INFO 2950: Intro to Data Science

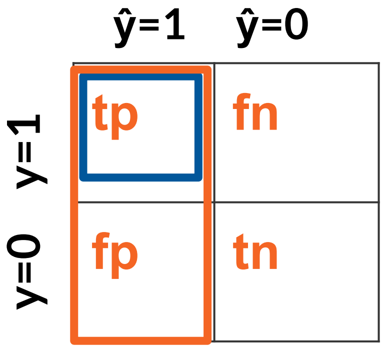
Lecture 14
2023-10-16

Agenda

1. **Binary Evaluation Metrics**
    a. Precision / Recall
    b. F1 / ROC-AUC

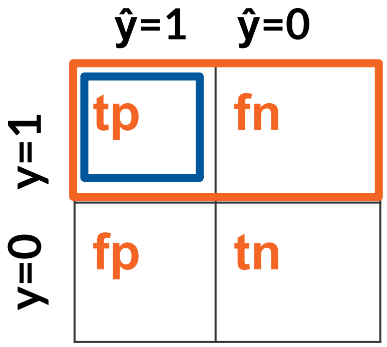
2. **Train / test sets**
    a. Cross Validation
    b. Sliding Windows

Precision and Recall



	ŷ=1	ŷ=0
y=1	tp	fn
y=0	fp	tn

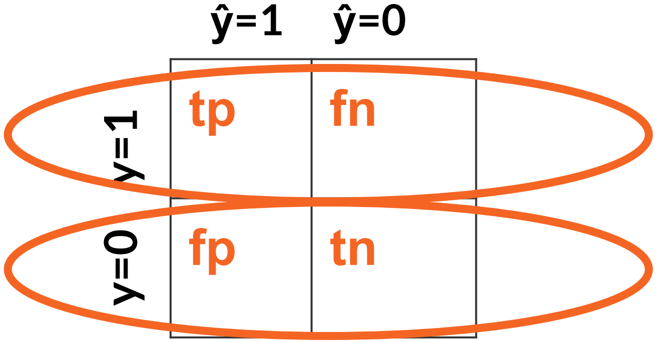
- Precision = tp / (tp + fp)



	ŷ=1	ŷ=0
y=1	tp	fn
y=0	fp	tn

- Recall = tp / (tp + fn)

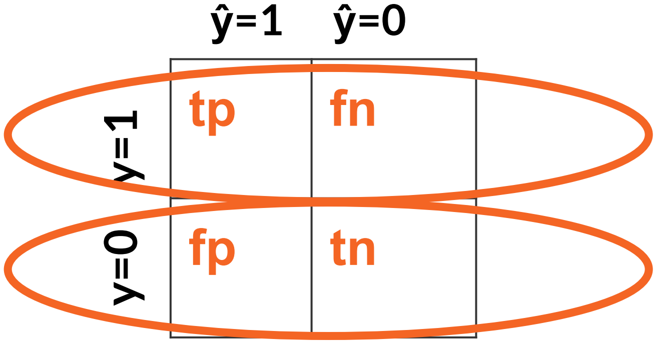
“True” Values Stay the Same



	ŷ=1	ŷ=0
y=1	tp	fn
y=0	fp	tn

- Positives = tp + fn
- Negatives = fp + tn

“True” Values Stay the Same

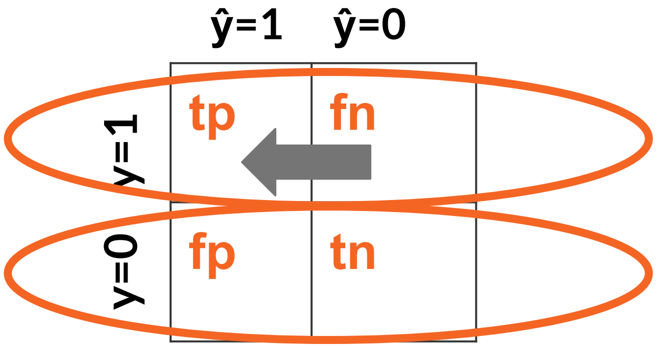


	ŷ=1	ŷ=0
y=1	tp	fn
y=0	fp	tn

- Positives = tp + fn
- Negatives = fp + tn

- If fn decreases, what happens to tp?

“True” Values Stay the Same

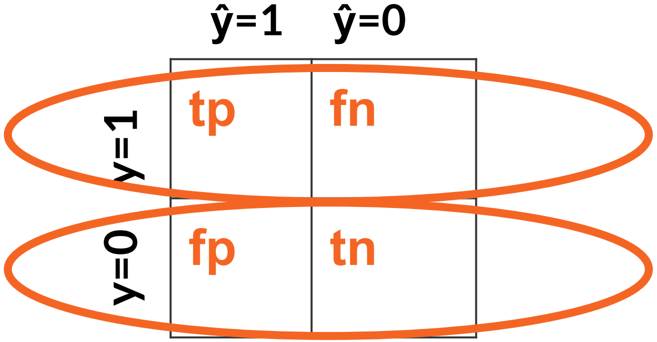


	ŷ=1	ŷ=0
y=1	tp	fn
y=0	fp	tn

- Positives = tp + fn
- Negatives = fp + tn

- If fn decreases, what happens to tp?
  - tp increases since total positives must stay the same

“True” Values Stay the Same

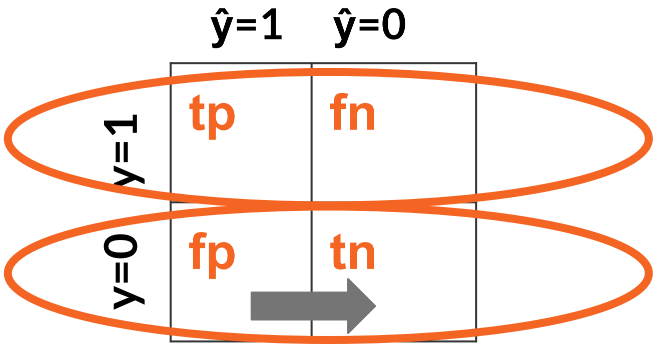


	ŷ=1	ŷ=0
y=1	tp	fn
y=0	fp	tn

- Positives = tp + fn
- Negatives = fp + tn

- If tn increases, what happens to fp?

“True” Values Stay the Same

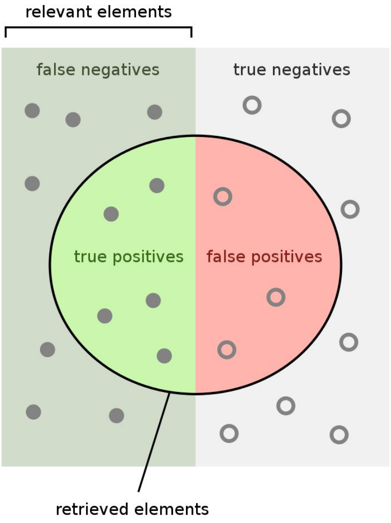


	ŷ=1	ŷ=0
y=1	tp	fn
y=0	fp	tn

- Positives = tp + fn
- Negatives = fp + tn

- If tn increases, what happens to fp?
  - fp decreases since total negatives must stay the same

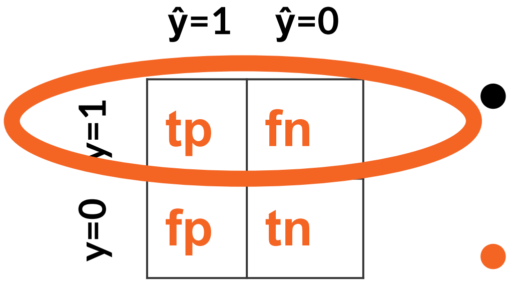
	ŷ=1	ŷ=0
y=1	tp	fn
y=0	fp	tn



How does recall change?

- Recall = tp / (tp + fn)
- If we change our model so the number of false negatives is decreased, what happens to recall?
    - Hint: let *p* represent the number of true y=1 values in your data

How does recall change?



	ŷ=1	ŷ=0
y=1	tp	fn
y=0	fp	tn

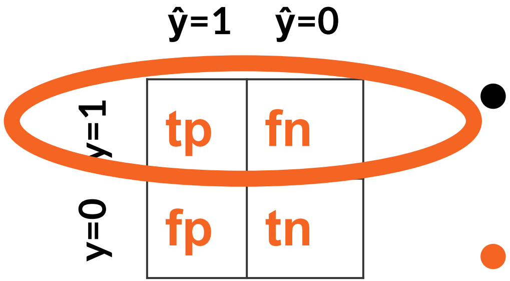
- Recall = tp / (tp + fn)
- p = tp+fn

How does recall change?

	ŷ=1	ŷ=0
y=1	tp	fn
y=0	fp	tn

- Recall = tp / (tp + fn)
- p = tp+fn
- Recall = (p - fn) / (p)

How does recall change?



	ŷ=1	ŷ=0
y=1	tp	fn
y=0	fp	tn

- Recall = tp / (tp + fn)
- p = tp+fn
- Recall = (p - fn) / (p)

How does recall change?

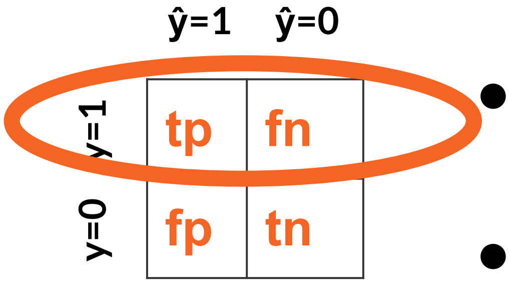
	ŷ=1	ŷ=0
y=1	tp	fn
y=0	fp	tn

- Recall = tp / (tp + fn)

- p = tp+fn
- Recall = (p - fn) / (p)

  = 1 - (fn/p)

How does recall change?



	ŷ=1	ŷ=0
y=1	tp	fn
y=0	fp	tn

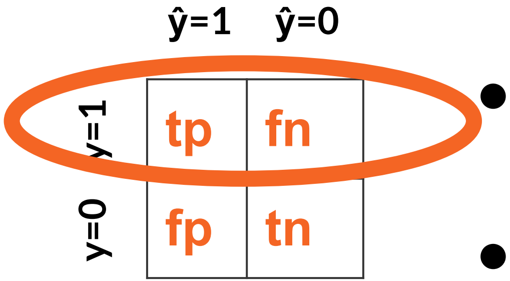
- Recall = tp / (tp + fn)

- p = tp+fn
- Recall = (p - fn) / (p)

  = 1 - (fn/p)

- If fn decreases, (fn/p) decreases since p is constant

How does recall change?



	$\hat{y}=1$	$\hat{y}=0$
$y=1$	tp	fn
$y=0$	fp	tn

- Recall = tp / (tp + fn)

- p = tp+fn
- Recall = (p - fn) / (p)

  = 1 - (fn/p)

- If fn decreases, (fn/p) decreases since p is constant
- ...so, recall [1-(fn/p)] always increases

Before we talk about precision, let's talk about the model itself

	Model predicts 1	Model predicts 0
True value is 1	**True positive** Correct prediction	**False negative**
True value is 0	**False positive**	**True negative** Correct prediction

What model do we use for binary output?

	Model predicts 1	Model predicts 0
True value is 1	**True positive** Correct prediction	**False negative**
True value is 0	**False positive**	**True negative** Correct prediction

Logistic transformation: $\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$

General logistic function where *p* denotes probability that y=1: $p(x) = \sigma(t) = \frac{1}{1+e^{-(\alpha+\beta\cdot x))}}$

Logistic regression

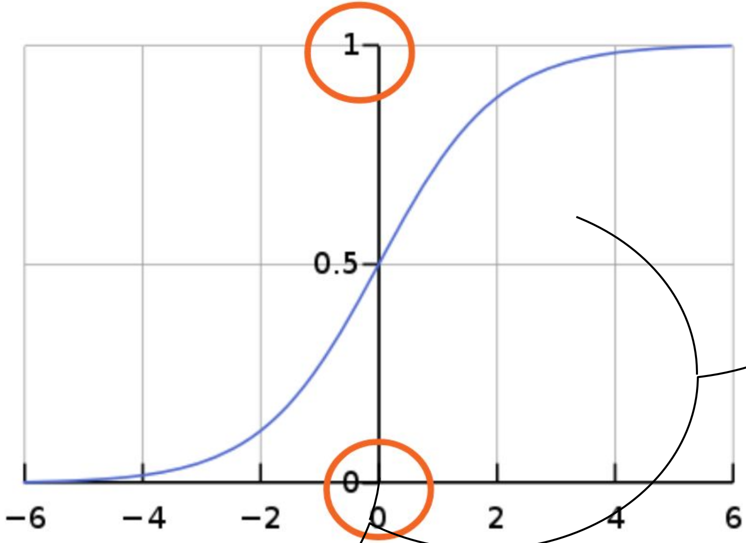
- When do we use it?
  - When output is ______

Logistic regression

- When do we use it?
  - When output is **binary**

- When you predict the ŷ for a given value of x, what do you get?
  - Is ŷ binary?
  - What’s going on under the hood?

Logistic (sigmoid) function



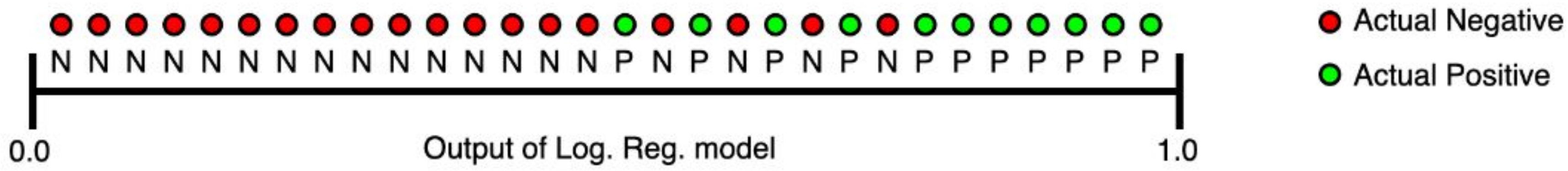
All outputs of a logistic function could be a probability (between 0 and 1!)

How do you get from probability to binary?

- Logistic regression interpretation (e.g. the “summarize” step) regards the odds of y=1
  - i.e., logit tells you about the **probability that y=1**

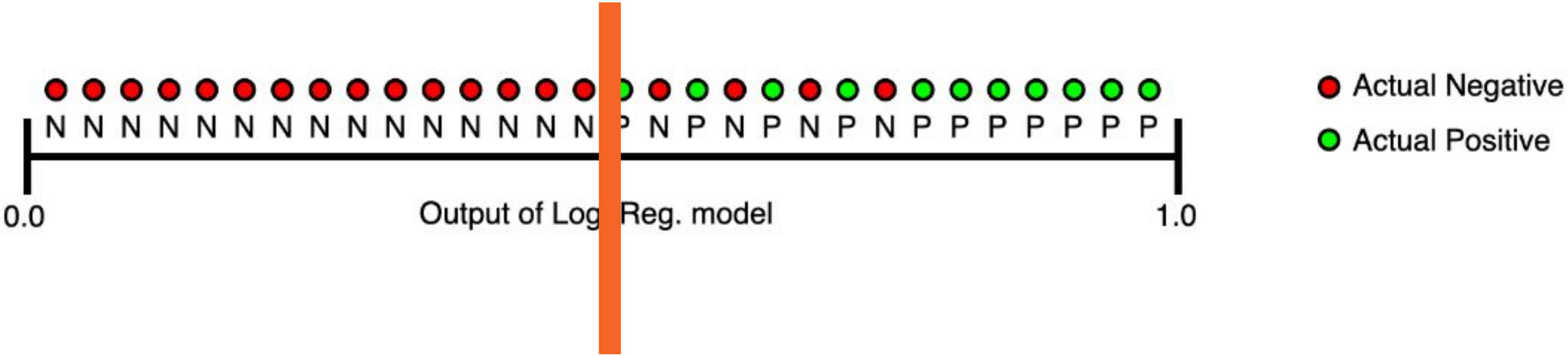
Sort the points by p (logistic output)

We have to make a decision.
How do we divide these?


N N N N N N N N N N N N N N N P N P N P N P N P P P P P P P
0.0
Output of Log. Reg. model
1.0
Actual Negative
Actual Positive

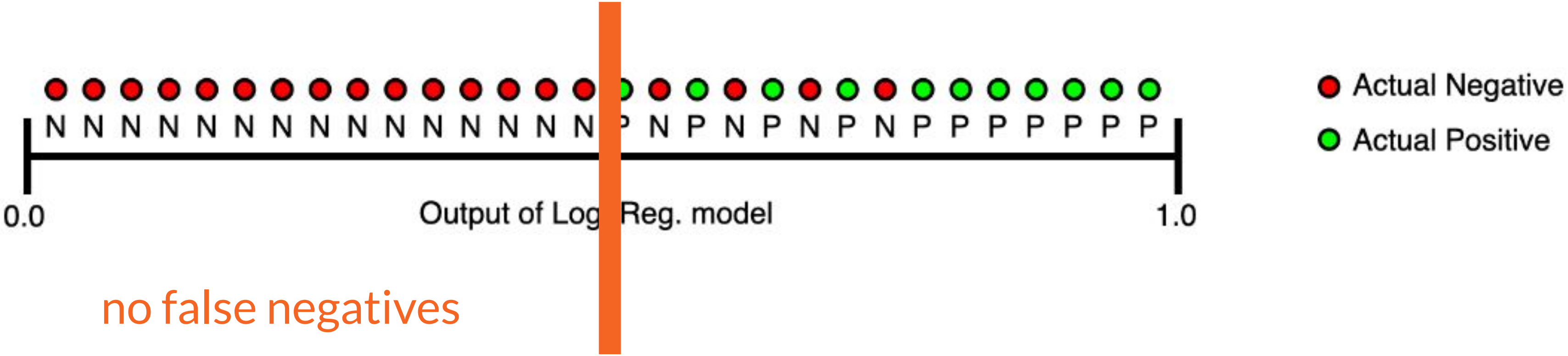
Sort the points by p (logistic output)

We have to make a decision.
How do we divide these?



