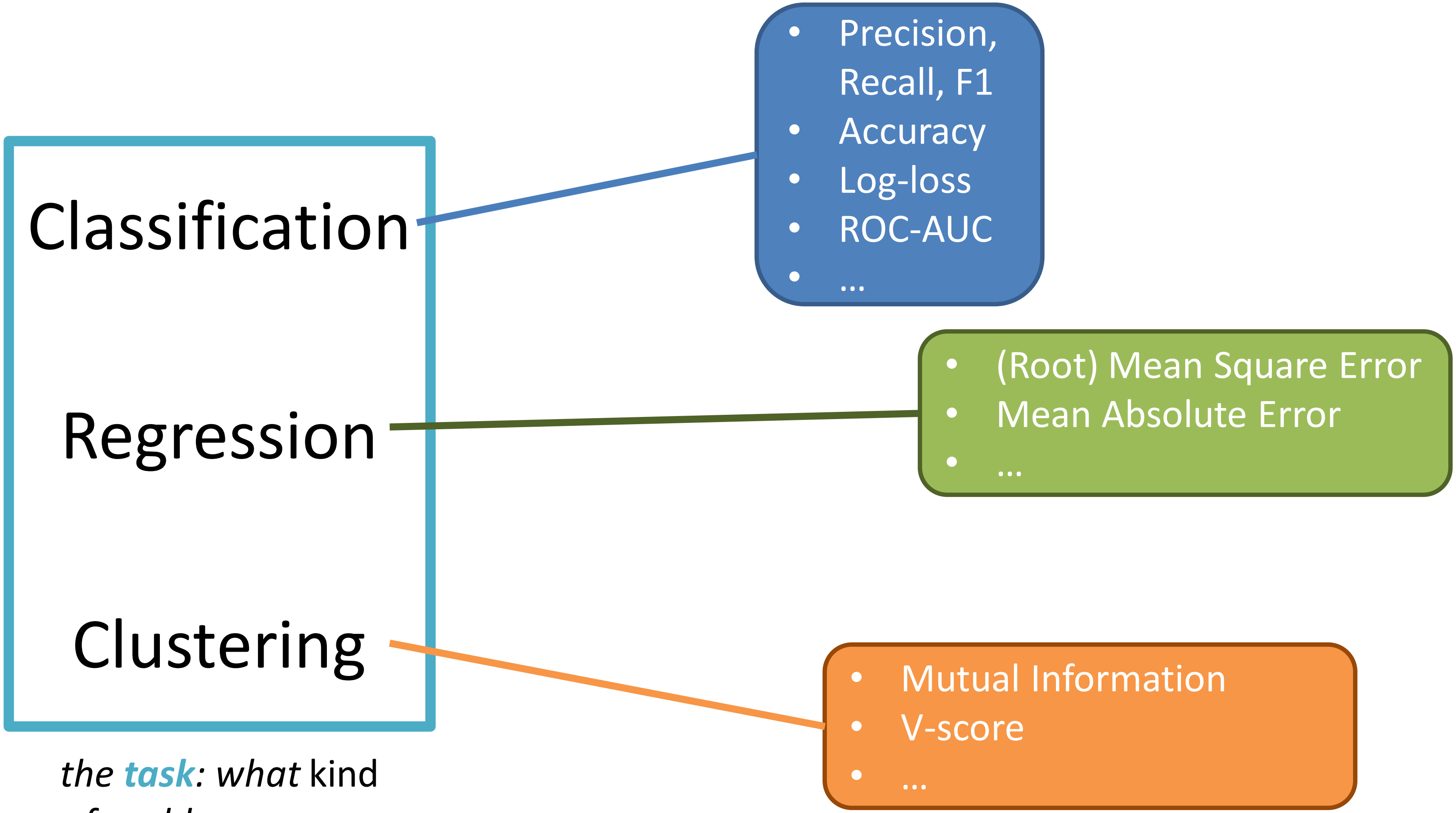
Experimental Setup,
Multi-class vs. Multi-label classification,
and
Evaluation

CMSC 678

UMBC

Central Question: How Well Are We Doing?



Classification

- Precision, Recall, F1
- Accuracy
- Log-loss
- ROC-AUC
- …

Regression

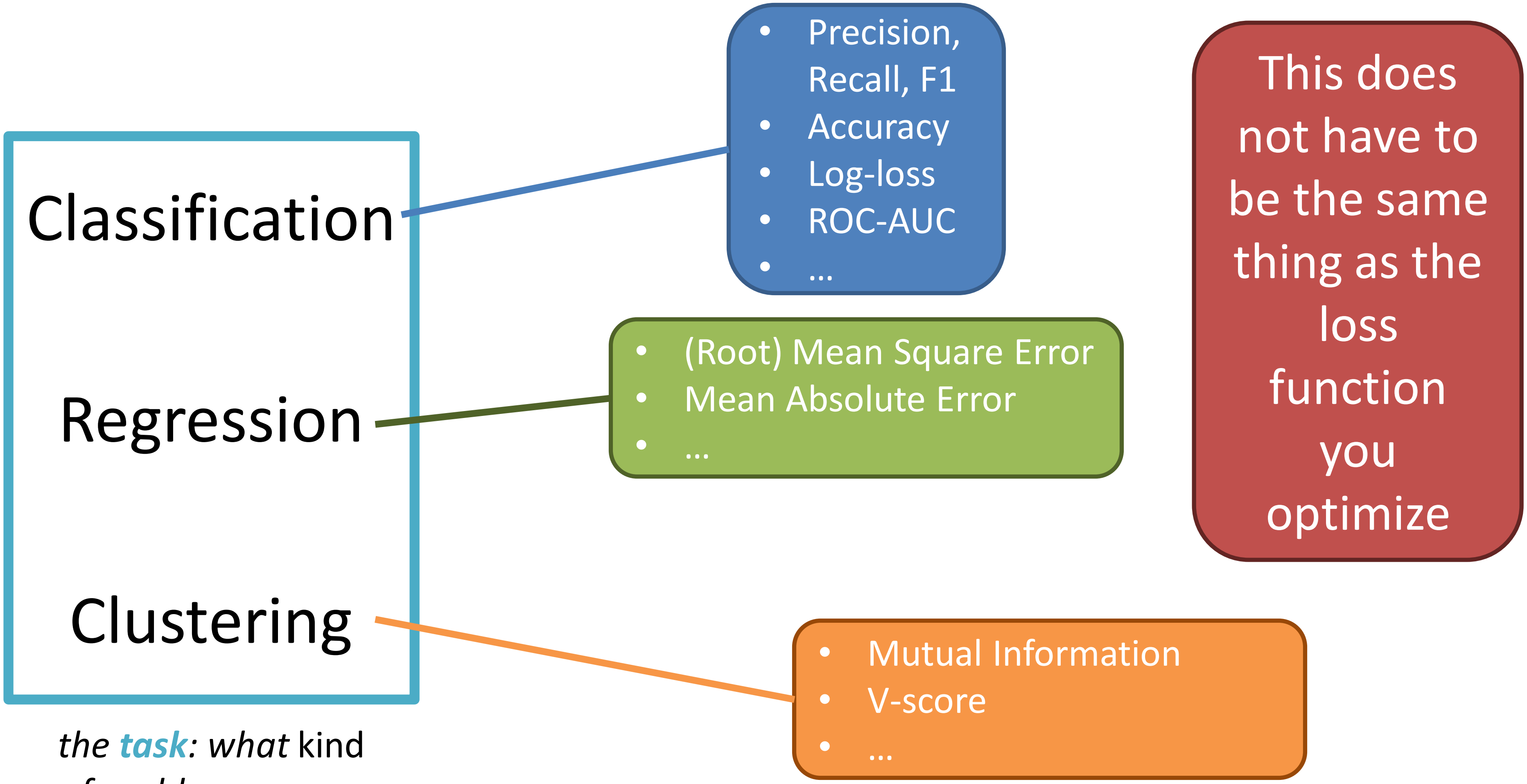
- (Root) Mean Square Error
- Mean Absolute Error
- …

Clustering

- Mutual Information
- V-score
- …

the ***task****: what* kind *of problem are you solving?*

Central Question: How Well Are We Doing?



