence estimate

   Threshold the scores and for each threshold compute precision and recall

Measure this Tradeoff: Area Under the Curve (AUC)



Min AUC: 0 ☹

Max AUC: 1 😀

AUC measures the area under this tradeoff curve

1. Computing the curve

   You need true labels & predicted labels with some score/confidence estimate

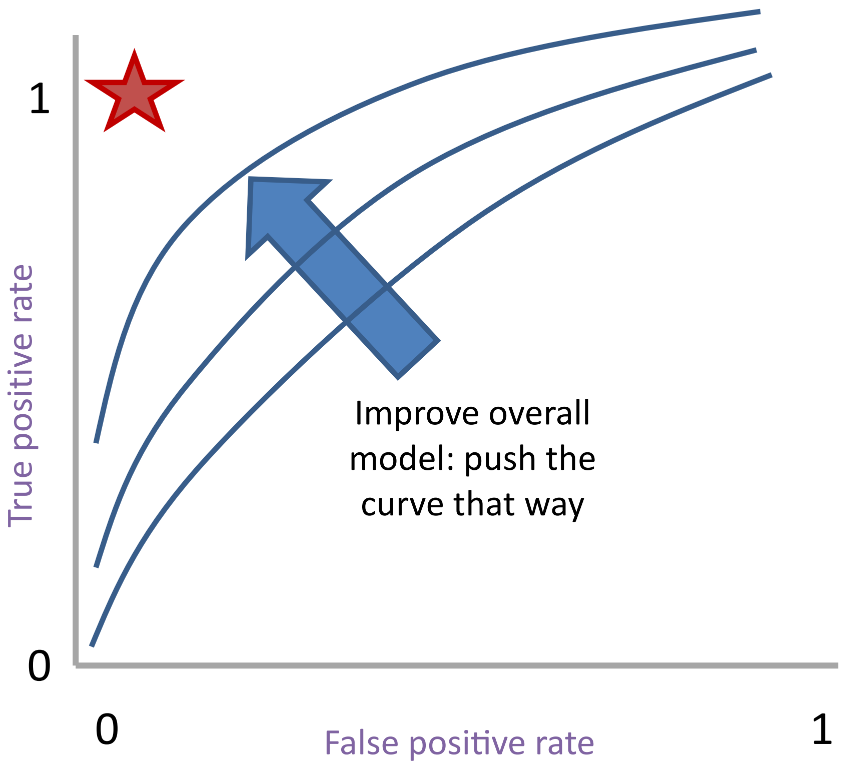
   Threshold the scores and for each threshold compute precision and recall

2. Finding the area

   How to implement: trapezoidal rule (& others)

In practice: external library like the sklearn.metrics module

Measure A Slightly Different Tradeoff: ROC-AUC



Min ROC-AUC: 0.5 ☹️
Max ROC-AUC: 1 😀

AUC measures the area under this tradeoff curve

1. Computing the curve
   - You need true labels & predicted labels with some score/confidence estimate
   - Threshold the scores and for each threshold compute metrics
2. Finding the area
   - How to implement: trapezoidal rule (& others)

In practice: external library like the sklearn.metrics module

Main variant: ROC-AUC

Same idea as before but with some flipped metrics

A combined measure: F

Weighted (harmonic) average of **P**recision & **R**ecall

$$F = \frac{1}{\alpha \frac{1}{P} + (1 - \alpha) \frac{1}{R}}$$

A combined measure: F

Weighted (harmonic) average of **P**recision & **R**ecall

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(1+\beta^2) \;*P\; *R}{(\beta^2 * P) + R}$$

algebra
(not important)

A combined measure: F

Weighted (harmonic) average of **P**recision & **R**ecall

$$F = \frac{(1+\beta^2) \ * P \ * R}{(\beta^2 * P) + R}$$

Balanced F1 measure: β=1

$$F_1 = \frac{2 \ * P \ * R}{P + R}$$

P/R/F in a Multi-class Setting: Micro- vs. Macro-Averaging

If we have more than one class, how do we combine multiple performance measures into one quantity?

Macroaveraging: Compute performance for each class, then average.

Microaveraging: Collect decisions for all classes, compute contingency table, evaluate.

P/R/F in a Multi-class Setting: Micro- vs. Macro-Averaging

Macroaveraging: Compute performance for each class, then average.

$$\text{macroprecision} = \sum_{c} \frac{\mathrm{TP_c}}{\mathrm{TP_c} + \mathrm{FP_c}} = \sum_{c} \text{precision}_c$$

Microaveraging: Collect decisions for all classes, compute contingency table, evaluate.

$$\text{microprecision} = \frac{\sum_{\mathrm{c}} \mathrm{TP_c}}{\sum_{\mathrm{c}} \mathrm{TP_c} + \sum_{\mathrm{c}} \mathrm{FP_c}}$$

P/R/F in a Multi-class Setting: Micro- vs. Macro-Averaging

Macroaveraging: Compute performance for each class, then average.

when to prefer the macroaverage?

$$\text{macroprecision} = \sum_c \frac{\mathrm{TP_c}}{\mathrm{TP_c} + \mathrm{FP_c}} = \sum_c \text{precision}_c$$

Microaveraging: Collect decisions for all classes, compute contingency table, evaluate.

when to prefer the microaverage?

$$\text{microprecision} = \frac{\sum_c \mathrm{TP_c}}{\sum_c \mathrm{TP_c} + \sum_c \mathrm{FP_c}}$$

Micro- vs. Macro-Averaging: Example

Class 1

	Truth : yes	Truth : no
Classifier: yes	10	10
Classifier: no	10	970

Class 2

	Truth : yes	Truth : no
Classifier: yes	90	10
Classifier: no	10	890

Micro Ave. Table

	Truth : yes	Truth : no
Classifier: yes	100	20
Classifier: no	20	1860

Macroaveraged precision: (0.5 + 0.9)/2 = 0.7

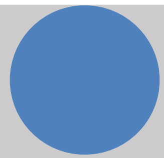
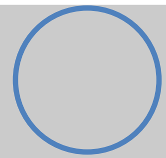
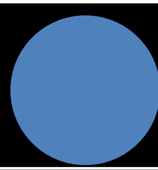
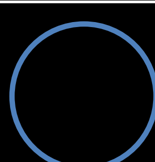
Microaveraged precision: 100/120 = .83

Microaveraged score is dominated by score on frequent classes

Confusion Matrix: Generalizing the 2-by-2 contingency table

		Correct Value		
Guessed Value		#	#	#
		#	#	#
		#	#	#

Confusion Matrix: Generalizing the 2-by-2 contingency table

		Correct Value		
				
Guessed Value		80	9	11
		7	86	7
		2	8	9

Q: Is this a good result?

Confusion Matrix: Generalizing the 2-by-2 contingency table

		Correct Value		
		●	○	▭
Guessed Value	●	30	40	30
	○	25	30	50
	▭	30	35	35

Q: Is this a good result?

Confusion Matrix: Generalizing the 2-by-2 contingency table

		Correct Value		
		●	○	▭
Guessed Value	●	7	3	90
	○	4	8	88
	▭	3	7	90

Q: Is this a good result?