Sort the points by p (logistic output)

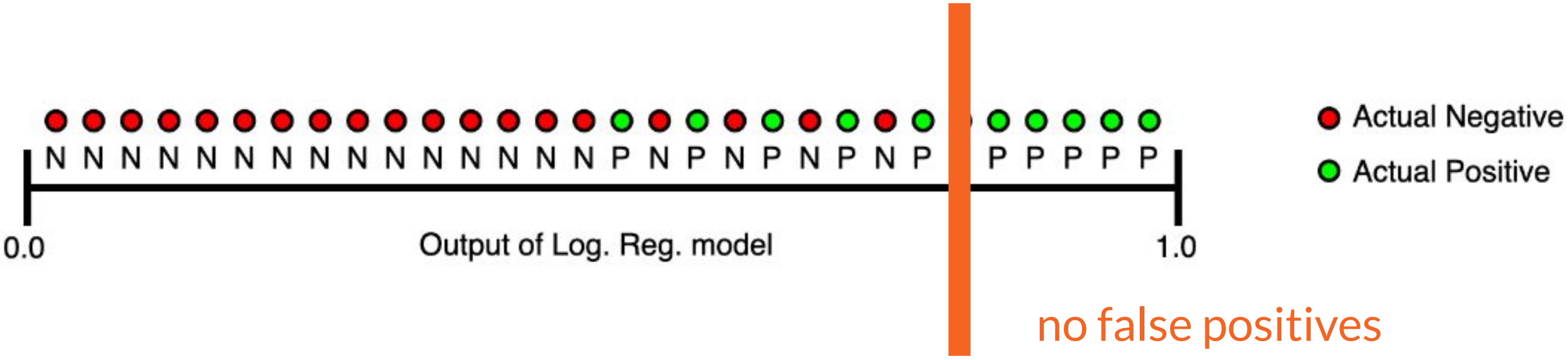
We have to make a decision.
How do we divide these?



no false negatives

Sort the points by p (logistic output)

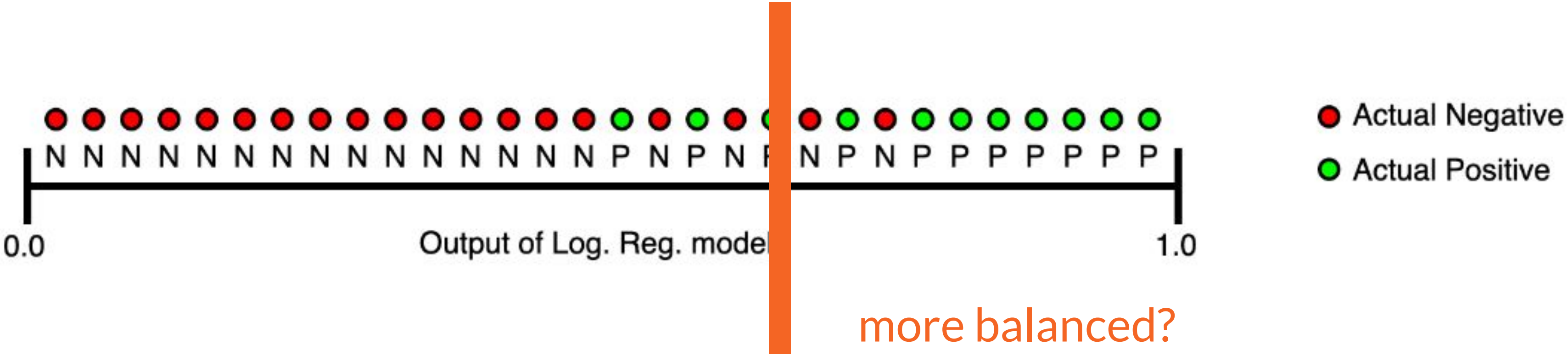
We have to make a decision.
How do we divide these?


N N N N N N N N N N N N N N N P N P N P N P N P P P P P P
0.0
Output of Log. Reg. model
1.0
● Actual Negative
● Actual Positive

no false positives

Sort the points by p (logistic output)

We have to make a decision.
How do we divide these?


N N N N N N N N N N N N N N N P N P N P N P N P P P P P P P
0.0
Output of Log. Reg. model
1.0
Actual Negative
Actual Positive

more balanced?

What do we do with p?

- You need to convert a **probability** (probability that y=1) to a **binary** output (y = either 1 or 0)

- **If I tell you the probability of y=1 is 100%, what would you say that y = ?**

What do we do with p?

- You need to convert a **probability** (probability that y=1) to a **binary** output (y = either 1 or 0)

- If I tell you the probability of y=1 is 100%, what would you say that y = ? (**1**)

What do we do with p?

- You need to convert a **probability** (probability that y=1) to a **binary** output (y = either 1 or 0)

- If I tell you the probability of y=1 is **90%**, what would you say that y = ?

What do we do with p?

- You need to convert a **probability** (probability that y=1) to a **binary** output (y = either 1 or 0)

- If I tell you the probability of y=1 is 90%, what would you say that y = ? (**1**)

What do we do with p?

- You need to convert a **probability** (probability that y=1) to a **binary** output (y = either 1 or 0)
- If I tell you the probability of y=1 is 90%, what would you say that y = ?
    - **What if the probability is 80, or 75, or 55, or 52, or 51, or 50.000001?**

Binarizing threshold

- Usually, you set a threshold of 50% to binarize a probability:
    - If you think there are better than 50/50 odds that y=1, then when choosing between outputting y=1 and y=0, you should choose y=1

Binarizing threshold

- What if we don’t choose 50% as a threshold?

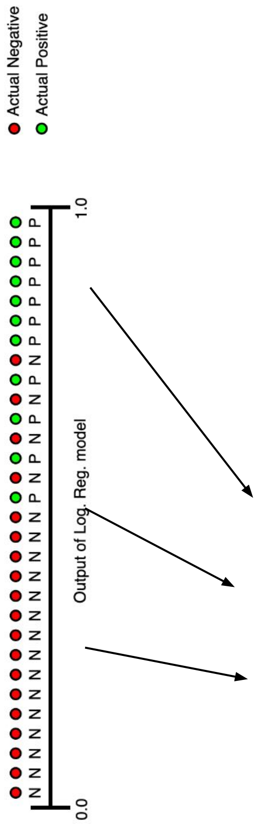


Image	ŷ Probability	50% Threshold	40% Threshold
Dog (1)	0.9	1	1
Dog (1)	0.4	0	1
Cat (0)	0.3	0	0

Changing thresholds

- Changing thresholds can let you change your precision and recall (possibly improving your metrics)

- We can use thresholds to show something about the relationship between precision and recall

Earlier: fn decrease → recall increase

	ŷ=1	ŷ=0
y=1	tp	fn
y=0	fp	tn

- Recall = tp / (tp + fn)
- p = tp+fn
- Recall = (p - fn) / (p)

  = 1 - (fn/p)

- If fn decreases, (fn/p) decreases since p is constant
- …so, recall [1-(fn/p)] always increases

	ŷ=1	ŷ=0
y=1	tp	fn
y=0	fp	tn

fn decrease → precision change?

- Precision = tp / (tp + fp)
- If we change thresholds and decrease the number of false negatives produced by our model, what happens to precision?

y (0 or 1)	ŷ (probability)	Threshold 1	Threshold 2
…	…	…	…

Decreasing fn can decrease precision = tp/(tp+fp)

Image	ŷ Probability	50% Threshold
Dog (1)	0.8	**1**
Dog (1)	0.2	0
Cat (0)	0.2	0
Cat (0)	0.2	0

https://datascience.stackexchange.com/questions/49117/inverse-relationship-between-precision-and-recall

Decreasing fn can decrease precision = tp/(tp+fp)

Image	ŷ Probability	50% Threshold (1 false negative)
Dog (1)	0.8	**1**
Dog (1)	0.2	0
Cat (0)	0.2	0
Cat (0)	0.2	0

https://datascience.stackexchange.com/questions/49117/inverse-relationship-between-precision-and-recall

Decreasing fn can decrease precision = tp/(tp+fp)

Precision = 1 / (1+0) = 1

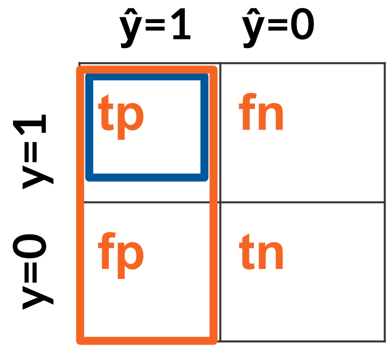
Image	ŷ Probability	50% Threshold (1 false negative)
Dog (1)	0.8	1
Dog (1)	0.2	0
Cat (0)	0.2	0
Cat (0)	0.2	0

	ŷ=1	ŷ=0
y=1	tp	fn
y=0	fp	tn

There are 0 false positives (cats predicted as dogs)

https://datascience.stackexchange.com/questions/49117/inverse-relationship-between-precision-and-recall

Decreasing fn can decrease precision = tp/(tp+fp)



	ŷ=1	ŷ=0
y=1	tp	fn
y=0	fp	tn

Precision = 1 / (1+0) = 1

Image	ŷ Probability	50% Threshold (1 false negative)	0% Threshold (0 false negatives)
Dog (1)	0.8	**1**	?
Dog (1)	0.2	0	?
Cat (0)	0.2	0	?
Cat (0)	0.2	0	?

Precision = ?

https://datascience.stackexchange.com/questions/49117/inverse-relationship-between-precision-and-recall

Decreasing fn can decrease precision = tp/(tp+fp)

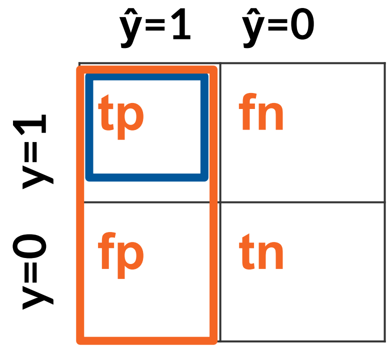
Precision = 1 / (1+0) = 1

Image	ŷ Probability	50% Threshold (1 false negative)	0% Threshold (0 false negatives)
Dog (1)	0.8	1	1
Dog (1)	0.2	0	1
Cat (0)	0.2	0	1
Cat (0)	0.2	0	1

Any probability > threshold = 1
All probabilities > 0 → Everything = 1!

https://datascience.stackexchange.com/questions/49117/inverse-relationship-between-precision-and-recall

Decreasing fn can decrease precision = tp/(tp+fp)



	ŷ=1	ŷ=0
y=1	tp	fn
y=0	fp	tn

Precision = 1 / (1+0) = 1

Image	ŷ Probability	50% Threshold (1 false negative)	0% Threshold (0 false negatives)
Dog (1)	0.8	1	1
Dog (1)	0.2	0	1
Cat (0)	0.2	0	1
Cat (0)	0.2	0	1

Precision = 2 / (2+2) = 0.5

https://datascience.stackexchange.com/questions/49117/inverse-relationship-between-precision-and-recall

Decreasing fn can decrease precision = tp/(tp+fp)

Precision = 1 / (1+0) = 1 → Precision = 2 / (2+2) = 0.5

Image	ŷ Probability	50% Threshold (1 false negative) →	0% Threshold (0 false negatives)
Dog (1)	0.8	**1**	**1**
Dog (1)	0.2	0	**1**
Cat (0)	0.2	0	**1**
Cat (0)	0.2	0	**1**

https://datascience.stackexchange.com/questions/49117/inverse-relationship-between-precision-and-recall

But decreasing fn can also increase precision = tp/(tp+fp)

Precision = 0 / (0+1) = 0

Image	ŷ Probability	50% Threshold (1 false negative)	0% Threshold (0 false negatives)
Cat (0)	1.0	**1**	?
Dog (1)	0.4	0	?
Cat (0)	0.1	0	?

Precision = ?

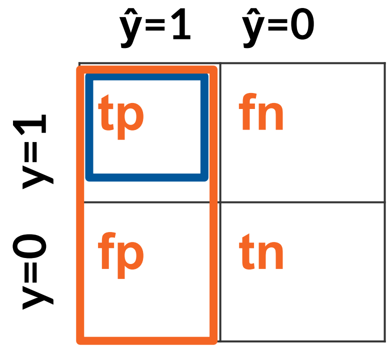
https://datascience.stackexchange.com/questions/49117/inverse-relationship-between-precision-and-recall

But decreasing fn can also increase precision = tp/(tp+fp)

Precision = 0 / (0+1) = 0

Image	ŷ Probability	50% Threshold (1 false negative)	0% Threshold (0 false negatives)
Cat (0)	1.0	**1**	**1**
Dog (1)	0.4	0	**1**
Cat (0)	0.1	0	**1**

Precision = 1 / (1+2) = 1/3

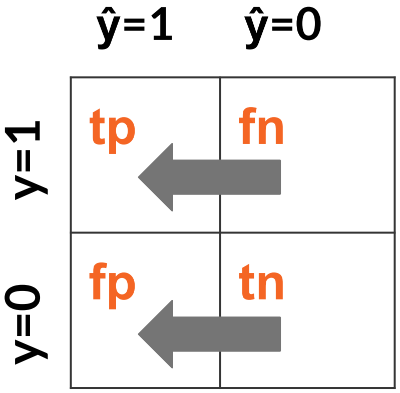


	ŷ=1	ŷ=0
y=1	tp	fn
y=0	fp	tn

https://datascience.stackexchange.com/questions/49117/inverse-relationship-between-precision-and-recall

Changing thresholds

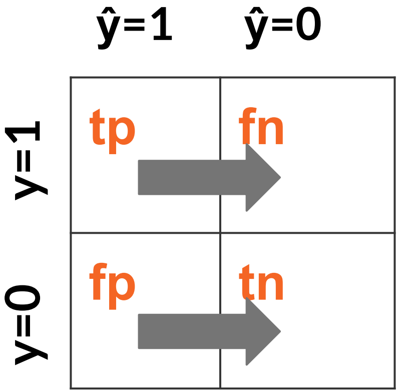
Decreasing the threshold (e.g. from 50% to 0%) can shift your false negatives and your true negatives towards positives



	ŷ=1	ŷ=0
y=1	tp	fn
y=0	fp	tn

Changing thresholds

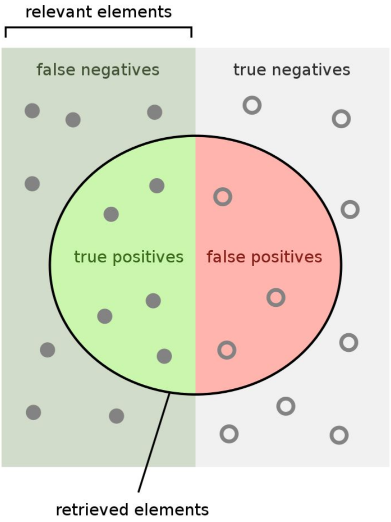
Increasing the threshold (e.g. from 0% to 50%) can shift your false positives and your true positives towards negatives



	ŷ=1	ŷ=0
y=1	tp	fn
y=0	fp	tn

Precision and Recall Recap

	ŷ=1	ŷ=0
y=1	tp	fn
y=0	fp	tn



- If we change the threshold to decrease the number of false negatives …
  - ○ Recall necessarily increases
  - ○ Precision could increase or decrease

Admin

- Phase 2 due this Thursday at 11:59 p.m.
  - No slip days are allowed for project deadlines
- Prelim regrade requests must be made in Gradescope by the end of October
- If project groups need to change after drop deadline, post on Ed to find a group / new members

INTERVIEW QUESTION!

A classification model separates email into two categories: "spam" or "not spam." If you raise the classification threshold, what will happen to precision? [Precision = tp / (tp+fp)]

1. Probably Decrease
2. Definitely Decrease
3. Probably Increase
4. Definitely Increase

https://developers.google.com/machine-learning/crash-course/classification/check-your-understanding-accuracy-precision-recall

INTERVIEW QUESTION!

A classification model separates email into two categories: "spam" or "not spam." If you raise the classification threshold, what will happen to precision?

1. Probably Decrease
2. Definitely Decrease
3. [✓ Probably Increase]
4. Definitely Increase

“In general, raising the classification threshold reduces false positives, thus raising precision.”

https://developers.google.com/machine-learning/crash-course/classification/check-your-understanding-accuracy-precision-recall

INTERVIEW QUESTION!

A classification model separates email into two categories: "spam" or "not spam." If you raise the classification threshold, what will happen to recall? [Recall = tp / (tp + fn)]

1. Always increase
2. Always stay constant
3. Always decrease or stay the same

https://developers.google.com/machine-learning/crash-course/classification/check-your-understanding-accuracy-precision-recall

INTERVIEW QUESTION!

A classification model separates email into two categories: "spam" or "not spam." If you raise the classification threshold, what will happen to recall?

1. Always increase
2. Always stay constant
3. [✓ Always decrease or stay the same]

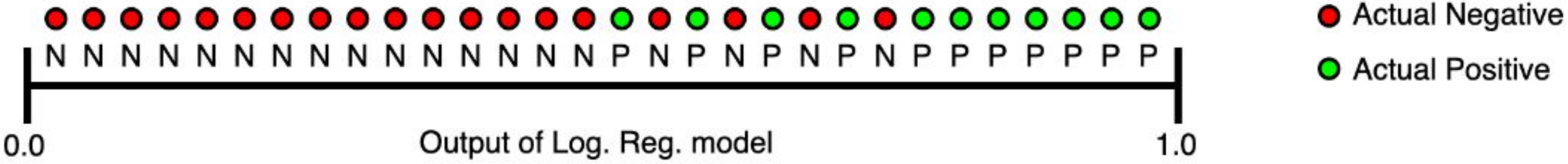
"Raising our classification threshold will cause the number of true positives to decrease or stay the same and will cause the number of false negatives to increase or stay the same. Thus, recall will either stay constant or decrease."

https://developers.google.com/machine-learning/crash-course/classification/check-your-understanding-accuracy-precision-recall

- What can we say about the relationship between recall and precision, both of which we want to be high?