Classification

Regression

Clustering

- Precision, Recall, F1
- Accuracy
- Log-loss
- ROC-AUC
- …

- (Root) Mean Square Error
- Mean Absolute Error
- …

- Mutual Information
- V-score
- …

This does not have to be the same thing as the loss function you optimize

the ***task****: what* kind *of problem are you solving?*

Outline

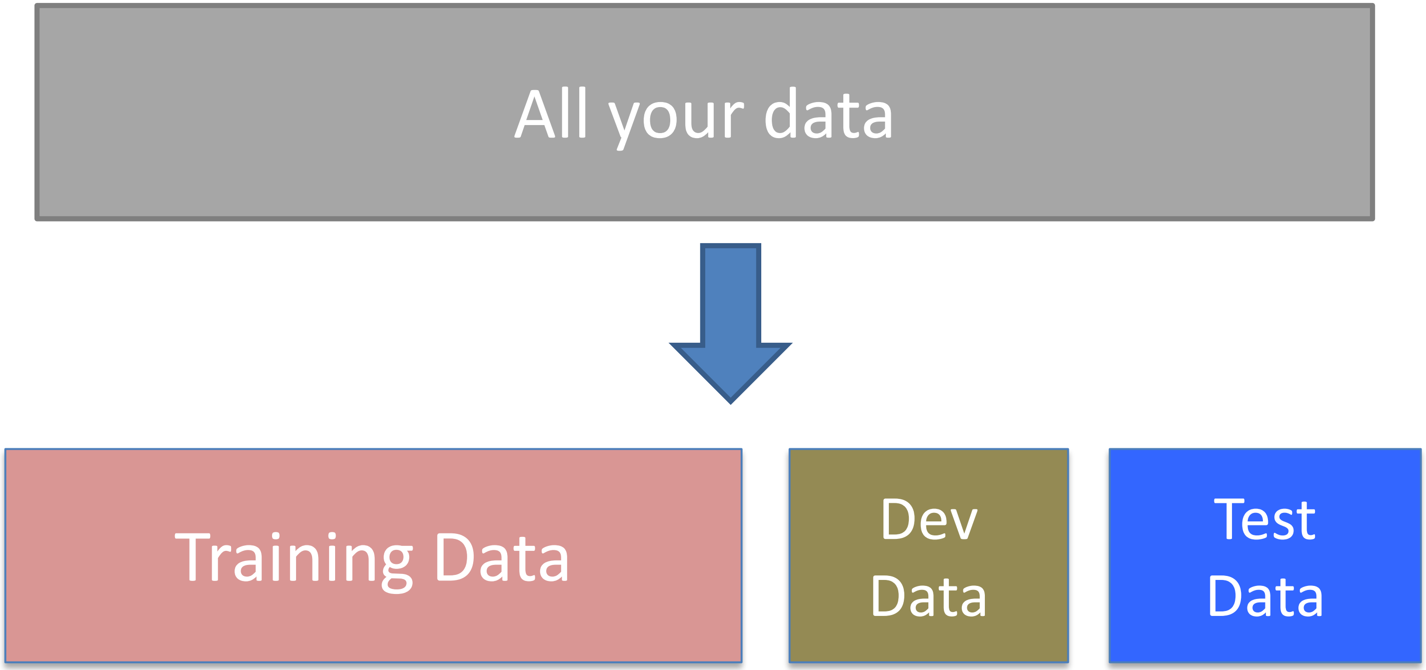
Experimental Design: Rule 1

Multi-class vs. Multi-label classification

Evaluation

- Regression Metrics
- Classification Metrics

Experimenting with Machine Learning Models



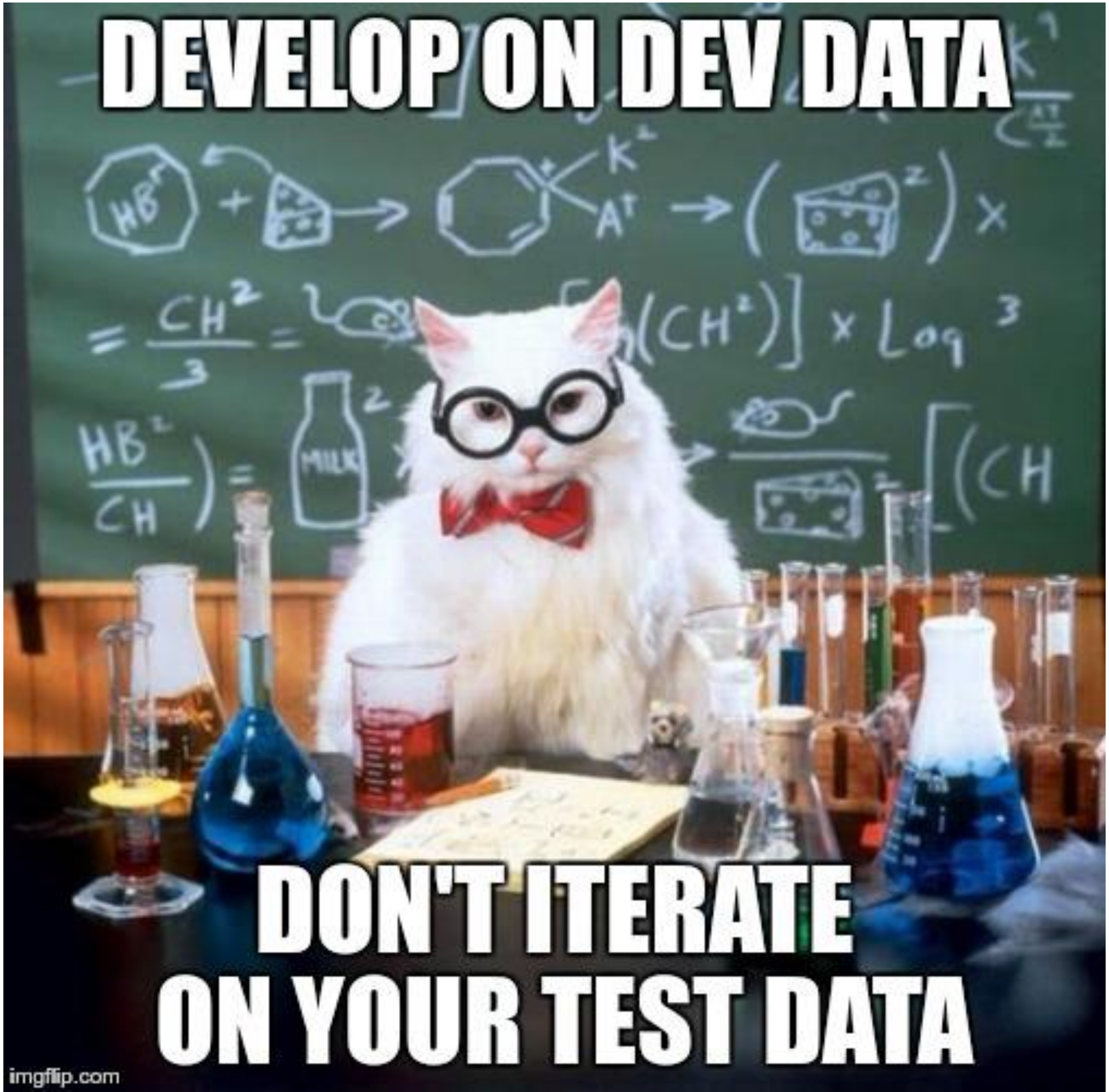
All your data

Training Data

Dev Data

Test Data

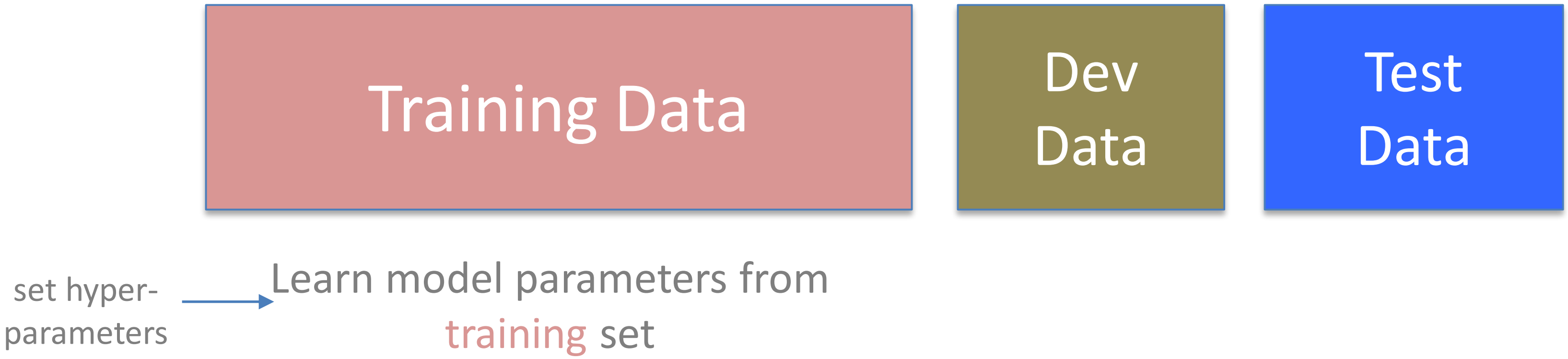
Rule #1


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

Experimenting with Machine Learning Models

What is "correct?"

What is working "well?"



Training Data

Dev Data

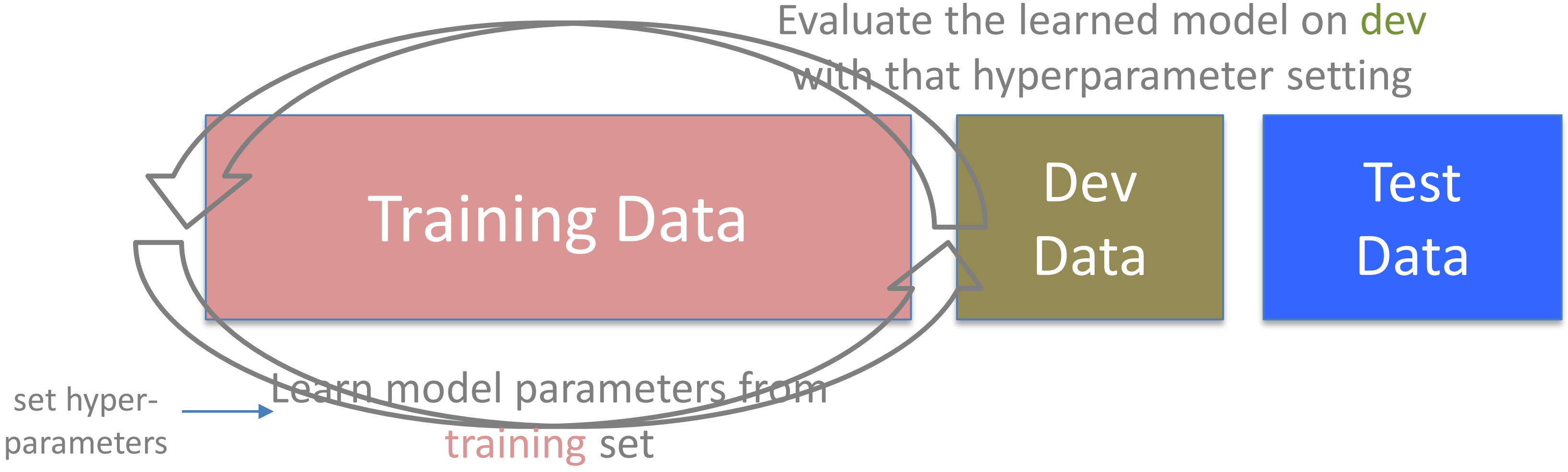
Test Data

set hyper-parameters → Learn model parameters from training set

Experimenting with Machine Learning Models

What is "correct?"

What is working "well?"



Evaluate the learned model on dev with that hyperparameter setting

Training Data

Dev Data

Test Data

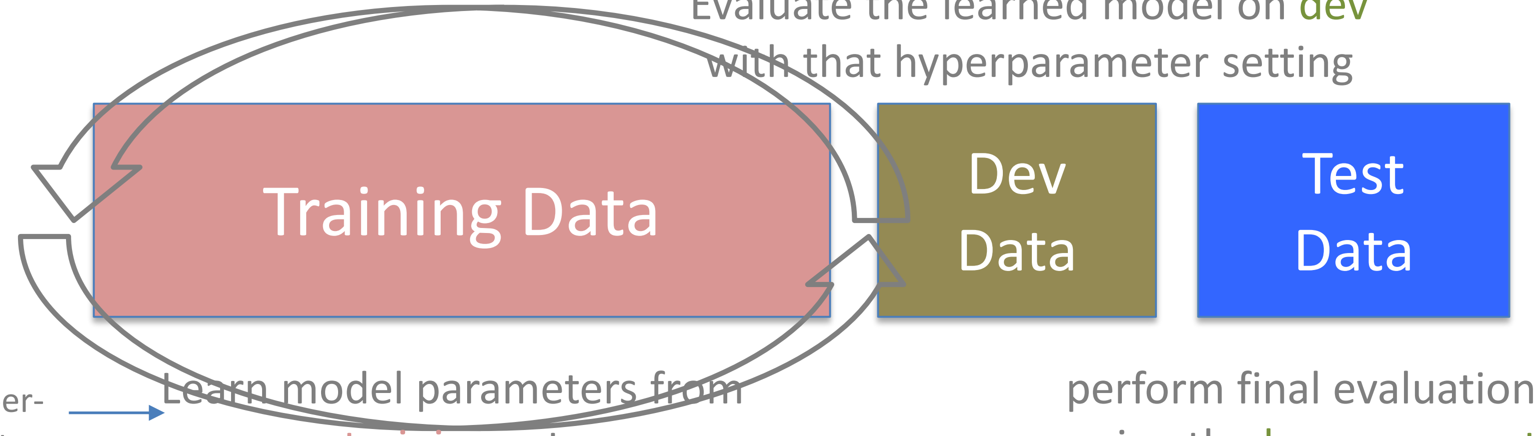
set hyper-parameters

Learn model parameters from training set

Experimenting with Machine Learning Models

What is "correct?"