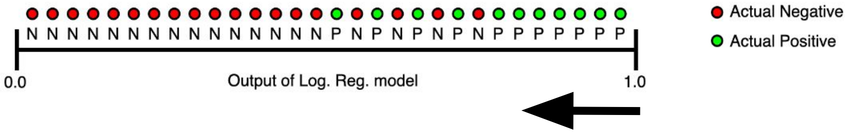
https://creatronix.de/classification-precision-and-recall/

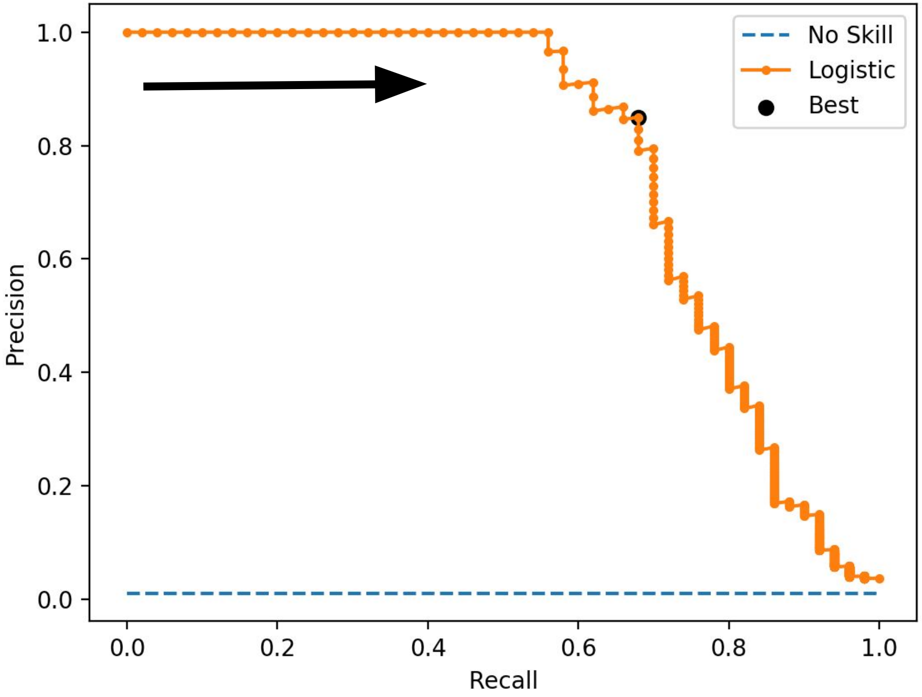
What if we consider a lot of thresholds all at once?


N N N N N N N N N N N N N N N P N P N P N P N P P P P P P P P
0.0
Output of Log. Reg. model
1.0
Actual Negative
Actual Positive

https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

Precision Recall Curves

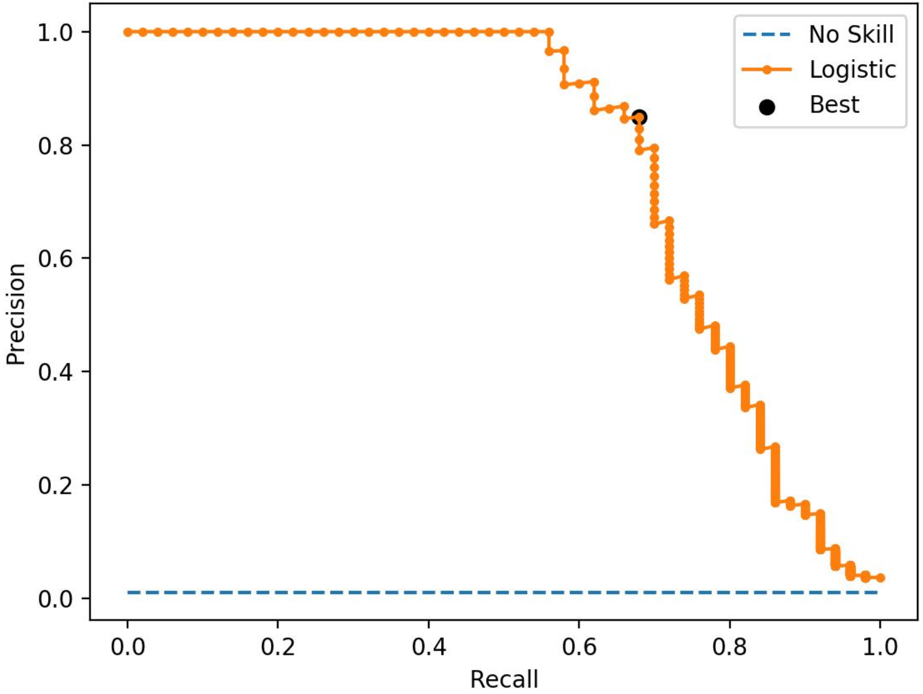

N N N N N N N N N N N N N N N P N P N P N P N P N P P P P P P
0.0
Output of Log. Reg. model
1.0
● Actual Negative
● Actual Positive



https://machinelearningmastery.com/threshold-moving-for-imbalanced-classification/

Precision Recall Curves

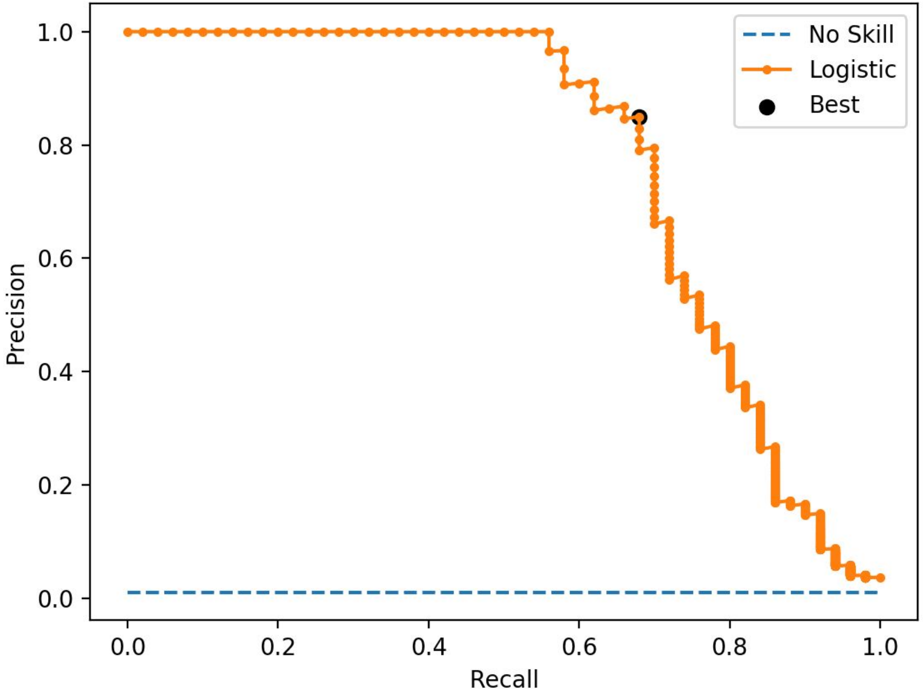
In general, for a model better than random, recall and precision have an inverse relationship



https://machinelearningmastery.com/threshold-moving-for-imbalanced-classification/

Precision Recall Curves

You can plot PR curves to help you think about what threshold is “Best” (giving you ideal achievable recall and precision)



https://machinelearningmastery.com/threshold-moving-for-imbalanced-classification/

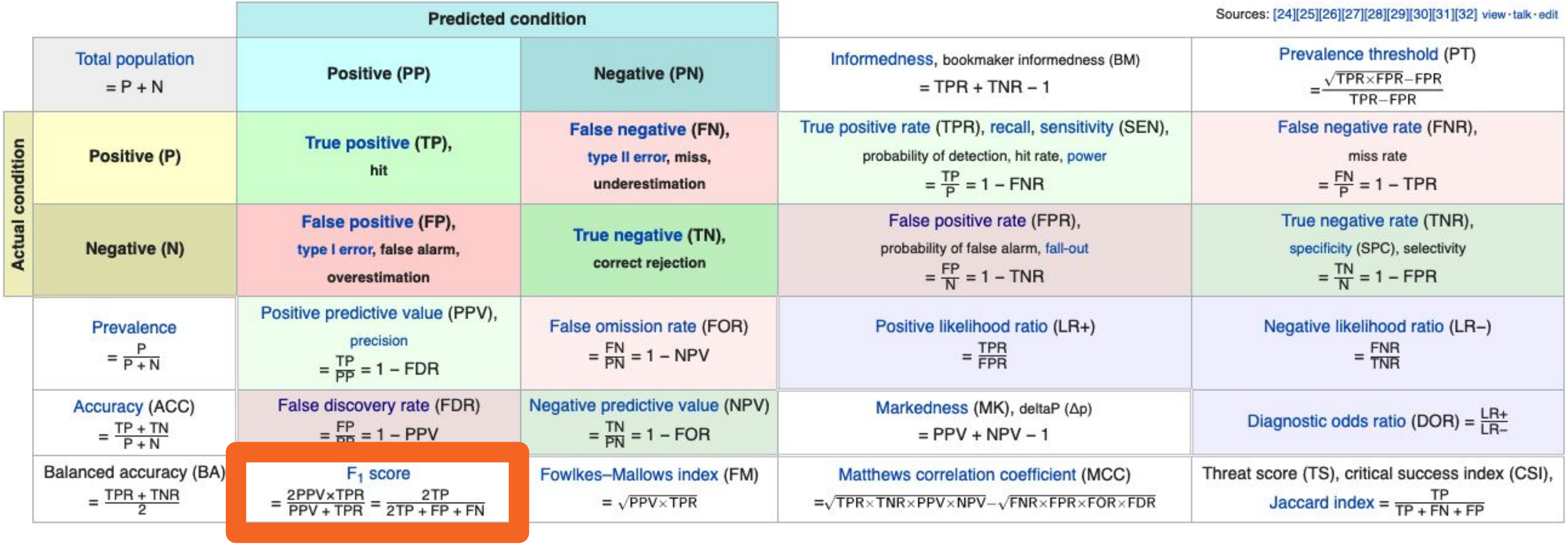
F1 combines precision and recall

- F_1 score: $\frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} = 2\frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{2\mathrm{tp}}{2\mathrm{tp} + \mathrm{fp} + \mathrm{fn}}$

- This is the *harmonic mean* of precision and recall

- Higher F_1 is better

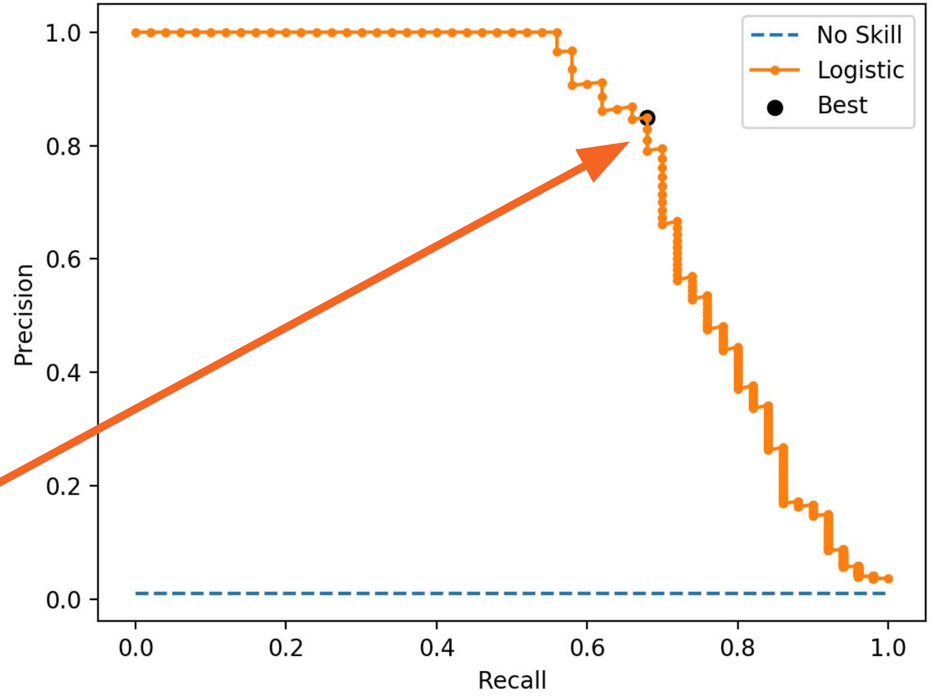
(Lots of options for classification)



		Predicted condition			Sources: [24][25][26][27][28][29][30][31][32] view · talk · edit
	Total population $= P + N$	Positive (PP)	Negative (PN)	Informedness, bookmaker informedness (BM) $= TPR + TNR - 1$	Prevalence threshold (PT) $= \frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$
Actual condition	Positive (P)	True positive (TP), hit	False negative (FN), type II error, miss, underestimation	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power $= \frac{TP}{P} = 1 - FNR$	False negative rate (FNR), miss rate $= \frac{FN}{P} = 1 - TPR$
	Negative (N)	False positive (FP), type I error, false alarm, overestimation	True negative (TN), correct rejection	False positive rate (FPR), probability of false alarm, fall-out $= \frac{FP}{N} = 1 - TNR$	True negative rate (TNR), specificity (SPC), selectivity $= \frac{TN}{N} = 1 - FPR$
	Prevalence $= \frac{P}{P + N}$	Positive predictive value (PPV), precision $= \frac{TP}{PP} = 1 - FDR$	False omission rate (FOR) $= \frac{FN}{PN} = 1 - NPV$	Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$	Negative likelihood ratio (LR−) $= \frac{FNR}{TNR}$
	Accuracy (ACC) $= \frac{TP + TN}{P + N}$	False discovery rate (FDR) $= \frac{FP}{PP} = 1 - PPV$	Negative predictive value (NPV) $= \frac{TN}{PN} = 1 - FOR$	Markedness (MK), deltaP (Δp) $= PPV + NPV - 1$	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$
	Balanced accuracy (BA) $= \frac{TPR + TNR}{2}$	F_1 score $= \frac{2PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$	Fowlkes–Mallows index (FM) $= \sqrt{PPV \times TPR}$	Matthews correlation coefficient (MCC) $= \sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$	Threat score (TS), critical success index (CSI), Jaccard index $= \frac{TP}{TP + FN + FP}$

https://en.wikipedia.org/wiki/Sensitivity_and_specificity

F1 score



This is the point that maximizes the F_1 score (harmonic mean of precision and recall)

https://machinelearningmastery.com/threshold-moving-for-imbalanced-classification/

Binary evaluation metrics, part 2

- F_1 score: $\frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} = 2\frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{2\mathrm{tp}}{2\mathrm{tp} + \mathrm{fp} + \mathrm{fn}}$

- AUC-ROC curve: fp / (tn + fp) a.k.a. specificity versus recall

- Higher F_1 and AUC is better

https://res.cloudinary.com/engineering-com/image/fetch/w_450,f_jpg,c_fill/http://image.engineering.com/239575/articles/jan2015/ti12/ti12-1.PNG

Another curve: ROC

- **Receiver Operating Characteristic** (ROC) plots the FPR vs. TPR
  - Originally developed for military radar purposes

https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

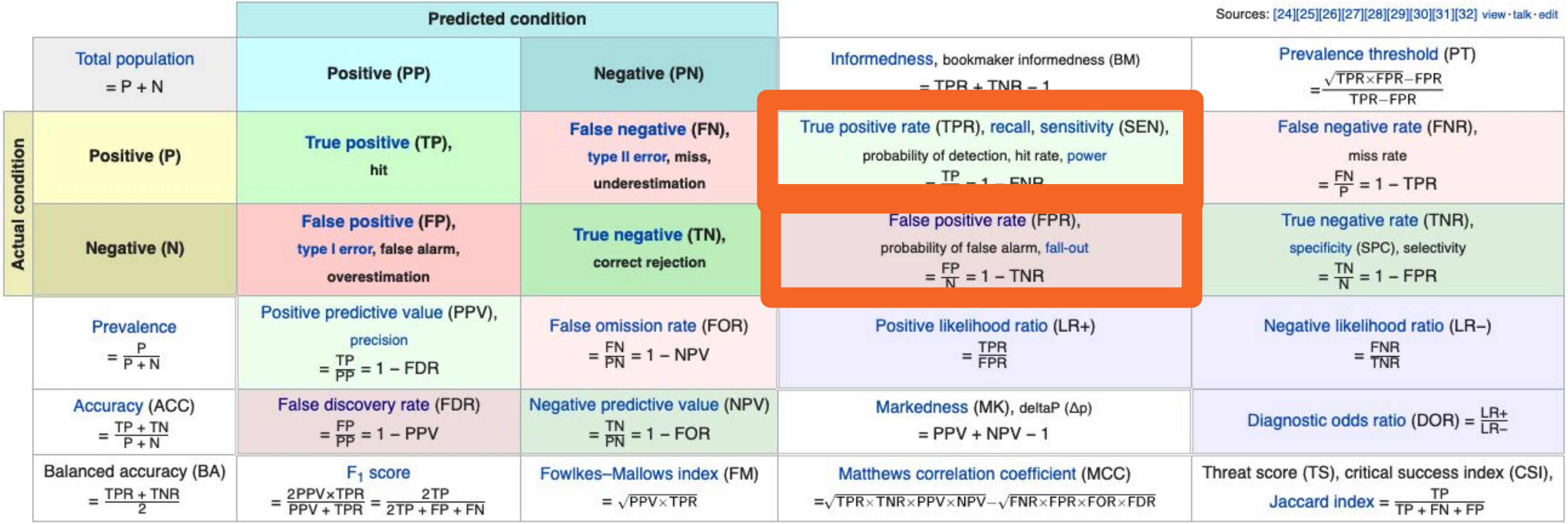
https://res.cloudinary.com/engineering-com/image/fetch/w_450,f_jpg,c_fill/http://image.engineering.com/239575/articles/jan2015/ti12/ti12-1.PNG

What is ROC?