What is working "well?"



Evaluate the learned model on dev with that hyperparameter setting

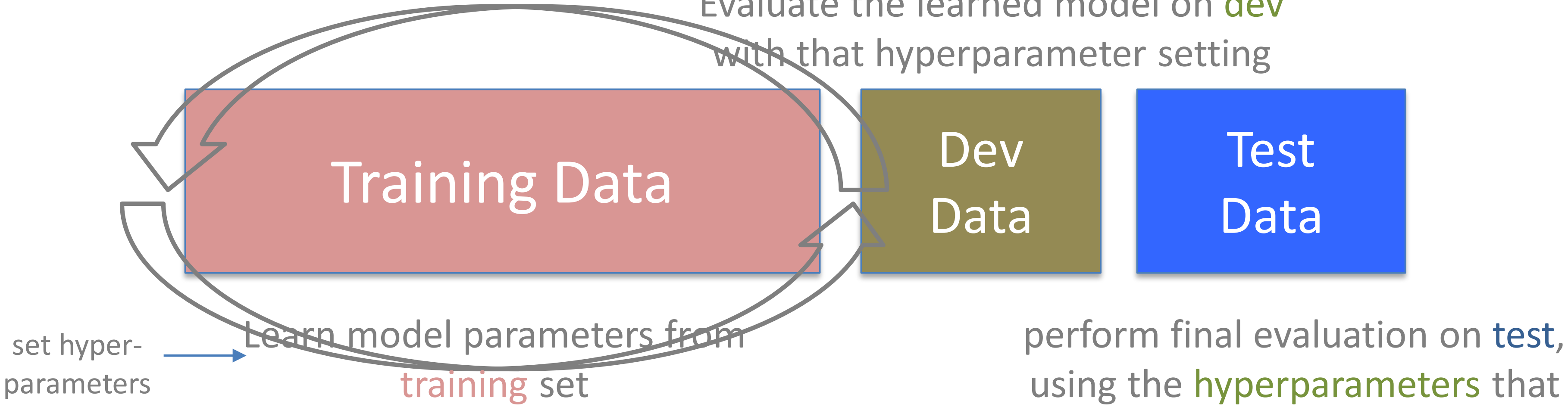
set hyper-parameters → Learn model parameters from training set

perform final evaluation on test, using the hyperparameters that optimized dev performance and *retraining* the model

Experimenting with Machine Learning Models

What is "correct?"

What is working "well?"



Evaluate the learned model on dev with that hyperparameter setting

set hyper-parameters

Learn model parameters from training set

perform final evaluation on test, using the hyperparameters that optimized dev performance and *retraining* the model

Rule 1: DO NOT ITERATE ON THE TEST DATA

On-board Exercise

Produce dev and test tables for a linear regression model with learned weights and set/fixed (non-learned) bias

Outline

Experimental Design: Rule 1

Multi-class vs. Multi-label classification

Evaluation

- Regression Metrics
- Classification Metrics

Multi-class Classification

Given input x, predict discrete label y

Multi-label Classification

Multi-class Classification

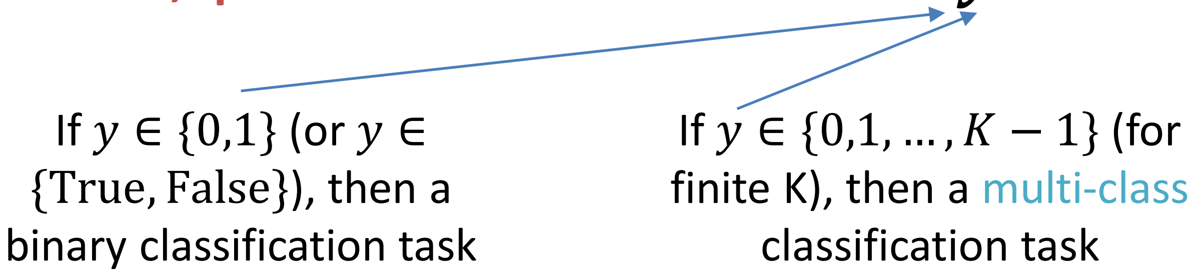
Given input x, predict discrete label y

If $y \in \{0,1\}$ (or $y \in \{\text{True}, \text{False}\}$), then a binary classification task

Multi-label Classification

Multi-class Classification

Given input x, predict discrete label y



If $y \in \{0,1\}$ (or $y \in \{\text{True}, \text{False}\}$), then a binary classification task

If $y \in \{0,1, \ldots, K-1\}$ (for finite K), then a multi-class classification task

Q: What are some examples of multi-class classification?

Multi-label Classification

Multi-class Classification

Given input x, predict discrete label y

If $y \in \{0,1\}$ (or $y \in \{\text{True}, \text{False}\}$), then a binary classification task

If $y \in \{0,1, \dots, K-1\}$ (for finite K), then a multi-class classification task

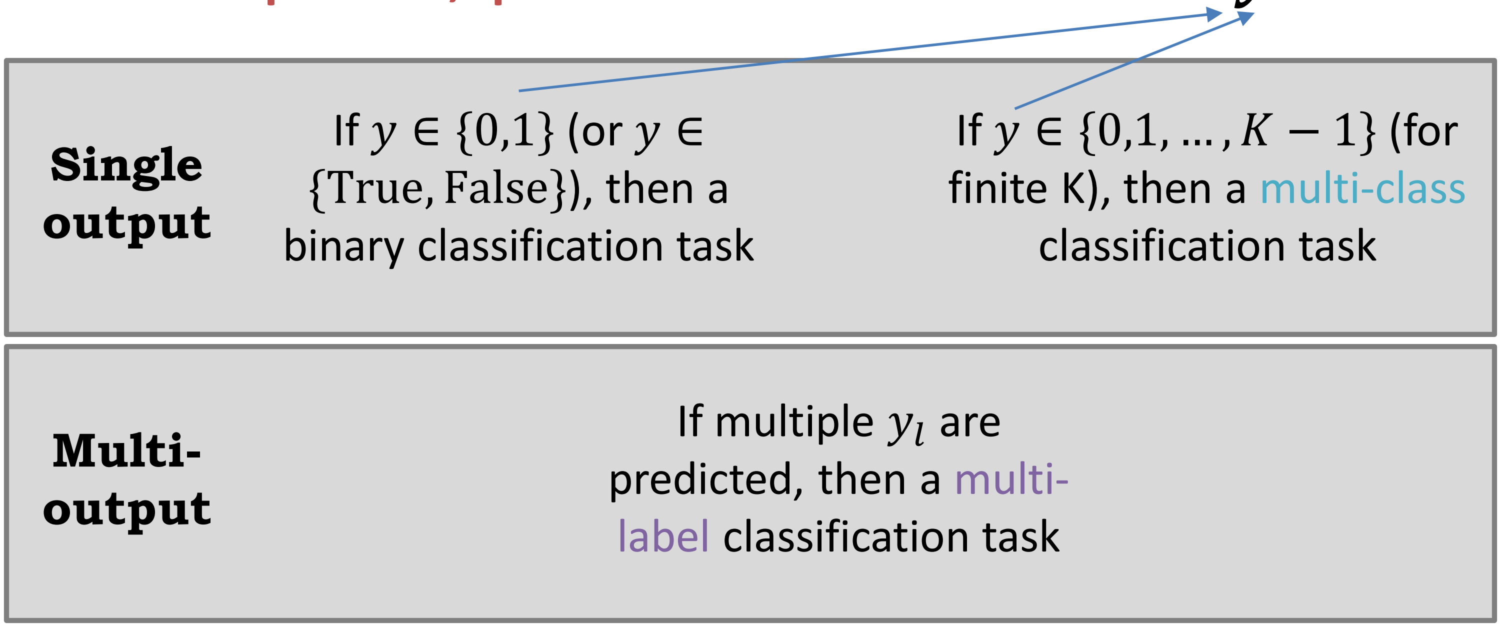
Q: What are some examples of multi-class classification?

A: Many possibilities. See A2, Q{1,2,4-7}

Multi-label Classification

Multi-class Classification

Given input x, predict discrete label y



Single output

If $y \in \{0,1\}$ (or $y \in \{\text{True}, \text{False}\}$), then a binary classification task

If $y \in \{0,1,\ldots,K-1\}$ (for finite K), then a multi-class classification task

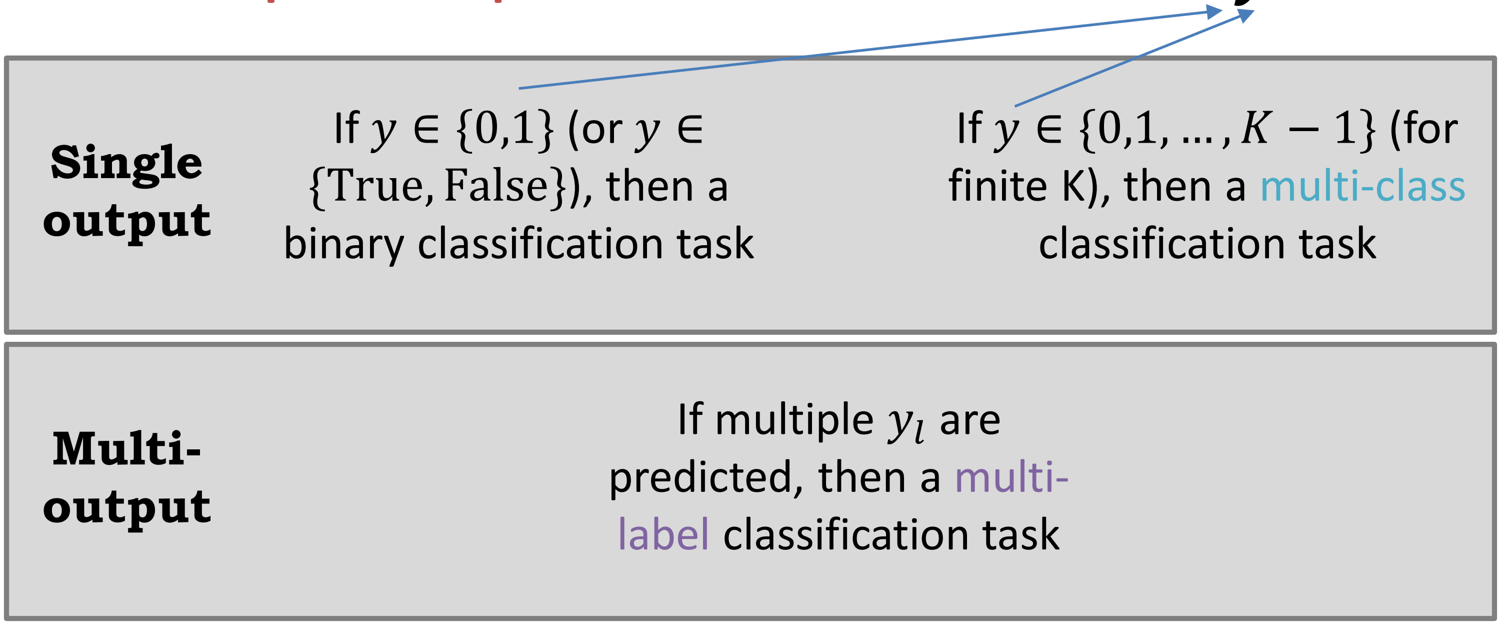
Multi-output

If multiple y_l are predicted, then a multi-label classification task

Multi-label Classification

Multi-class Classification

Given input x, predict discrete label y



Single output	If $y \in \{0,1\}$ (or $y \in \{\text{True}, \text{False}\}$), then a binary classification task	If $y \in \{0,1,\ldots,K-1\}$ (for finite K), then a multi-class classification task
Multi-output	If multiple y_l are predicted, then a multi-label classification task	

Given input x, predict multiple discrete labels $y = (y_1, \ldots, y_L)$

Multi-label Classification

Multi-class Classification

Given input x, predict discrete label y

Single output	If $y \in \{0,1\}$ (or $y \in \{\text{True}, \text{False}\}$), then a binary classification task	If $y \in \{0,1,\ldots,K-1\}$ (for finite K), then a multi-class classification task
Multi-output	If multiple y_l are predicted, then a multi-label classification task	Each y_l could be binary or multi-class

Given input x, predict multiple discrete labels $y = (y_1, \ldots, y_L)$

Multi-label Classification

Multi-Label Classification…

Will not be a primary focus of this course

Many of the single output classification methods apply to multi-label classification

Predicting “in the wild” can be trickier

Evaluation can be trickier

We’ve only developed binary classifiers so far…

Option 1: Develop a multi-class version

Option 2: Build a one-vs-all (OvA) classifier

Option 3: Build an all-vs-all (AvA) classifier

(there can be others)

We’ve only developed binary classifiers so far…

Option 1: Develop a multi-class version

Option 2: Build a one-vs-all (OvA) classifier

Option 3: Build an all-vs-all (AvA) classifier

(there can be others)

Loss function may (or may not) need to be extended & the model structure may need to change (big or small)

We've only developed binary classifiers so far…

Option 1: Develop a multi-class version

Option 2: Build a one-vs-all (OvA) classifier

Option 3: Build an all-vs-all (AvA) classifier

(there can be others)

Loss function may (or may not) need to be extended & the model structure may need to change (big or small)

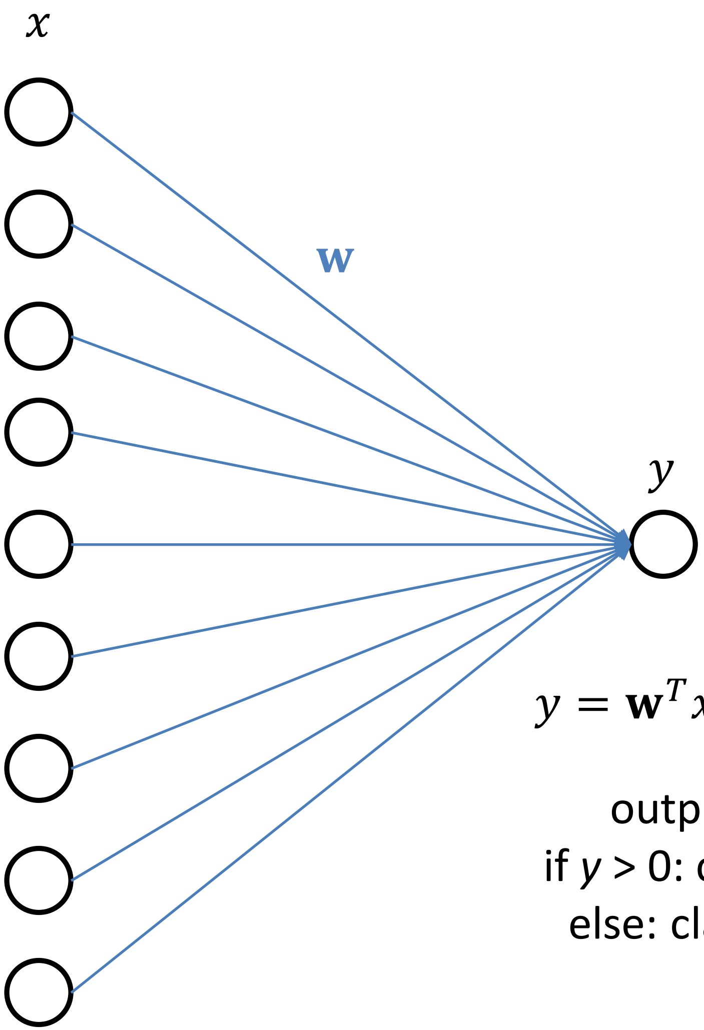
Common change:

instead of a single weight vector w, keep a weight vector $w^{(c)}$ for each class *c*

Compute class specific scores, e.g.,

$$\widehat{y_i^{(c)}} = \left(w^{(c)}\right)^T x + b^{(c)}$$

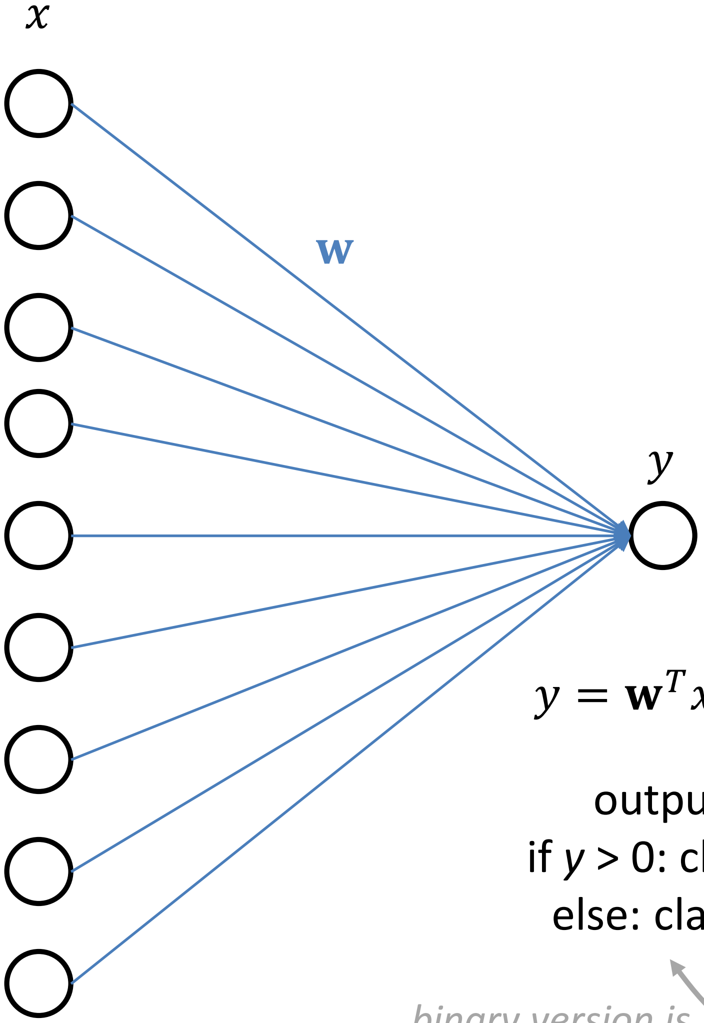
Multi-class Option 1: Linear Regression/Perceptron



$$y = \mathbf{w}^T x + b$$

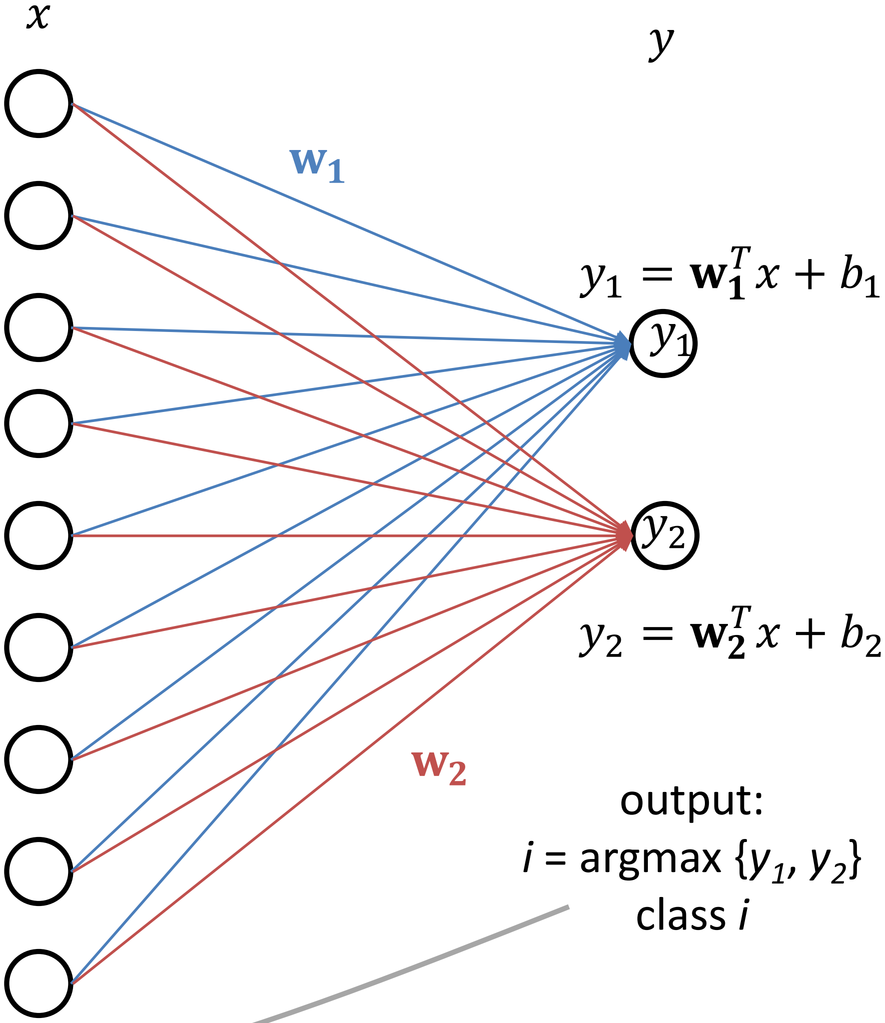
output:
if $y > 0$: class 1
else: class 2