- **Receiver Operating Characteristic** (ROC) plots the FPR vs. TPR
  - Originally developed for military radar purposes
- **False Positive Rate** (a.k.a. 1-specificity)
  - fp / (fp + tn)
- **True Positive Rate** (a.k.a. recall, sensitivity)
  - tp / (tp + fn)

https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

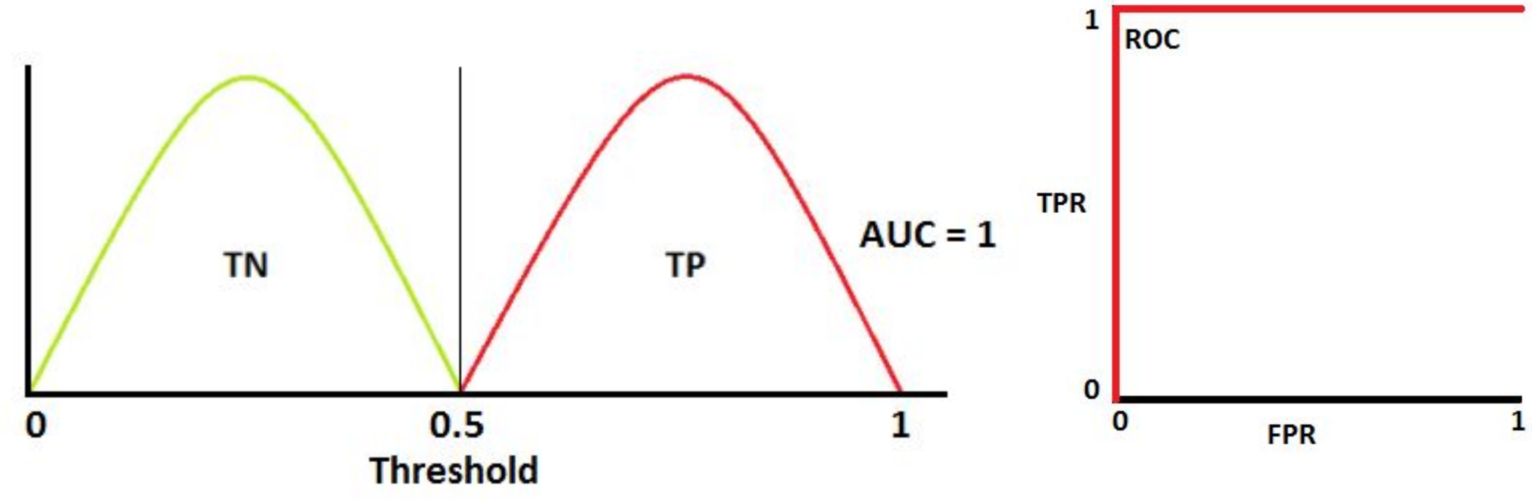
(Lots of options for classification)



		Predicted condition			Sources: [24][25][26][27][28][29][30][31][32] view · talk · edit
	Total population = P + N	Positive (PP)	Negative (PN)	Informedness, bookmaker informedness (BM) = TPR + TNR − 1	Prevalence threshold (PT) $= \frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$
Actual condition	Positive (P)	**True positive (TP),** **hit**	**False negative (FN),** **type II error, miss,** **underestimation**	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power $= \frac{TP}{P} = 1 - FNR$	False negative rate (FNR), miss rate $= \frac{FN}{P} = 1 - TPR$
	Negative (N)	**False positive (FP),** **type I error, false alarm,** **overestimation**	**True negative (TN),** **correct rejection**	False positive rate (FPR), probability of false alarm, fall-out $= \frac{FP}{N} = 1 - TNR$	True negative rate (TNR), specificity (SPC), selectivity $= \frac{TN}{N} = 1 - FPR$
	Prevalence $= \frac{P}{P + N}$	Positive predictive value (PPV), precision $= \frac{TP}{PP} = 1 - FDR$	False omission rate (FOR) $= \frac{FN}{PN} = 1 - NPV$	Positive likelihood ratio (LR+) $= \frac{TPR}{FPR}$	Negative likelihood ratio (LR−) $= \frac{FNR}{TNR}$
	Accuracy (ACC) $= \frac{TP + TN}{P + N}$	False discovery rate (FDR) $= \frac{FP}{PP} = 1 - PPV$	Negative predictive value (NPV) $= \frac{TN}{PN} = 1 - FOR$	Markedness (MK), deltaP (Δp) = PPV + NPV − 1	Diagnostic odds ratio (DOR) $= \frac{LR+}{LR-}$
	Balanced accuracy (BA) $= \frac{TPR + TNR}{2}$	F_1 score $= \frac{2PPV \times TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$	Fowlkes–Mallows index (FM) $= \sqrt{PPV \times TPR}$	Matthews correlation coefficient (MCC) $= \sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$	Threat score (TS), critical success index (CSI), Jaccard index $= \frac{TP}{TP + FN + FP}$

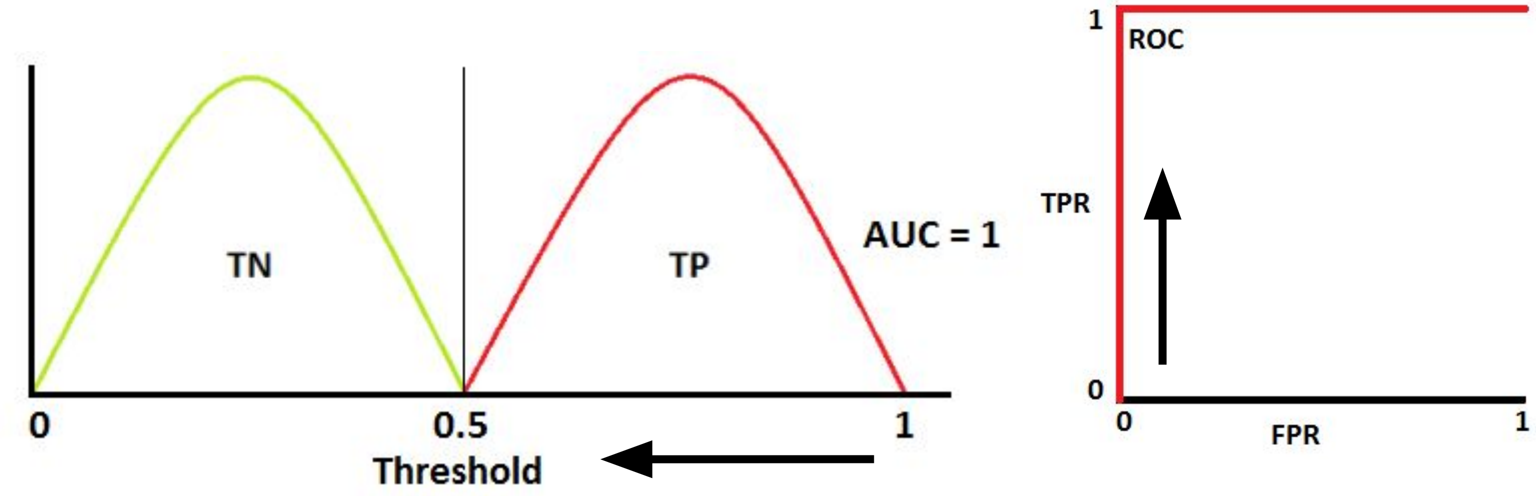
https://en.wikipedia.org/wiki/Sensitivity_and_specificity

Perfectly separated data → perfect ROC curve



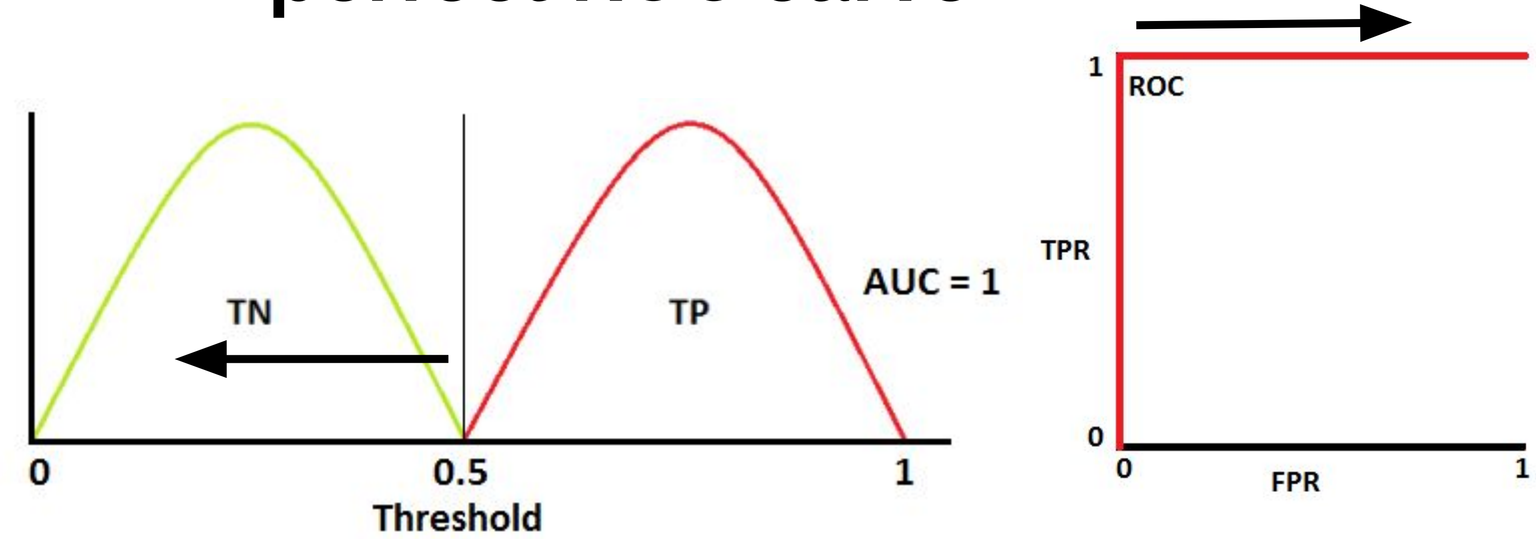
https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

Perfectly separated data → perfect ROC curve



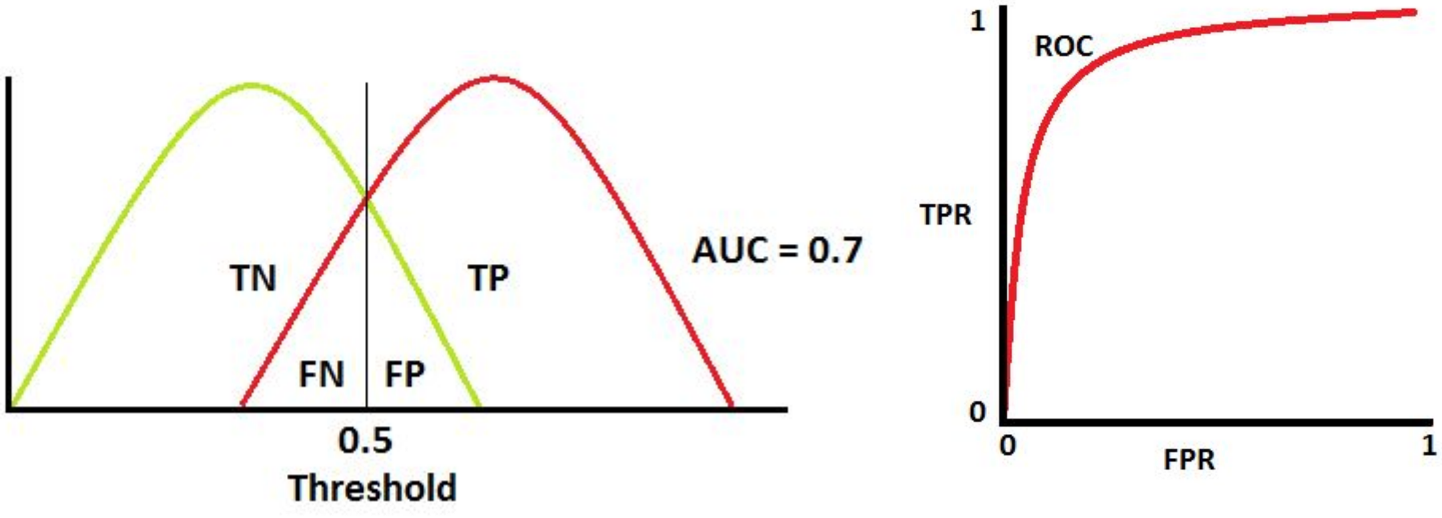
https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

Perfectly separated data → perfect ROC curve



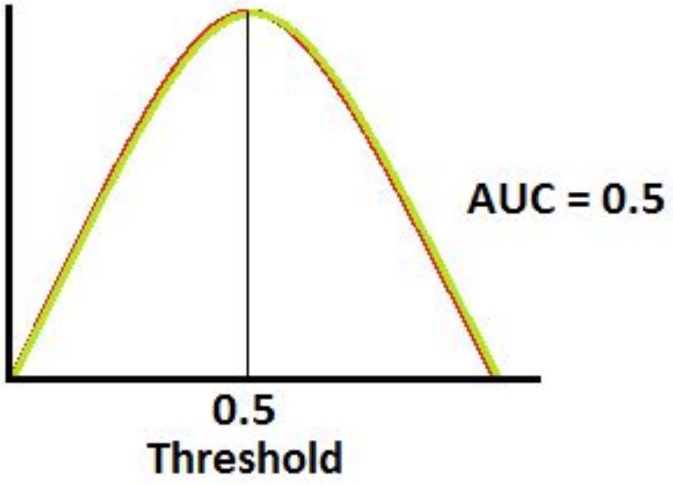
https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

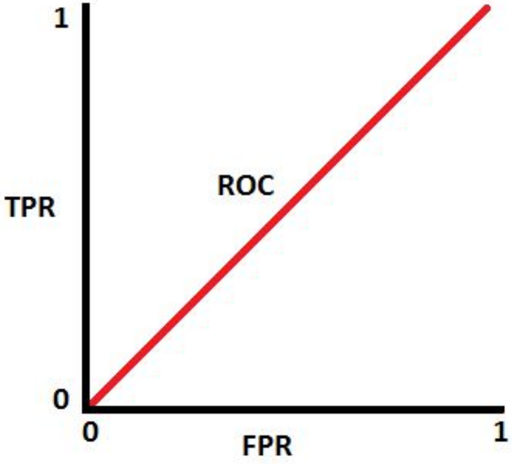
Slightly overlapping data → slightly lower ROC curve



https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

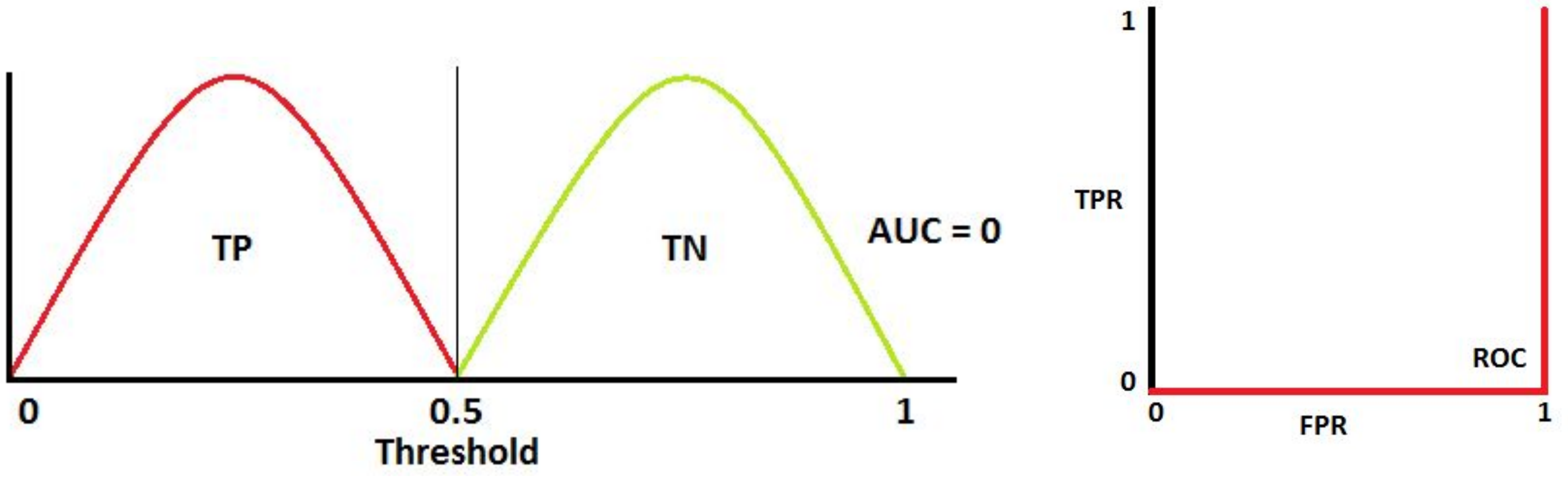
Exactly overlapping data → diagonal ROC curve (as good as random)