Multi-class Option 1: Linear Regression/Perceptron: A Per-Class View



$$y = \mathbf{w}^T x + b$$

output:
if $y > 0$: class 1
else: class 2



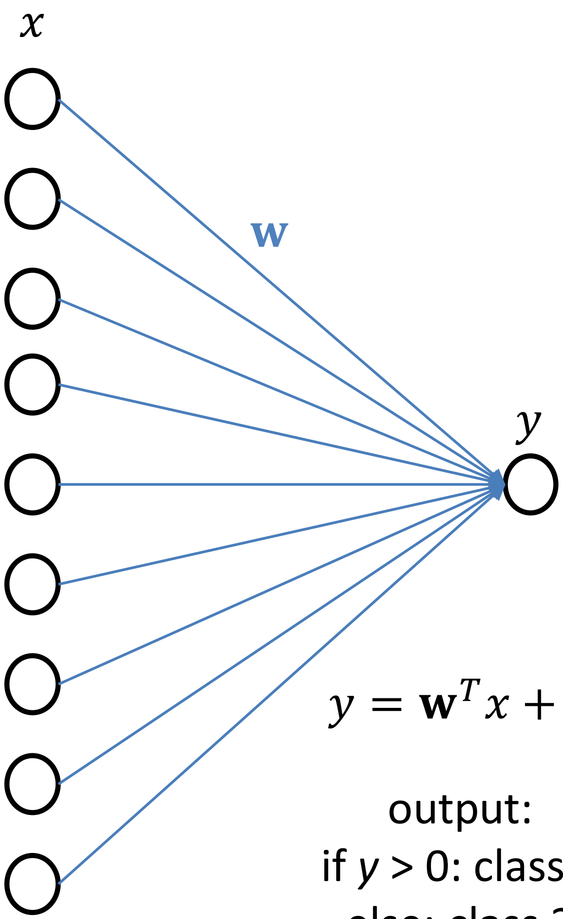
$$y_1 = \mathbf{w_1}^T x + b_1$$

$$y_2 = \mathbf{w_2}^T x + b_2$$

output:
i = argmax $\{y_1, y_2\}$
class i

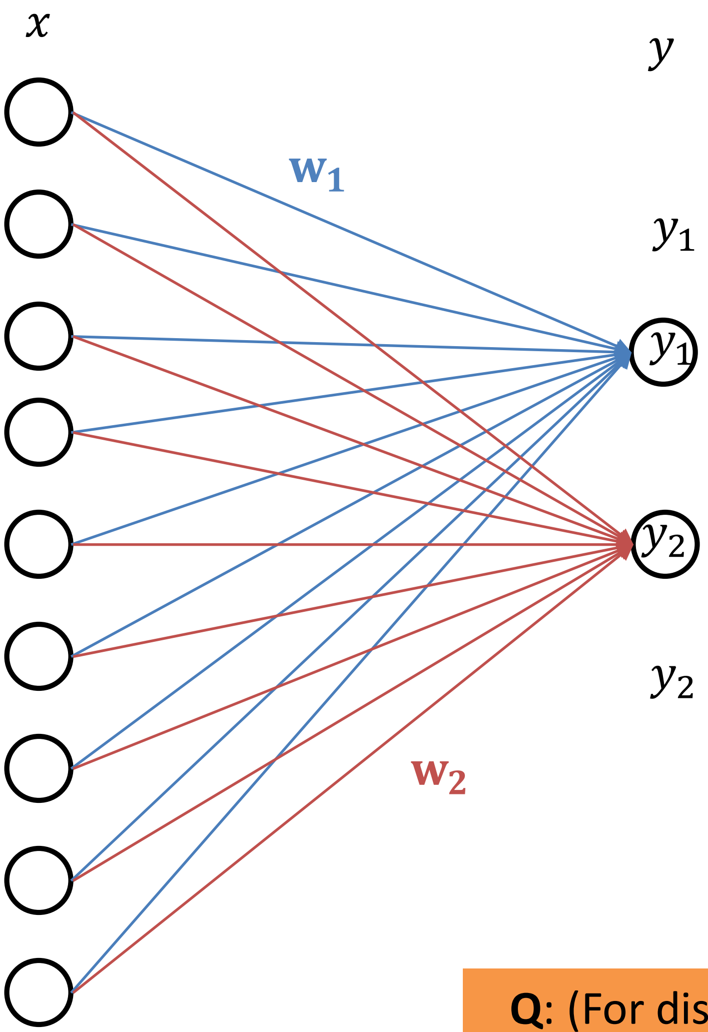
binary version is special case

Multi-class Option 1: Linear Regression/Perceptron: A Per-Class View (alternative)



$$y = \mathbf{w}^T x + b$$

output:
if y > 0: class 1
else: class 2



$$y_1 = [\boldsymbol{w_1}; \boldsymbol{w_2}]^{\boldsymbol{T}}[x; \mathbf{0}] + b_1$$

concatenation

$$y_2 = [\boldsymbol{w_1}; \boldsymbol{w_2}]^{\boldsymbol{T}}[\mathbf{0}; x] + b_2$$

output:
i = argmax $\{y_1, y_2\}$
class *i*

Q: (For discussion) Why does this work?

We’ve only developed binary classifiers so far…

Option 1: Develop a multi-class version

Option 2: Build a one-vs-all (OvA) classifier

Option 3: Build an all-vs-all (AvA) classifier

(there can be others)

With C classes:

Train C different binary classifiers $\gamma_c(x)$

$\gamma_c(x)$ predicts 1 if x is likely class c, 0 otherwise

We’ve only developed binary classifiers so far…

Option 1: Develop a multi-class version

Option 2: Build a one-vs-all (OvA) classifier

Option 3: Build an all-vs-all (AvA) classifier

(there can be others)

With C classes:

Train C different binary classifiers $\gamma_c(x)$

$\gamma_c(x)$ predicts 1 if x is likely class c, 0 otherwise

To test/predict a new instance *z*:

Get scores $s^c = \gamma_c(z)$

Output the max of these scores, $\hat{y} = \operatorname{argmax}_c s^c$

We’ve only developed binary classifiers so far…

Option 1: Develop a multi-class version

Option 2: Build a one-vs-all (OvA) classifier

Option 3: Build an all-vs-all (AvA) classifier

(there can be others)

With C classes:

Train $\binom{C}{2}$ different binary classifiers $\gamma_{c_1,c_2}(x)$

We’ve only developed binary classifiers so far…

Option 1: Develop a multi-class version

Option 2: Build a one-vs-all (OvA) classifier

Option 3: Build an all-vs-all (AvA) classifier

(there can be others)

With C classes:

Train $\binom{C}{2}$ different binary classifiers $\gamma_{c_1,c_2}(x)$

- $\gamma_{c_1,c_2}(x)$ predicts 1 if x is likely class c_1, 0 otherwise (likely class c_2)

We’ve only developed binary classifiers so far…

Option 1: Develop a multi-class version

Option 2: Build a one-vs-all (OvA) classifier

Option 3: Build an all-vs-all (AvA) classifier

(there can be others)

With C classes:

Train $\binom{C}{2}$ different binary classifiers $\gamma_{c_1,c_2}(x)$

- $\gamma_{c_1,c_2}(x)$ predicts 1 if x is likely class c_1, 0 otherwise (likely class c_2)

To test/predict a new instance *z*:

- Get scores or predictions $s^{c_1,c_2} = \gamma_{c_1,c_2}(z)$

We've only developed binary classifiers so far…

Option 1: Develop a multi-class version

Option 2: Build a one-vs-all (OvA) classifier

Option 3: Build an all-vs-all (AvA) classifier

(there can be others)

With C classes:

Train $\binom{C}{2}$ different binary classifiers $\gamma_{c_1,c_2}(x)$

$\gamma_{c_1,c_2}(x)$ predicts 1 if x is likely class c_1, 0 otherwise (likely class c_2)

To test/predict a new instance *z*:

Get scores or predictions $s^{c_1,c_2} = \gamma_{c_1,c_2}(z)$

Multiple options for final prediction:

(1) count # times a class *c* was predicted

(2) margin-based approach

We’ve only developed binary classifiers so far…

Option 1: Develop a multi-class version

Option 2: Build a one-vs-all (OvA) classifier

Option 3: Build an all-vs-all (AvA) classifier

(there can be others)

Q: (to discuss)

Why might you want to use option 1 or options OvA/AvA?

What are the benefits of OvA vs. AvA?

We’ve only developed binary classifiers so far…

Option 1: Develop a multi-class version

Option 2: Build a one-vs-all (OvA) classifier

Option 3: Build an all-vs-all (AvA) classifier

(there can be others)

Q: (to discuss)

Why might you want to use option 1 or options OvA/AvA?

What are the benefits of OvA vs. AvA?

What if you start with a balanced dataset, e.g., 100 instances per class?

Outline

Experimental Design: Rule 1

Multi-class vs. Multi-label classification

Evaluation

- Regression Metrics
- Classification Metrics

Regression Metrics

(Root) Mean Square Error

$$RMSE = \sqrt{\frac{1}{N}\sum_{i}^{N}(y_i - \widehat{y_i})^2}$$

Regression Metrics

(Root) Mean Square Error

$$RMSE = \sqrt{\frac{1}{N}\sum_{i}^{N}(y_i - \widehat{y_i})^2}$$

Mean Absolute Error

$$MAE = \frac{1}{N}\sum_{i}^{N}|y_i - \widehat{y_i}|$$

Regression Metrics

(Root) Mean Square Error

Mean Absolute Error

$$RMSE = \sqrt{\frac{1}{N}\sum_{i}^{N}(y_i - \widehat{y_i})^2}$$

$$MAE = \frac{1}{N}\sum_{i}^{N}|y_i - \widehat{y_i}|$$

Q: How can these reward/punish predictions differently?

Regression Metrics

(Root) Mean Square Error

Mean Absolute Error

$$RMSE = \sqrt{\frac{1}{N}\sum_{i}^{N}(y_i - \widehat{y_i})^2}$$

$$MAE = \frac{1}{N}\sum_{i}^{N}|y_i - \widehat{y_i}|$$

Q: How can these reward/punish predictions differently?

A: RMSE punishes outlier predictions more harshly

Outline

Experimental Design: Rule 1

Multi-class vs. Multi-label classification

Evaluation

- Regression Metrics
- Classification Metrics

Training Loss vs. Evaluation Score

In training, compute loss to update parameters

Sometimes loss is a computational compromise

- surrogate loss

The loss you use might not be as informative as you'd like

Binary classification: 90 of 100 training examples are +1, 10 of 100 are -1

Some Classification Metrics

Accuracy

Precision
Recall

AUC (Area Under Curve)

F1

Confusion Matrix

Classification Evaluation: the 2-by-2 contingency table

	Actually Correct	Actually Incorrect
Selected/ Guessed		
Not selected/ not guessed		

● ○

Classes/Choices

Classification Evaluation: the 2-by-2 contingency table

	Actually Correct	**Actually Incorrect**
Selected/ Guessed	True Positive ● *Correct* (TP) ● *Guessed*	
Not selected/ not guessed		

● ○

Classes/Choices

Classification Evaluation:
the 2-by-2 contingency table

	Actually Correct	**Actually Incorrect**
Selected/ Guessed	True Positive ● (TP) ● *Correct* *Guessed*	False Positive ○ (FP) ● *Correct* *Guessed*
Not selected/ not guessed		

● ○

Classes/Choices

Classification Evaluation: the 2-by-2 contingency table

	Actually Correct	**Actually Incorrect**
Selected/ Guessed	True Positive ● *Correct* (TP) ● *Guessed*	False Positive ○ *Correct* (FP) ● *Guessed*
Not selected/ not guessed	False Negative ● *Correct* (FN) ○ *Guessed*	

● ○

Classes/Choices

Classification Evaluation: the 2-by-2 contingency table

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

● ○

Classes/Choices

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

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

Precision: % of selected items that are correct

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

	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 😬

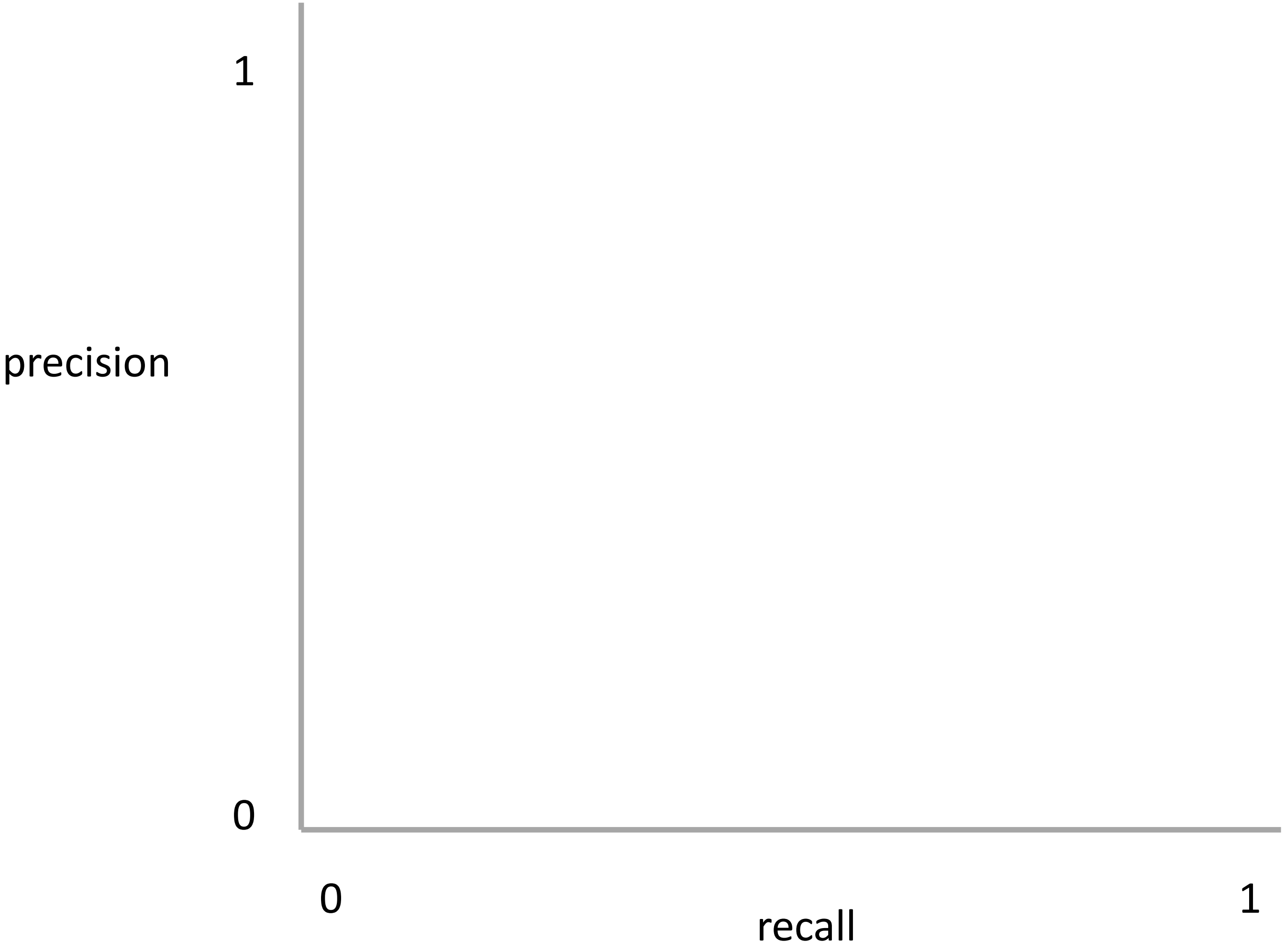
Recall: % of correct items that are selected

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

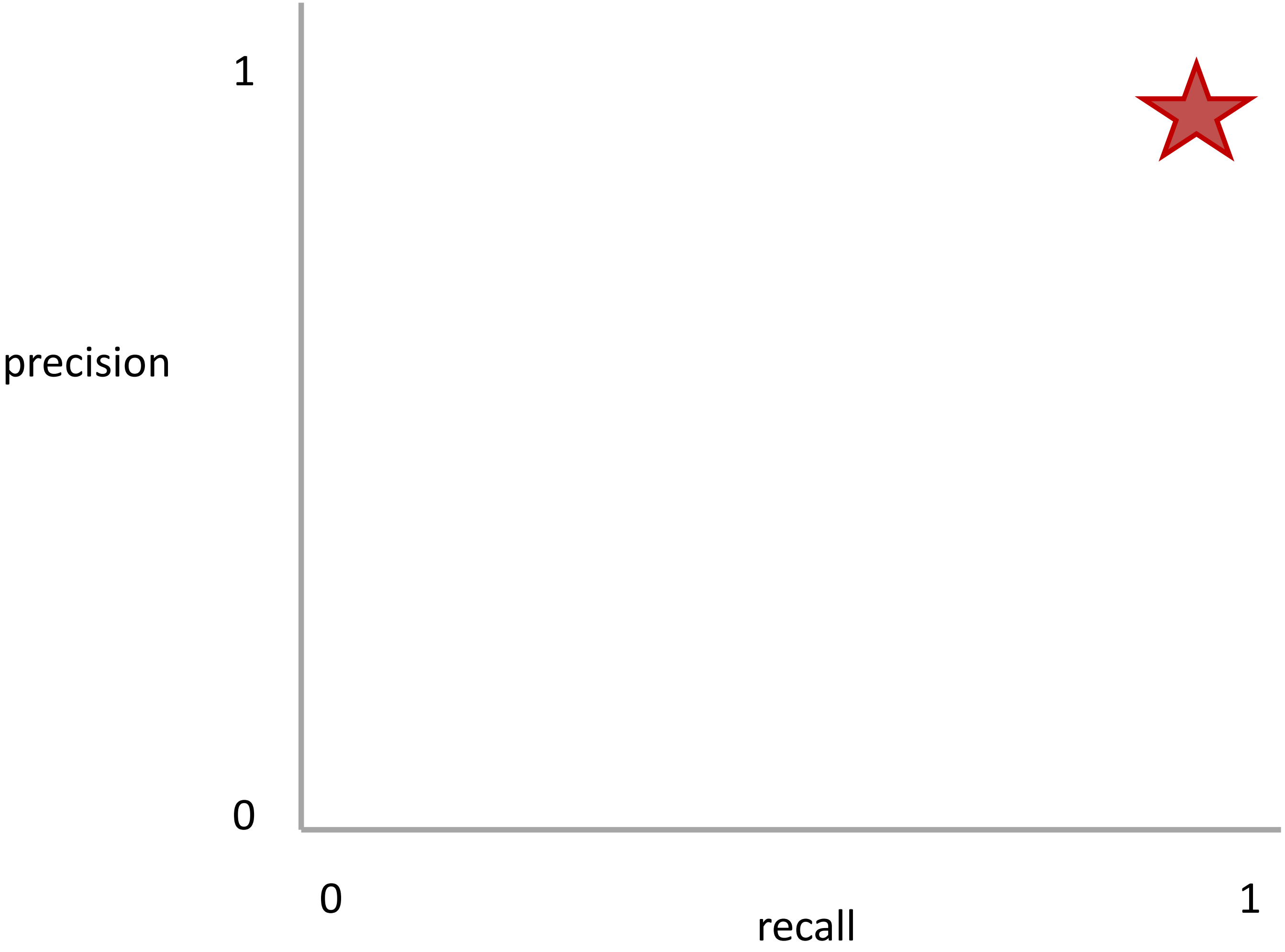
	Actually Correct	**Actually Incorrect**
Selected/Guessed	True Positive (TP)	False Positive (FP)
Not select/not guessed	False Negative (FN)	True Negative (TN)

Precision and Recall Present a Tradeoff

Q: Where do you want your ideal model ?



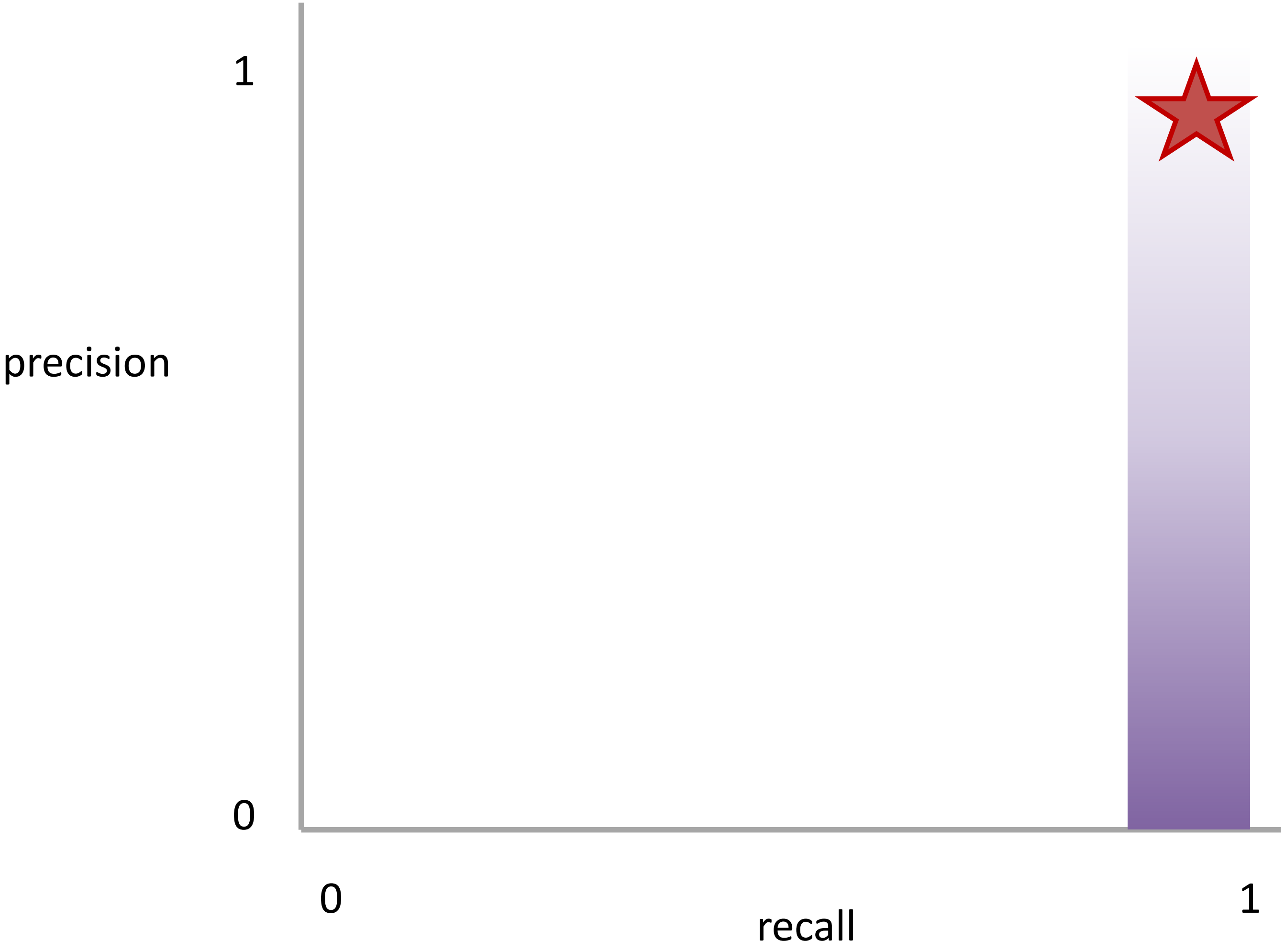
Precision and Recall Present a Tradeoff



Q: Where do you want your ideal model ?