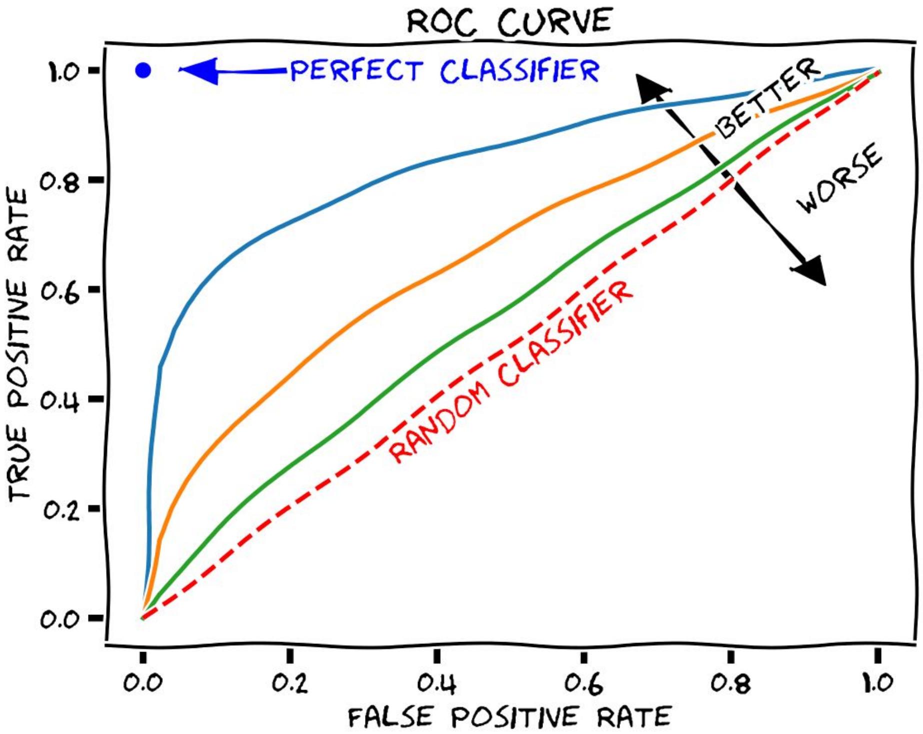
https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

Perfectly flipped data → worst ROC curve



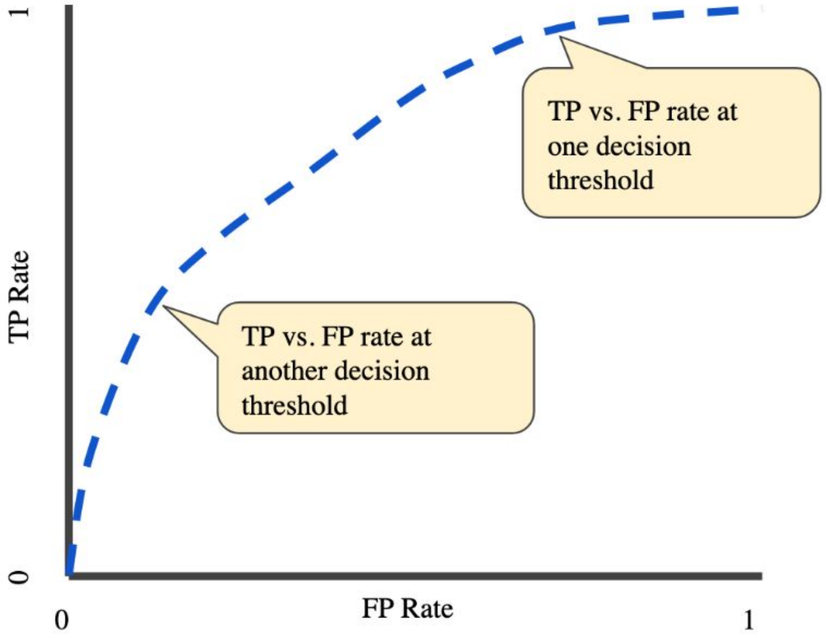
https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

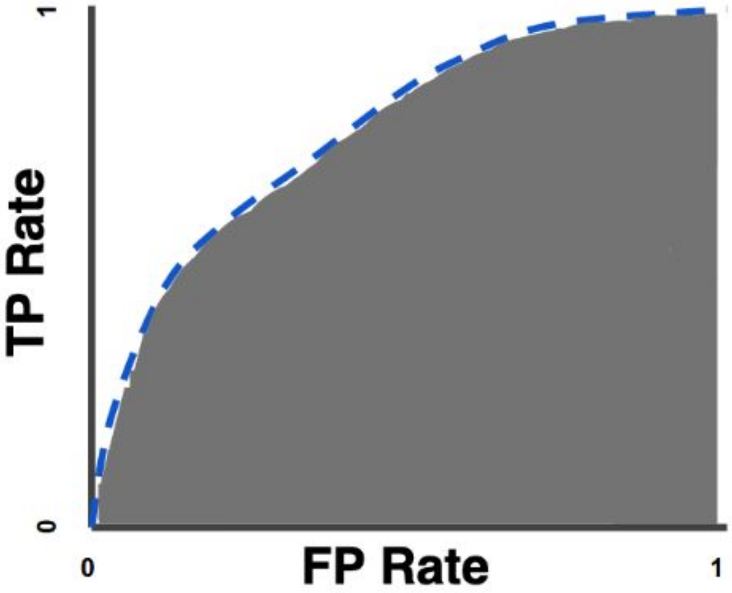
Good vs. Bad ROC



https://commons.wikimedia.org/wiki/File:Roc-draft-xkcd-style.svg

ROC curve → AUC



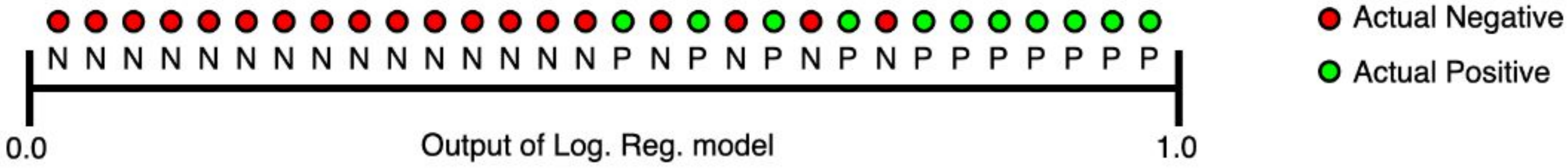


AUC (Area under the ROC Curve).

https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

AUC interpretation

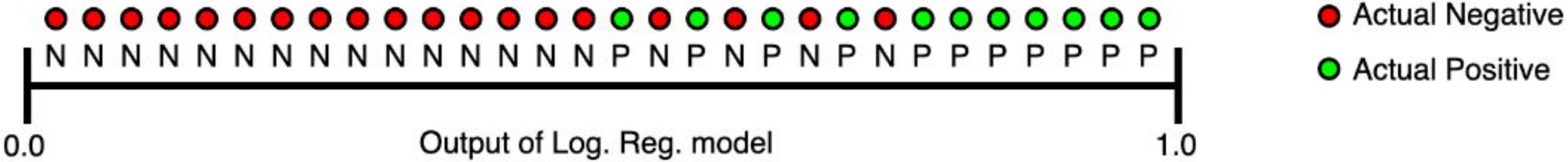
- Ability of your classifier to distinguish between two classes
- Summarizes the ROC curve


N N N N N N N N N N N N N N N P N P N P N P N P P P P P P P
0.0
Output of Log. Reg. model
1.0
● Actual Negative
● Actual Positive

https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

AUC interpretation

Probability that your logit model ranks a random positive example more highly than a random negative example


N N N N N N N N N N N N N N N P N P N P N P N P P P P P P P
0.0
Output of Log. Reg. model
1.0
Actual Negative
Actual Positive

https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

Classification prediction metrics in Python

- `from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score`

- `precision_score(y_true,y_hat)`
- `recall_score(y_true,y_hat)`
- `f1_score(y_true,y_hat)`
- `roc_auc_score(y_true,y_hat)`

Nomenclature for these slides

When providing formulas in these slides, we'll generalize to these

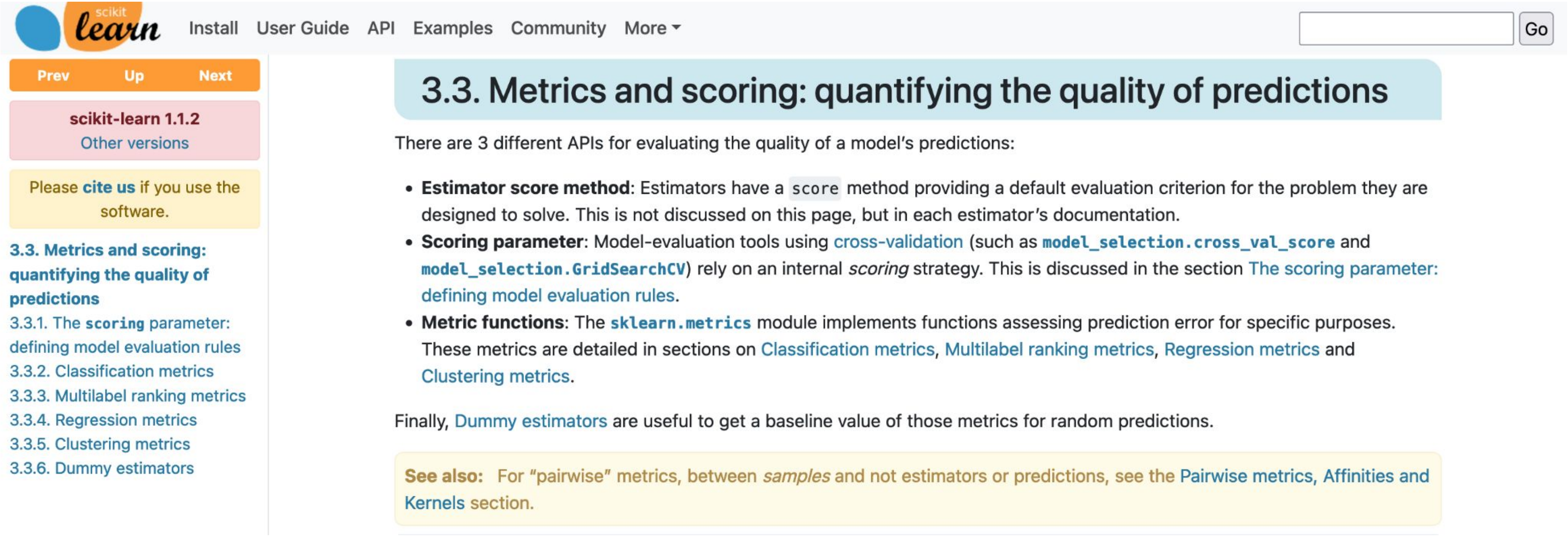
General case	“y_true”, “y_i”	“y_hat”, “$\hat{y}_i$”
Train set	y_train	y_hat_train
Test set	y_test	y_hat_test

Classification prediction metrics in Python

- `from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score`

- `precision_score(y_true,y_hat)`
- `recall_score(y_true,y_hat)`
- `f1_score(y_true,y_hat)`
- `roc_auc_score(y_true,y_hat)`

Check the Scikit Learn documentation!



scikit-learn 1.1.2
Other versions

Please **cite us** if you use the software.

3.3. Metrics and scoring: quantifying the quality of predictions
3.3.1. The `scoring` parameter: defining model evaluation rules
3.3.2. Classification metrics
3.3.3. Multilabel ranking metrics
3.3.4. Regression metrics
3.3.5. Clustering metrics
3.3.6. Dummy estimators

3.3. Metrics and scoring: quantifying the quality of predictions

There are 3 different APIs for evaluating the quality of a model's predictions:

- **Estimator score method**: Estimators have a `score` method providing a default evaluation criterion for the problem they are designed to solve. This is not discussed on this page, but in each estimator's documentation.
- **Scoring parameter**: Model-evaluation tools using cross-validation (such as `model_selection.cross_val_score` and `model_selection.GridSearchCV`) rely on an internal *scoring* strategy. This is discussed in the section The scoring parameter: defining model evaluation rules.
- **Metric functions**: The `sklearn.metrics` module implements functions assessing prediction error for specific purposes. These metrics are detailed in sections on Classification metrics, Multilabel ranking metrics, Regression metrics and Clustering metrics.

Finally, Dummy estimators are useful to get a baseline value of those metrics for random predictions.

See also: For "pairwise" metrics, between *samples* and not estimators or predictions, see the Pairwise metrics, Affinities and Kernels section.

https://scikit-learn.org/stable/modules/model_evaluation.html

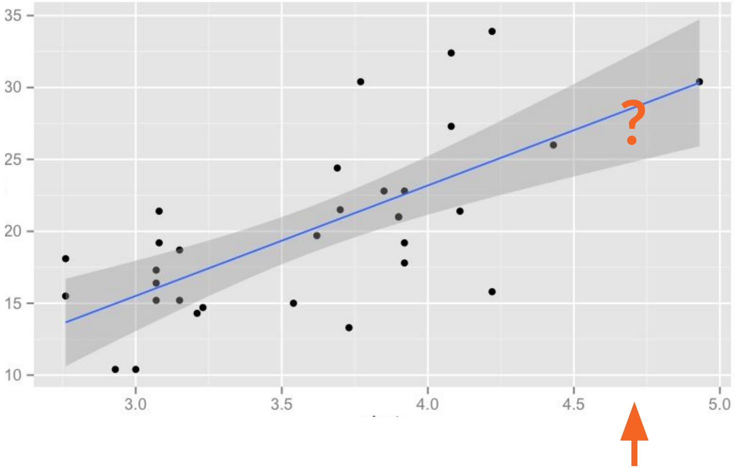
1 minute break & attendance

tinyurl.com/ykv3f9a9

THE BEST WAY TO
EXPLAIN OVERFITTING
imgflip.com
ADDTEXT.COM

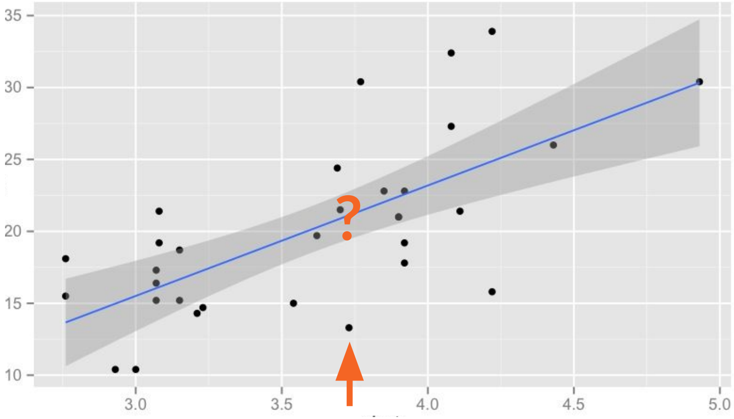
https://pbs.twimg.com/media/DdkUUTMV4AAsohU.jpg

Motivation

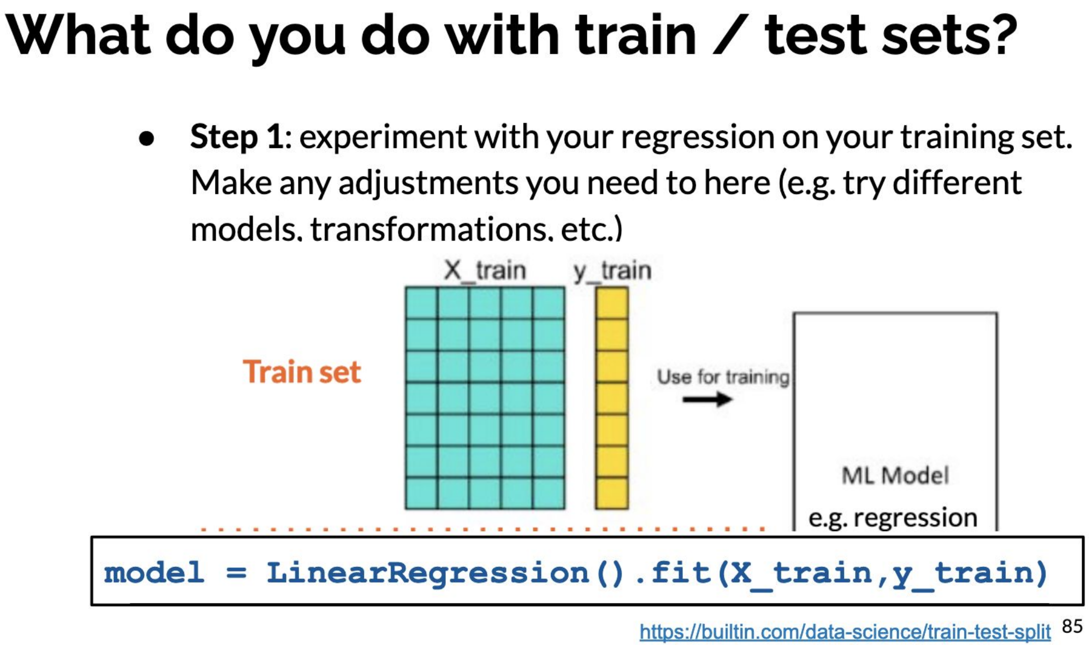


- How confident can we be that the patterns in the things we *have* seen will apply to the things we *haven't* seen?

Motivation



- How confident can we be that the patterns in the things we *have* seen will apply to the things we *haven't* seen?
- **Key insight: pretend we *haven't* seen some of the things we *have* seen, and check if we would have guessed correctly**



What do you do with train / test sets?

- **Step 1**: experiment with your regression on your training set. Make any adjustments you need to here (e.g. try different models, transformations, etc.)

```
model = LinearRegression().fit(X_train,y_train)
```

https://builtin.com/data-science/train-test-split 85

What could go wrong with train/test?