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

Precision and Recall Present a Tradeoff

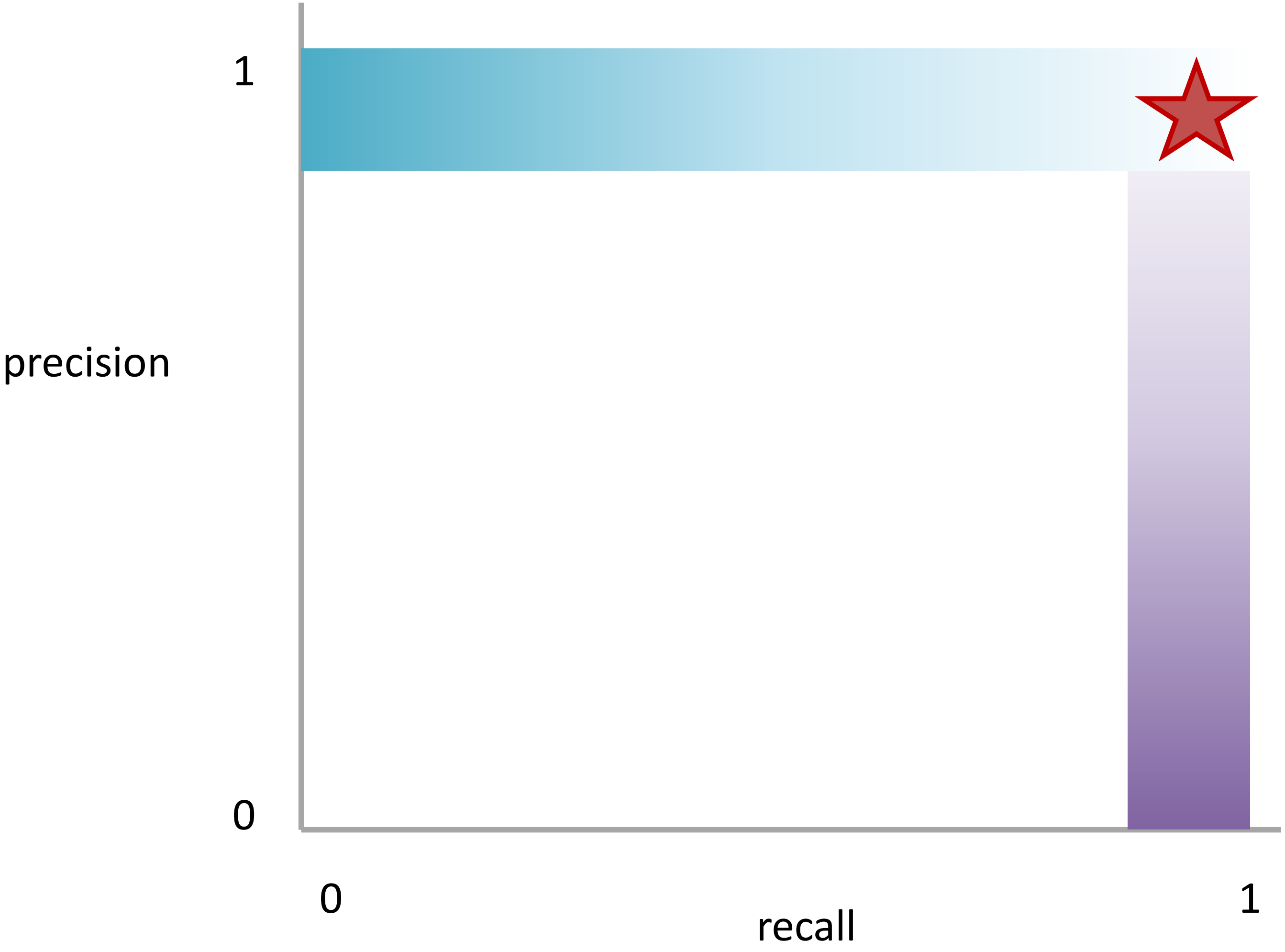


Q: Where do you want your ideal model ?

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

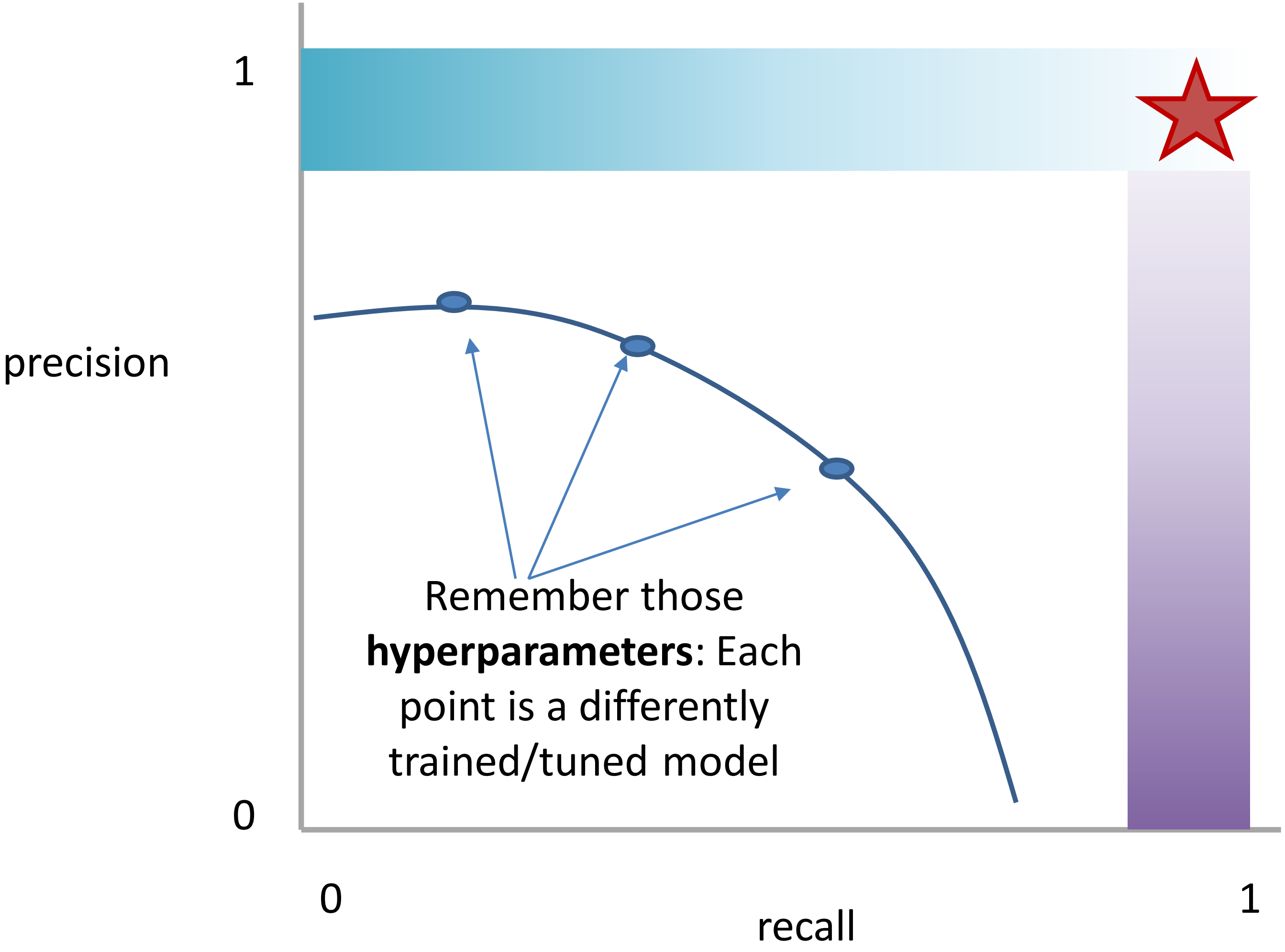


Q: Where do you want your ideal model ?

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



Remember those **hyperparameters**: Each point is a differently trained/tuned model

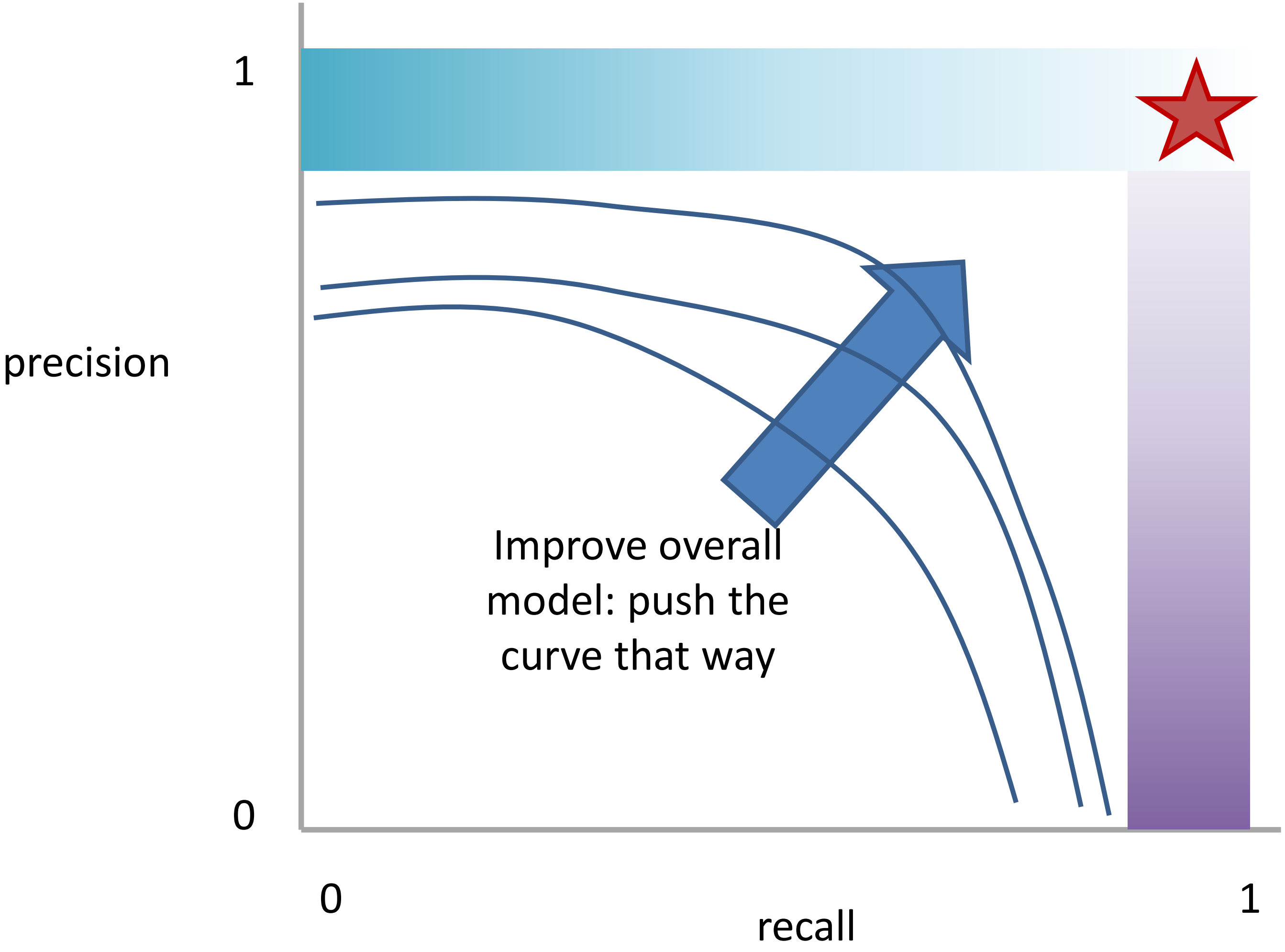
Q: Where do you want your ideal model ?