- Train set proportion too big (test set not broad enough)

What could go wrong with train/test?

- Train set proportion too big (test set not broad enough)
- Test set proportion too big (not enough training data)

What could go wrong with train/test?

- Train set proportion too big (test set not broad enough)
- Test set proportion too big (not enough training data)
- Test set data too similar/dissimilar to train set data (overestimating/underestimating generalization)

What could go wrong with train/test?

- Train set proportion too big (test set not broad enough)
- Test set proportion too big (not enough training data)
- Test set data too similar/dissimilar to train set data (overestimating/underestimating generalization)

If your train set is only side sleepers and your test set is only back sleepers, no matter how you change your model, you still won't get good evaluations and will overfit to side sleepers!

What could go wrong with train/test?

- Train set proportion too big (test set not broad enough)
- Test set proportion too big (not enough training data)
- Test set data too similar/dissimilar to train set data (overestimating/underestimating generalization)

If your original set is only US undergrads, no amount of splitting will test generalization to middle-aged Brazilian manufacturing workers

What could go wrong with train/test?

- Train set proportion too big (test set not broad enough)
- Test set proportion too big (not enough training data)
- Test set data too similar/dissimilar to train set data (overestimating/underestimating generalization)
- Having ordered rows in a df (shouldn’t randomize x-axis order of time series)

What could go wrong with train/test?

- Train set proportion too big (test set not broad enough)
- Test set proportion too big (not enough training data)
- Test set data too similar/dissimilar to train set data (overestimating/underestimating generalization)
- Having ordered rows in a df (shouldn't randomize x-axis order of time series)

If you split your time series data randomly into train and test data, you'll have time series with a bunch of random missing days. Missing chronological data is bad!

What could go wrong with train/test?

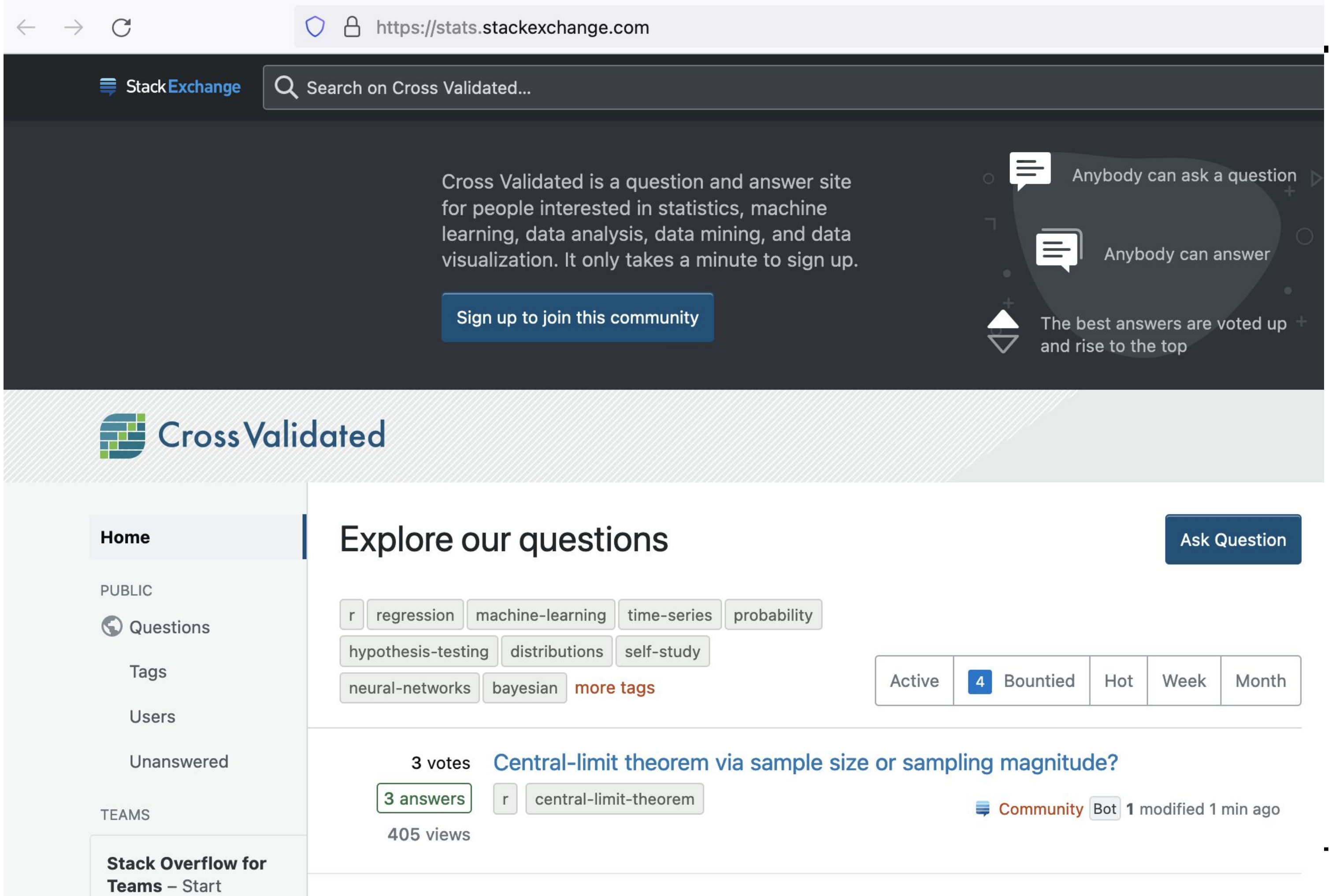
Solution:
Cross-Validation

- Train set proportion too big (test set not broad enough)
- Test set proportion too big (not enough training data)
- Test set data too similar/dissimilar to train set data (overestimating/underestimating generalization)
- Having ordered rows in a df (shouldn’t randomize x-axis order of time series)

What could go wrong with train/test?

- Train set proportion too big (test set not broad enough)
- Test set proportion too big (not enough training data)
- Test set data too similar/dissimilar to train set data (overestimating/underestimating generalization)
- Having ordered rows in a df (shouldn’t randomize x-axis order of time series)

Solution:
Sliding Windows



https://stats.stackexchange.com

Cross Validated is a question and answer site for people interested in statistics, machine learning, data analysis, data mining, and data visualization. It only takes a minute to sign up.

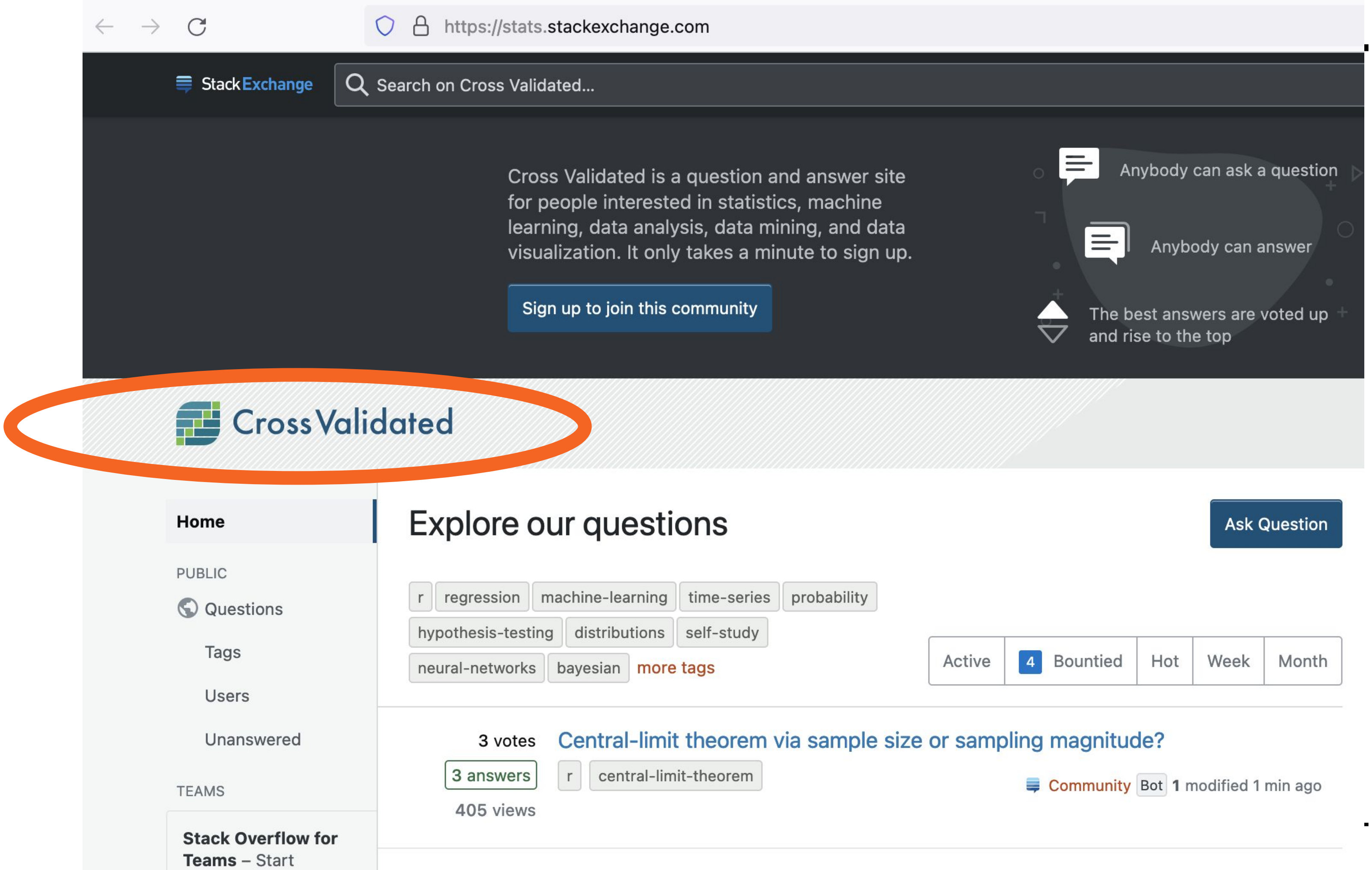
Anybody can ask a question

Anybody can answer

The best answers are voted up and rise to the top

Explore our questions

Central-limit theorem via sample size or sampling magnitude?



Cross Validated is a question and answer site for people interested in statistics, machine learning, data analysis, data mining, and data visualization. It only takes a minute to sign up.

Sign up to join this community
Anybody can ask a question
Anybody can answer
The best answers are voted up and rise to the top
Cross Validated
Home
PUBLIC
Questions
Tags
Users
Unanswered
TEAMS
Stack Overflow for Teams – Start

Explore our questions

3 votes
3 answers
405 views

Central-limit theorem via sample size or sampling magnitude?

r central-limit-theorem

Community Bot 1 modified 1 min ago

Validation Sets

- We know we can split data into train / test sets
- But there are a lot of reasons this may not be reliable
- What if we do the train / test splitting process ***again*** with the training set?

We have lots of choices of prediction models

- Which variables should we include as inputs?
- Should we use transformations?
- Should we filter training data?
- Should we use more complicated models? Interactions?

We have lots of choices of prediction models

- Which variables should we include as inputs?
- Should we use transformations?
- Should we filter training data?
- Should we use more complicated models? Interactions?

Danger: The more models we try, the more chances to get good results by pure luck

The more tries you take, the more likely you’ll get good eval metrics

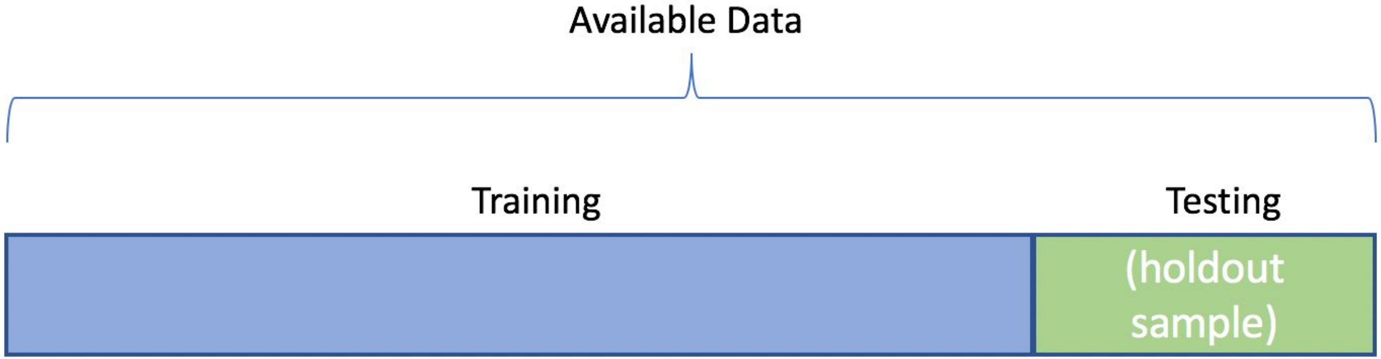
https://en.wikipedia.org/wiki/Darts

Validation Sets

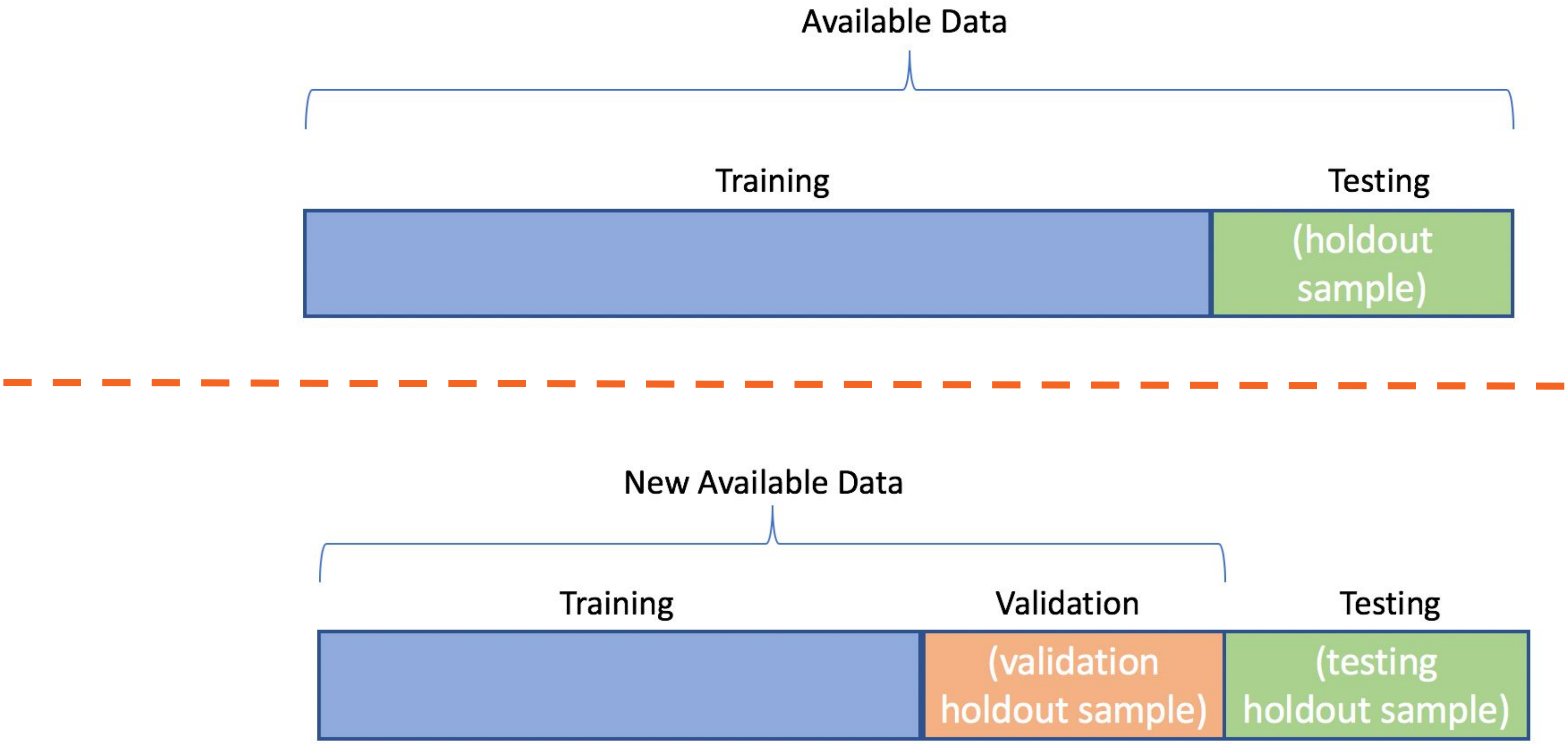
- Don’t want to cheat by overusing your test set

- You can only use your test set so many times before it’s no longer useful to discover overfitting (since you’ll start fitting to your test set too)

- Use validation set to check your evaluation metrics before checking your “final” test set