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



Improve overall model: push the curve that way

Q: Where do you want your ideal model ?

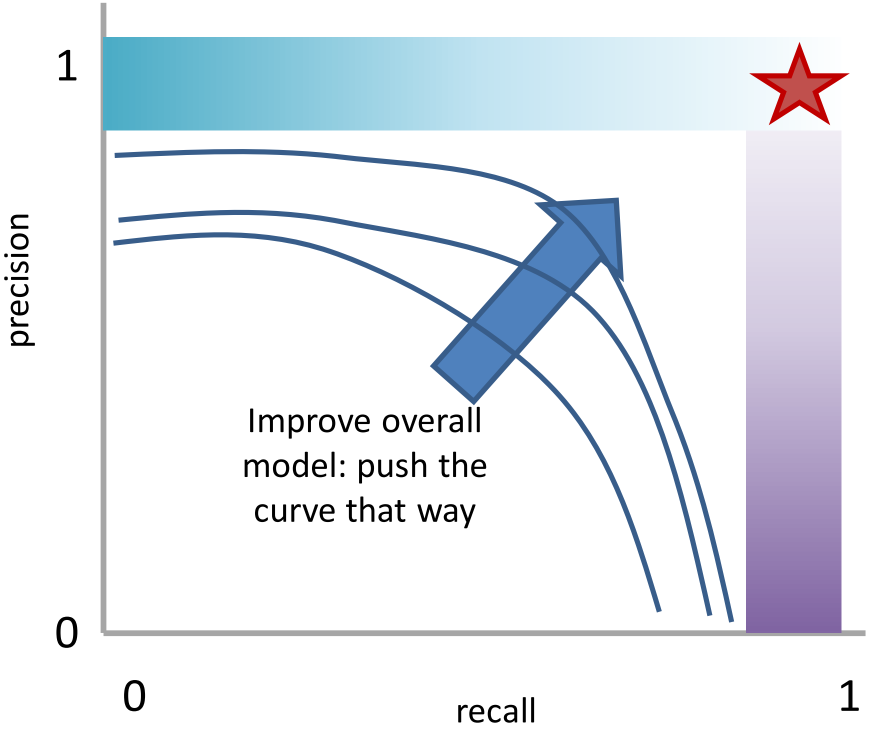
Q: You have a model that always identifies correct instances. Where on this graph is it?

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

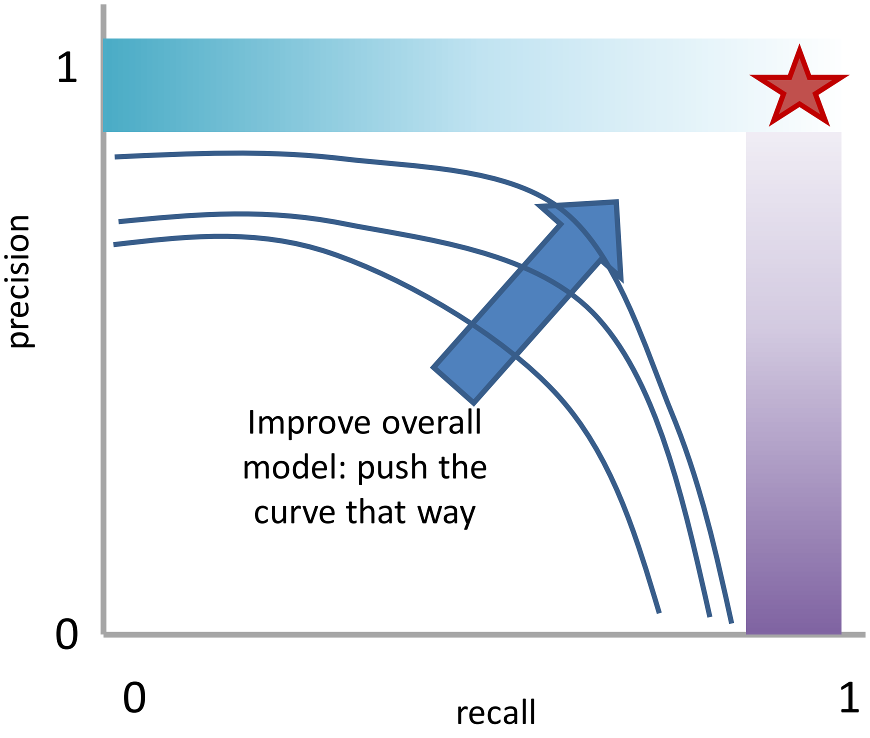


Improve overall model: push the curve that way

Min AUC: 0 ☹

Max AUC: 1 😁

Measure this Tradeoff: Area Under the Curve (AUC)



Min AUC: 0 ☹
Max AUC: 1 😁

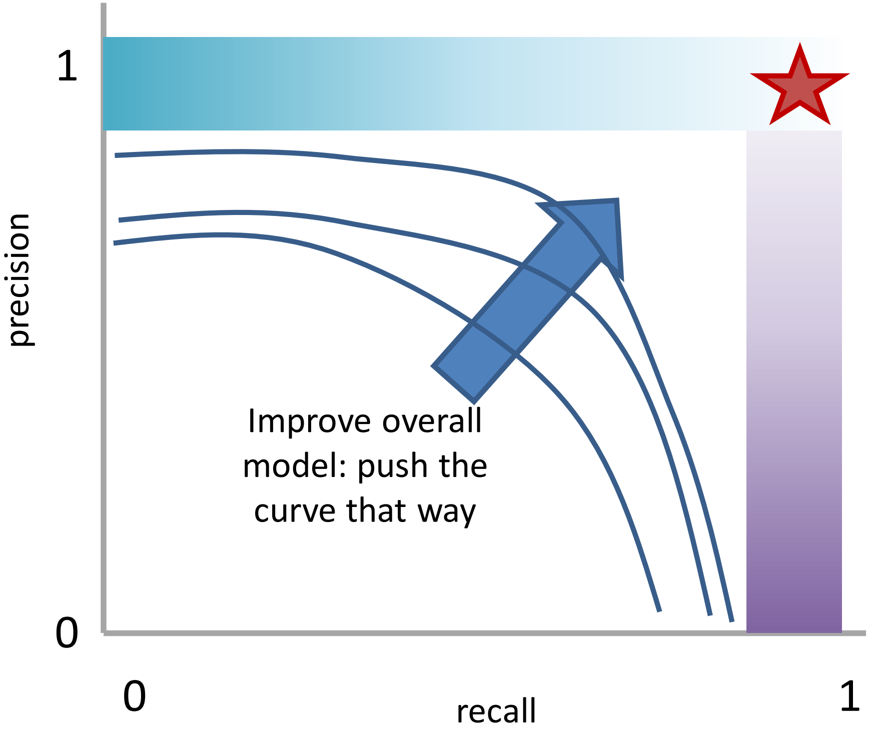
AUC measures the area under this tradeoff curve

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)



Improve overall model: push the curve that way

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

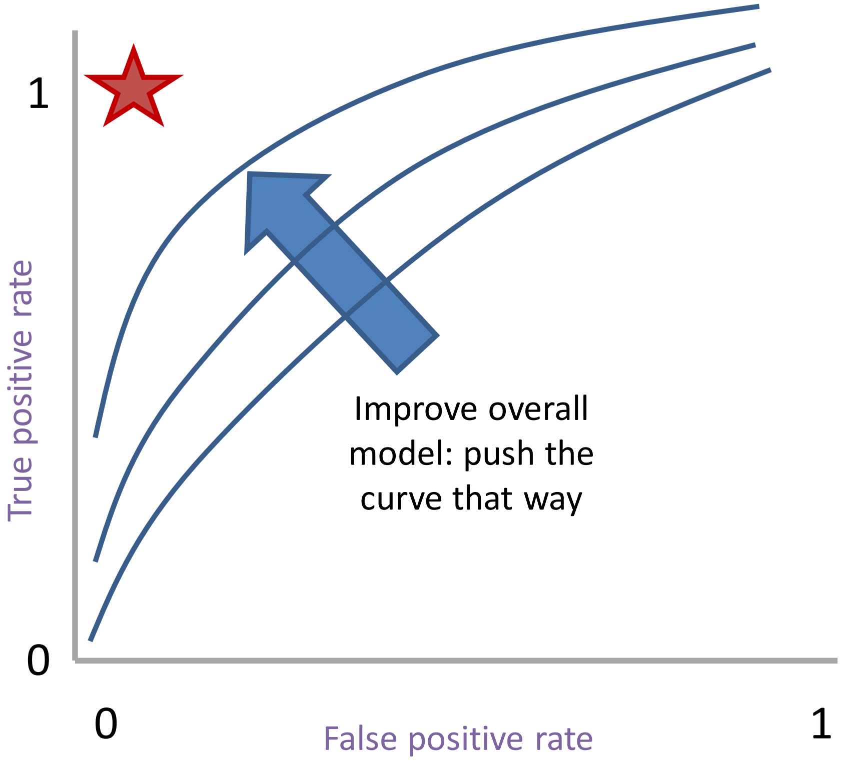
   Threshold the scores and for each threshold compute precision and recall

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?

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

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?

Some Classification Metrics

Accuracy

Precision
Recall

AUC (Area Under Curve)

F1

Confusion Matrix

Outline

Experimental Design: Rule 1

Multi-class vs. Multi-label classification

Evaluation

- Regression Metrics
- Classification Metrics