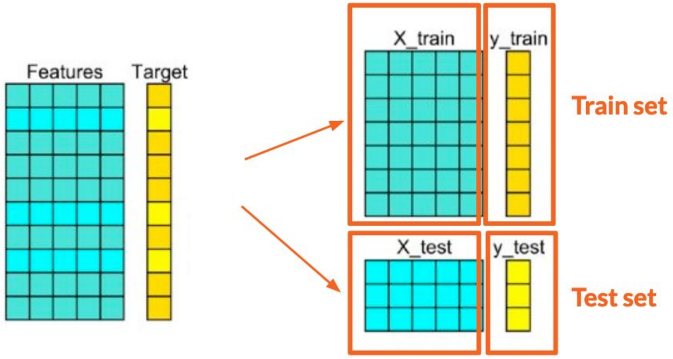
https://algotrading101.com/learn/train-test-split/



https://algotrading101.com/learn/train-test-split/

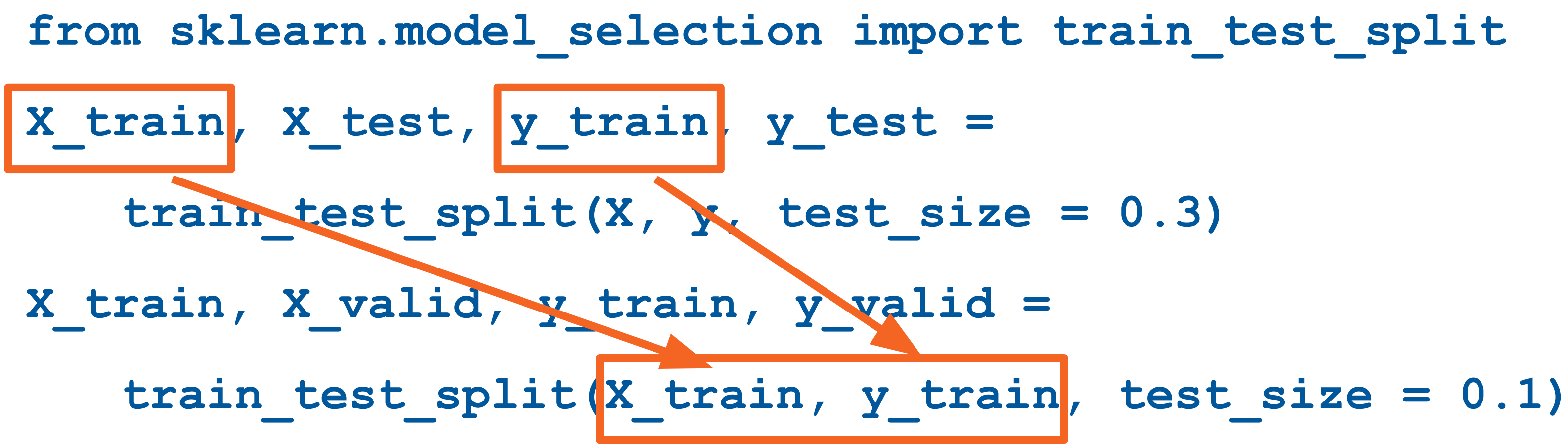
Sub-split training to get validation

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size = 0.3)
```



https://builtin.com/data-science/train-test-split

Sub-split training to get validation



```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size = 0.3)
X_train, X_valid, y_train, y_valid =
    train_test_split(X_train, y_train, test_size = 0.1)
```

Sub-split training to get validation

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size = 0.3)
X_train, X_valid, y_train, y_valid =
    train_test_split(X_train, y_train, test_size = 0.1)
```

Sub-split training to get validation

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size = 0.3)
X_train, X_valid, y_train, y_valid =
    train_test_split(X_train, y_train, test_size = 0.1)
```

70/30 train/test split in first step

?

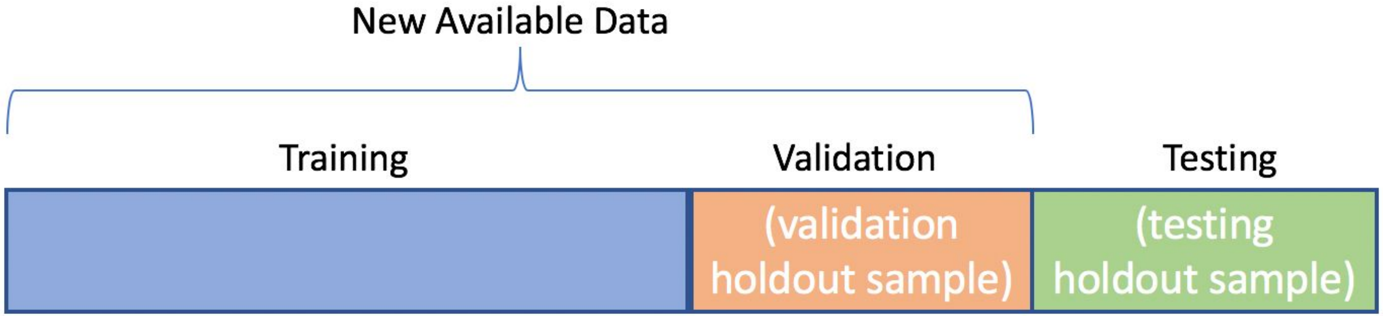
What % of the overall dataset are your train / valid / test sets?

Sub-split training to get validation

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size = 0.3)
X_train, X_valid, y_train, y_valid =
    train_test_split(X_train, y_train, test_size = 0.1)
```

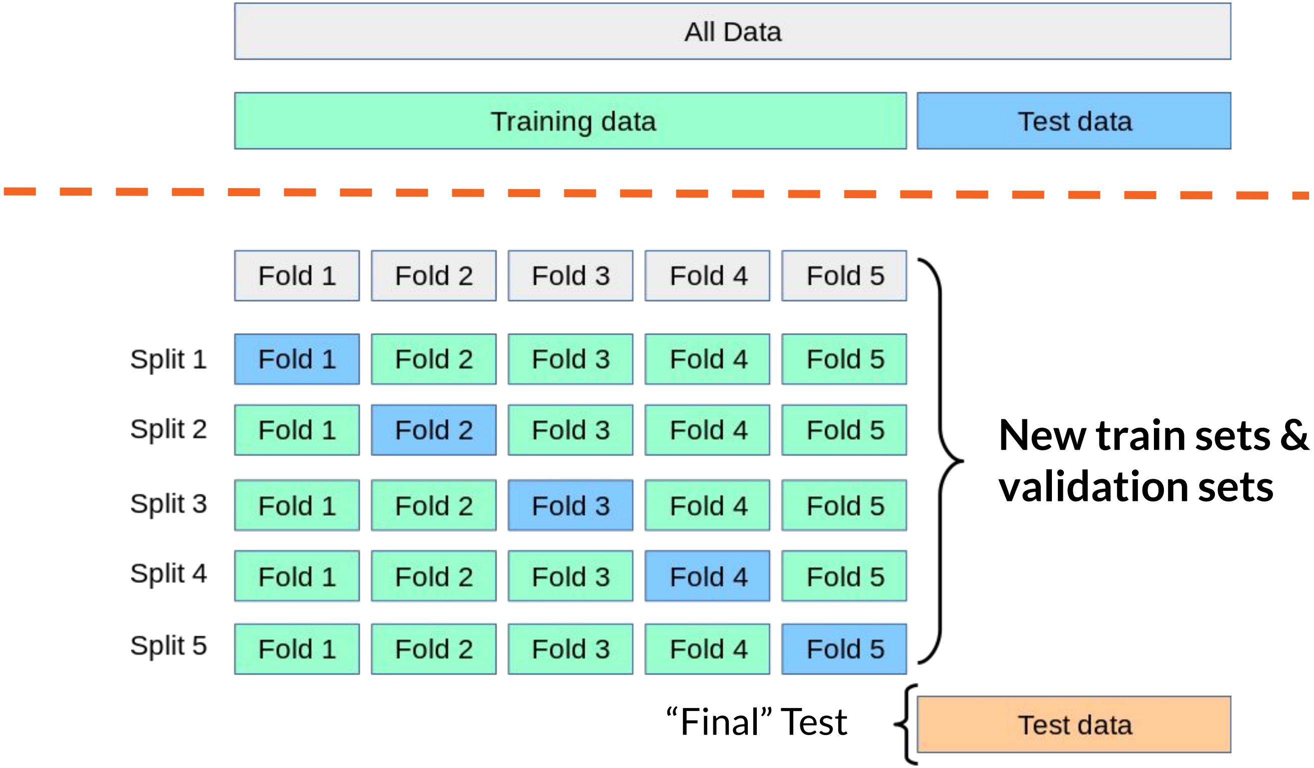
63/7/30 train/val/test split in second step (since the 70% train set gets split into 70*90% → train and 70*10% → validation)

One Validation Set



https://algotrading101.com/learn/train-test-split/

Cross Validation



All Data

Training data | Test data

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 3	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 4	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 5	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5

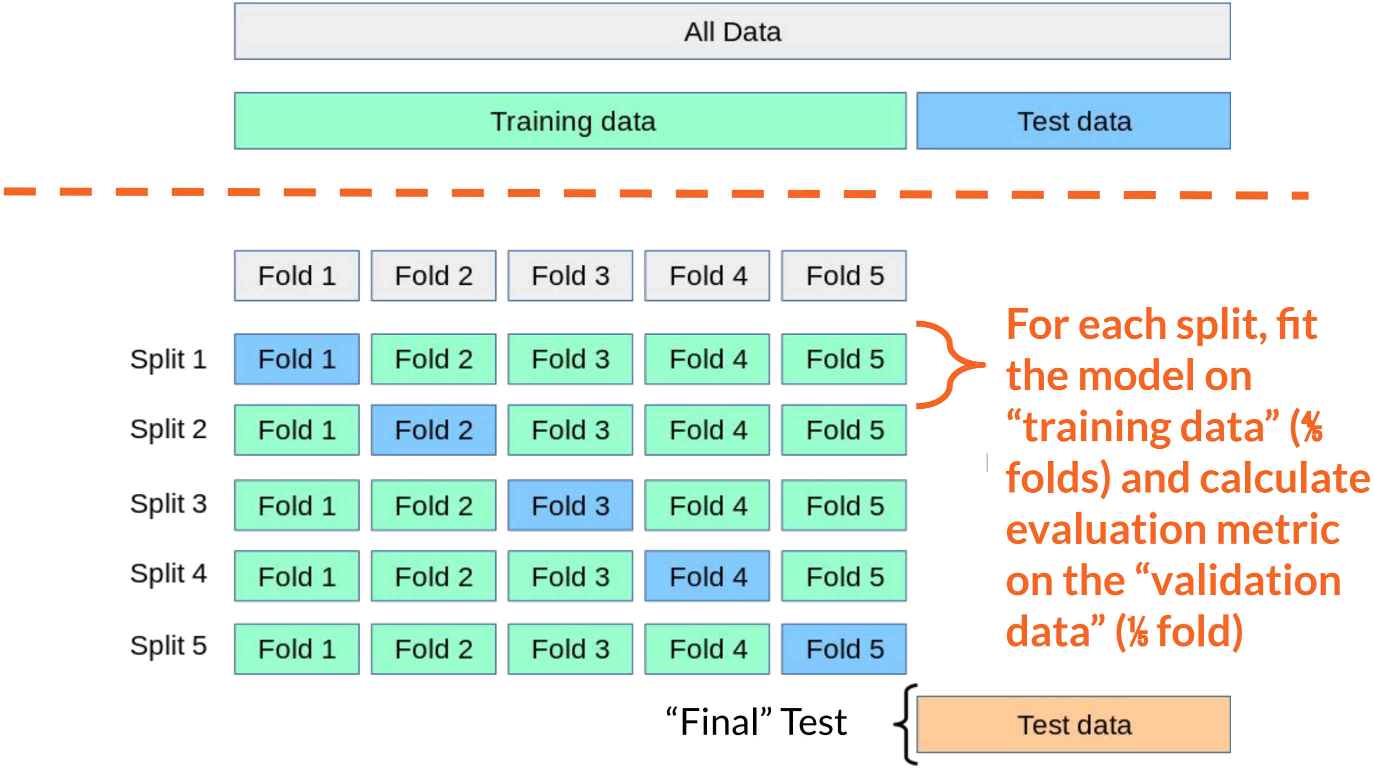
New train sets & validation sets

"Final" Test

Test data

https://scikit-learn.org/stable/modules/cross_validation.html

Cross Validation



All Data

Training data

Test data

Fold 1 Fold 2 Fold 3 Fold 4 Fold 5

Split 1 Fold 1 Fold 2 Fold 3 Fold 4 Fold 5

Split 2 Fold 1 Fold 2 Fold 3 Fold 4 Fold 5

Split 3 Fold 1 Fold 2 Fold 3 Fold 4 Fold 5

Split 4 Fold 1 Fold 2 Fold 3 Fold 4 Fold 5

Split 5 Fold 1 Fold 2 Fold 3 Fold 4 Fold 5

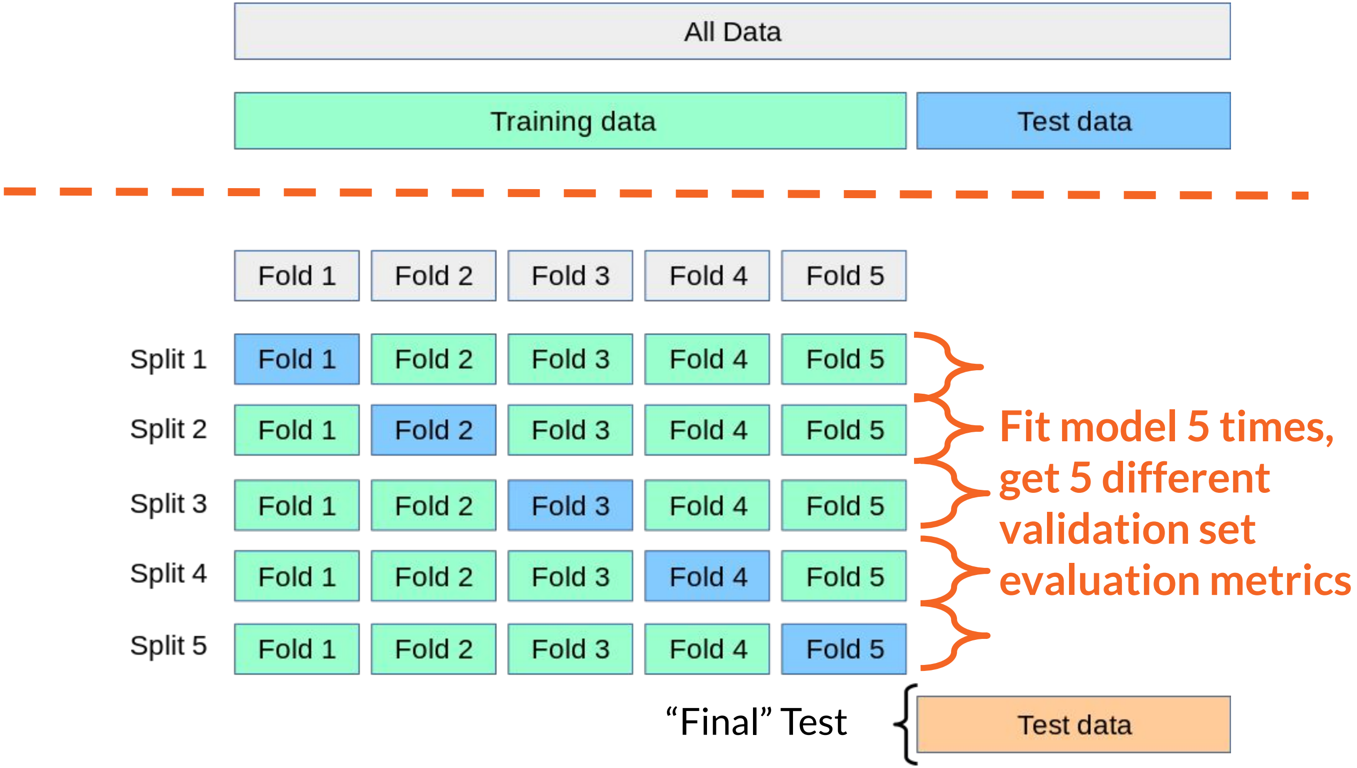
For each split, fit the model on “training data” (⅘ folds) and calculate evaluation metric on the “validation data” (⅕ fold)

“Final” Test

Test data

https://scikit-learn.org/stable/modules/cross_validation.html

Cross Validation



All Data

🟩 Training data | 🟦 Test data

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 1	🟦 Fold 1	🟩 Fold 2	🟩 Fold 3	🟩 Fold 4	🟩 Fold 5
Split 2	🟩 Fold 1	🟦 Fold 2	🟩 Fold 3	🟩 Fold 4	🟩 Fold 5
Split 3	🟩 Fold 1	🟩 Fold 2	🟦 Fold 3	🟩 Fold 4	🟩 Fold 5
Split 4	🟩 Fold 1	🟩 Fold 2	🟩 Fold 3	🟦 Fold 4	🟩 Fold 5
Split 5	🟩 Fold 1	🟩 Fold 2	🟩 Fold 3	🟩 Fold 4	🟦 Fold 5

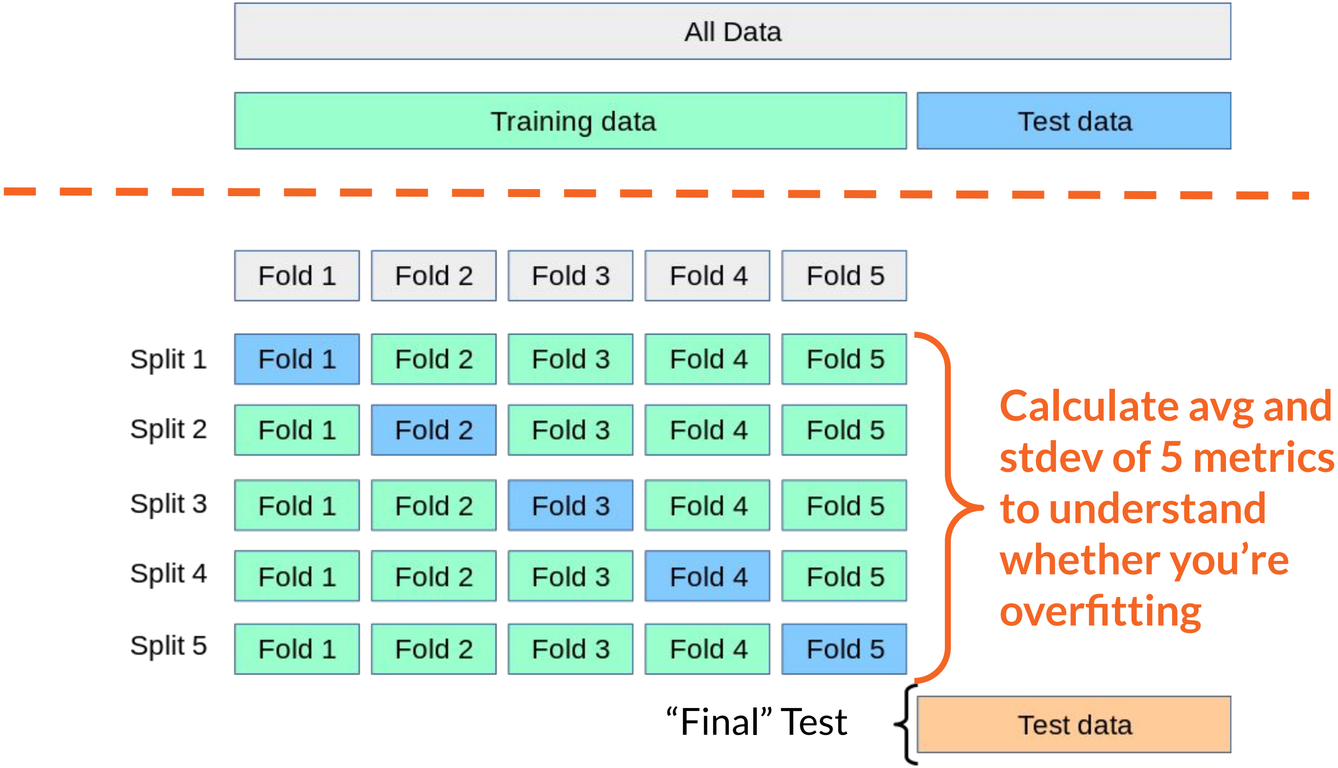
Fit model 5 times, get 5 different validation set evaluation metrics

“Final” Test

Test data

https://scikit-learn.org/stable/modules/cross_validation.html

Cross Validation



All Data

Training data | Test data

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 3	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 4	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 5	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5

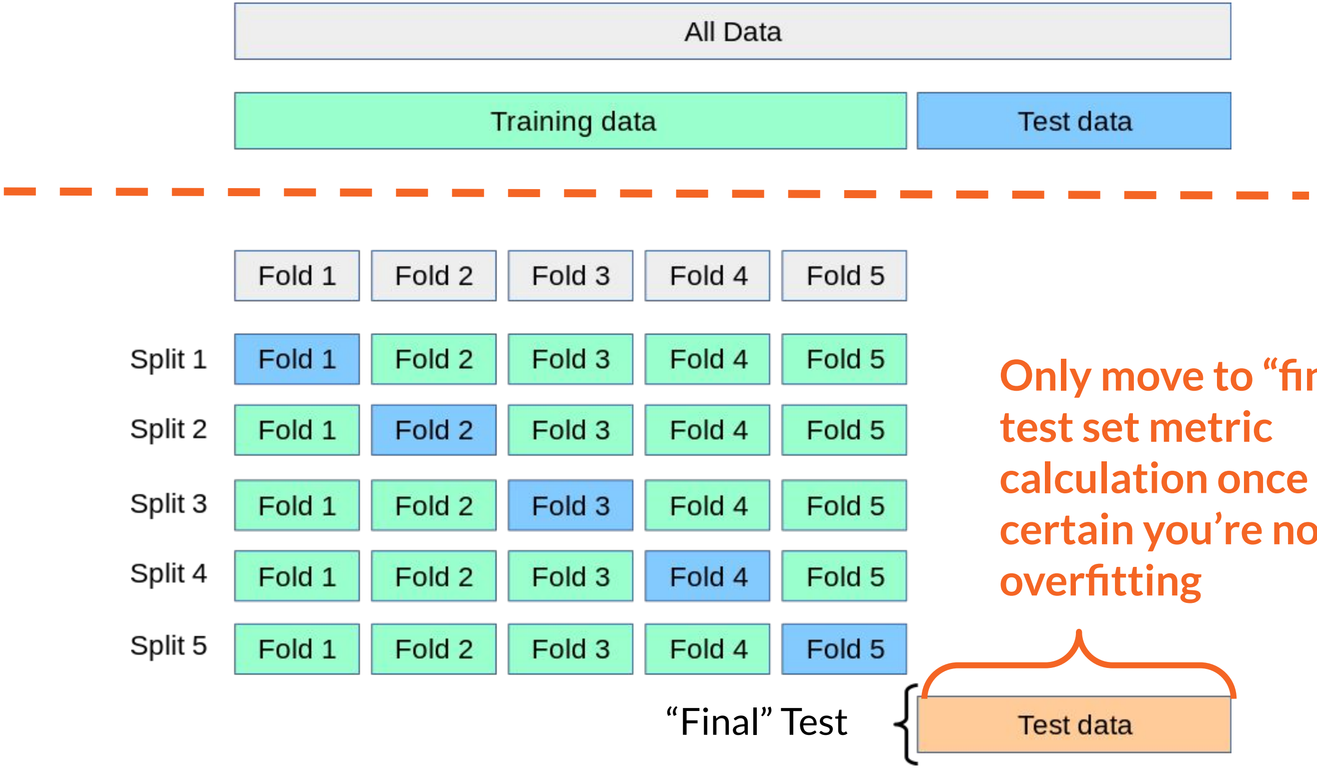
Calculate avg and stdev of 5 metrics to understand whether you’re overfitting

“Final” Test

Test data

https://scikit-learn.org/stable/modules/cross_validation.html

Cross Validation



All Data

Training data | Test data

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 3	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 4	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 5	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5

“Final” Test { Test data

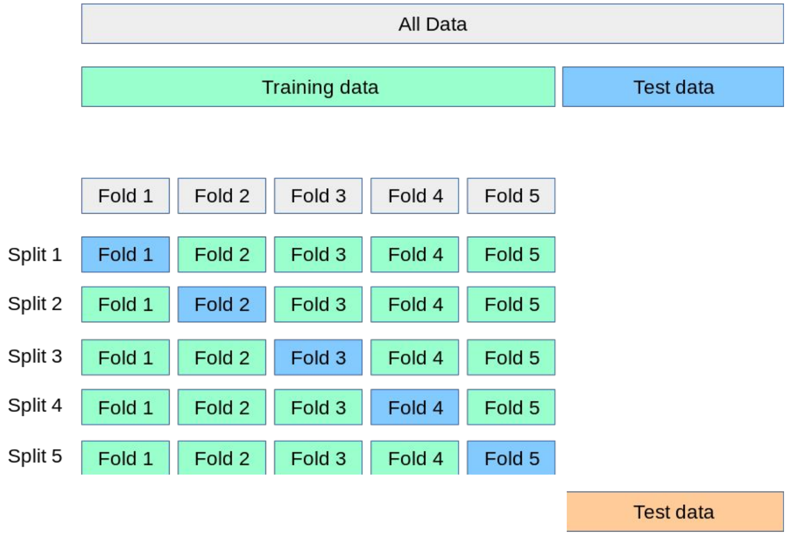
Only move to “final” test set metric calculation once certain you’re not overfitting

https://scikit-learn.org/stable/modules/cross_validation.html

Why CV?

- Avoids pitfalls discussed earlier (getting lucky with one test set)
- Allows you to update your model without overfitting to a specific set of data
- Using CV with 5 folds → 5 values of an evaluation metric. Use summary stats to understand model performance **across different data samples**

Options for cross-validation



All Data

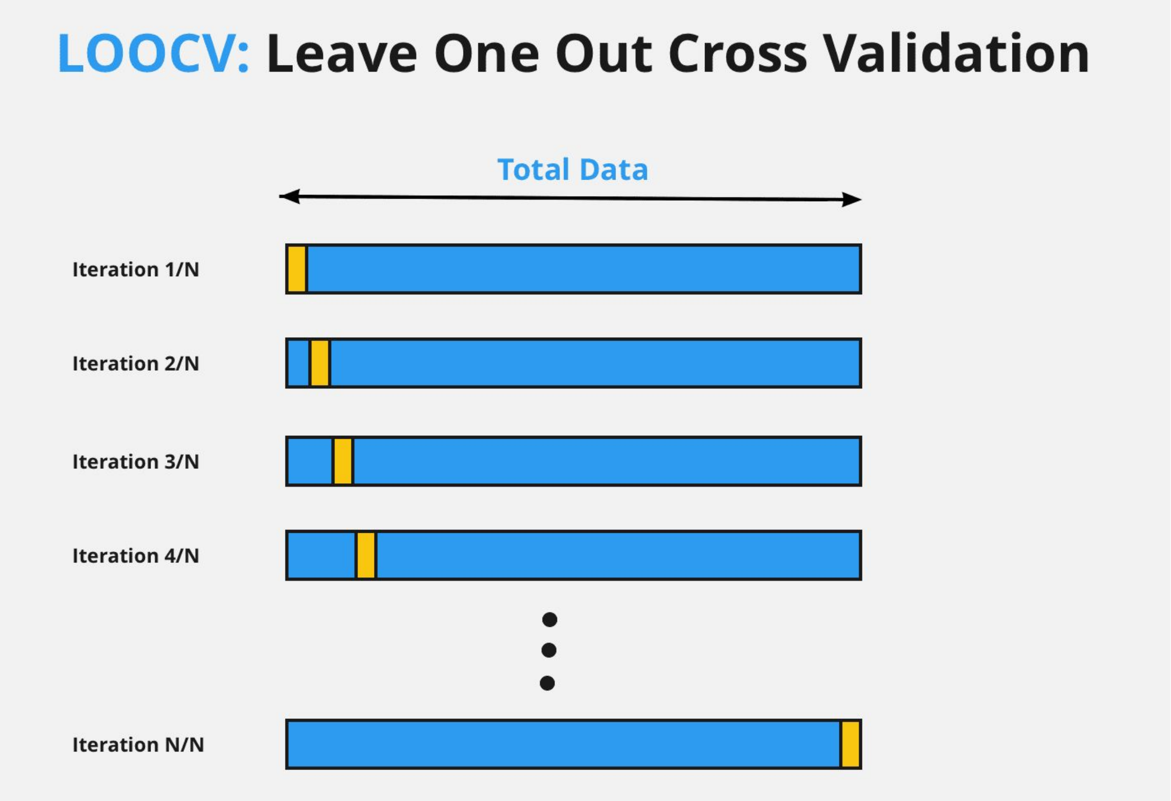
Training data | Test data

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 3	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 4	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 5	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5

Test data

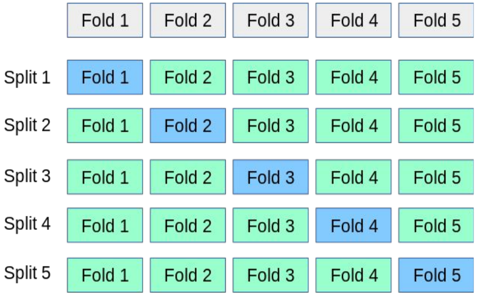
- Partitions or independent random splits
- Number of splits
- Special case: leave-one-out (LOO) cross-validation

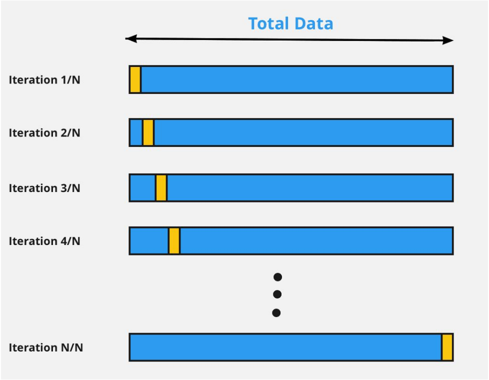
https://scikit-learn.org/stable/modules/cross_validation.html



https://dataaspirant.com/7-loocv-leave-one-out-cross-validation/

Cross Validation Variance

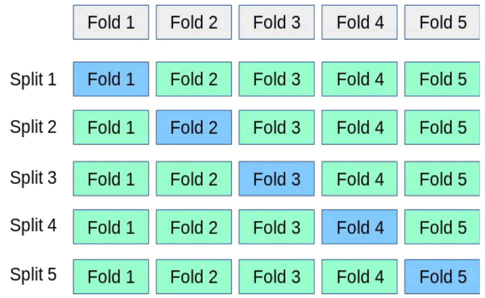




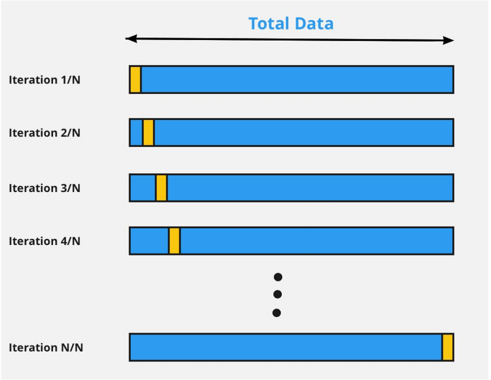
- If you increase the number of splits during cross-validation, does the *variance* or your estimated evaluation metric (the average of metrics across splits) tend to **increase or decrease**?

https://scikit-learn.org/stable/modules/cross_validation.html | https://dataaspirant.com/7-loocv-leave-one-out-cross-validation/

Cross Validation Variance



	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 1	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 2	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 3	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 4	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
Split 5	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5



- If you increase the number of splits during cross-validation, does the *variance* or your estimated evaluation metric (the average of metrics across splits) increase or decrease?
- Generally, having more metrics to average (e.g. 1 per data point in the LOO case) → more likely to get outliers → variance increases

https://scikit-learn.org/stable/modules/cross_validation.html | https://dataaspirant.com/7-loocv-leave-one-out-cross-validation/

How to generate cross-validation splits

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size = 0.3)
for f in folds:
    X_train[f], X_valid[f], y_train[f], y_valid[f] =
      train_test_split(X_train, y_train, test_size = 0.1)
```

How to generate cross-validation splits

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
   train_test_split(X, y, test_size = 0.3)
for f in folds:
   X_train[f], X_valid[f], y_train[f], y_valid[f] =
     train_test_split(X_train, y_train, test_size = 0.1)
```

should I use a fixed random seed here?

How to generate cross-validation splits

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size = 0.3)
for f in folds:
    X_train[f], X_valid[f], y_train[f], y_valid[f] =
      train_test_split(X_train, y_train, test_size = 0.1)
```

No! That would just give me the same split `folds` times

https://m.facebook.com/convolutionalmemes/posts/788912368229932

How to generate cross-validation splits

```
from sklearn.model_selection import KFold

kf = KFold(n_splits=2)
```

New package

https://scikit-learn.org/stable/modules/cross_validation.html

How to generate cross-validation splits

```
from sklearn.model_selection import KFold

   kf = KFold(n_splits=2)
```

Define how many splits to use

https://scikit-learn.org/stable/modules/cross_validation.html

How to generate cross-validation splits

```
from sklearn.model_selection import KFold

   X = ["a", "b", "c", "d"]

   kf = KFold(n_splits=2)
```

Defining some sample data (think of this as a column of input data in the training set)

https://scikit-learn.org/stable/modules/cross_validation.html

How to generate cross-validation splits

```
from sklearn.model_selection import KFold
   X = ["a", "b", "c", "d"]
   kf = KFold(n_splits=2)
   for train, test in kf.split(X):
      print(f"{train} {test}")
```

Split the input data X

https://scikit-learn.org/stable/modules/cross_validation.html

How to generate cross-validation splits

```
from sklearn.model_selection import KFold
   X = ["a", "b", "c", "d"]
   kf = KFold(n_splits=2)
   for train, test in kf.split(X):
      print(f"{train} {test}")
```

Returns the indices assigned to train/test

https://scikit-learn.org/stable/modules/cross_validation.html

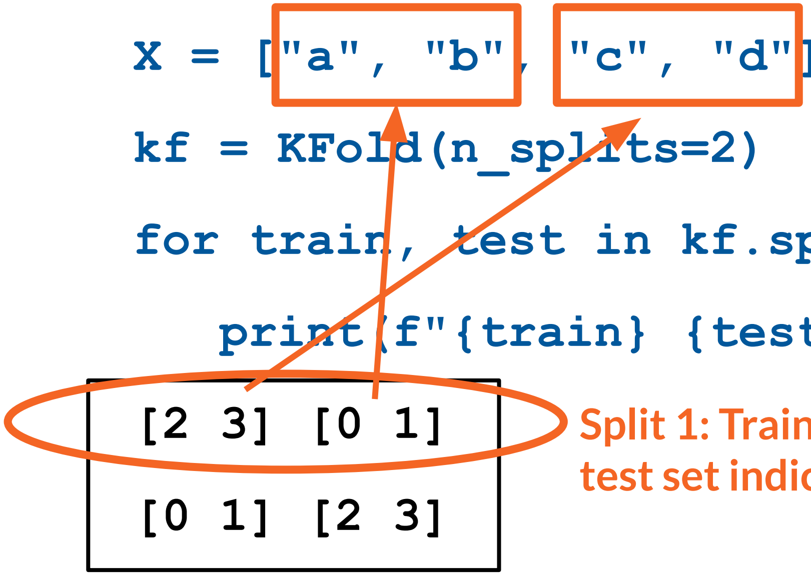
How to generate cross-validation splits

```
from sklearn.model_selection import KFold
   X = ["a", "b", "c", "d"]
   kf = KFold(n_splits=2)
   for train, test in kf.split(X):
      print(f"{train} {test}")
```

```
[2 3] [0 1]
[0 1] [2 3]
```

https://scikit-learn.org/stable/modules/cross_validation.html

How to generate cross-validation splits



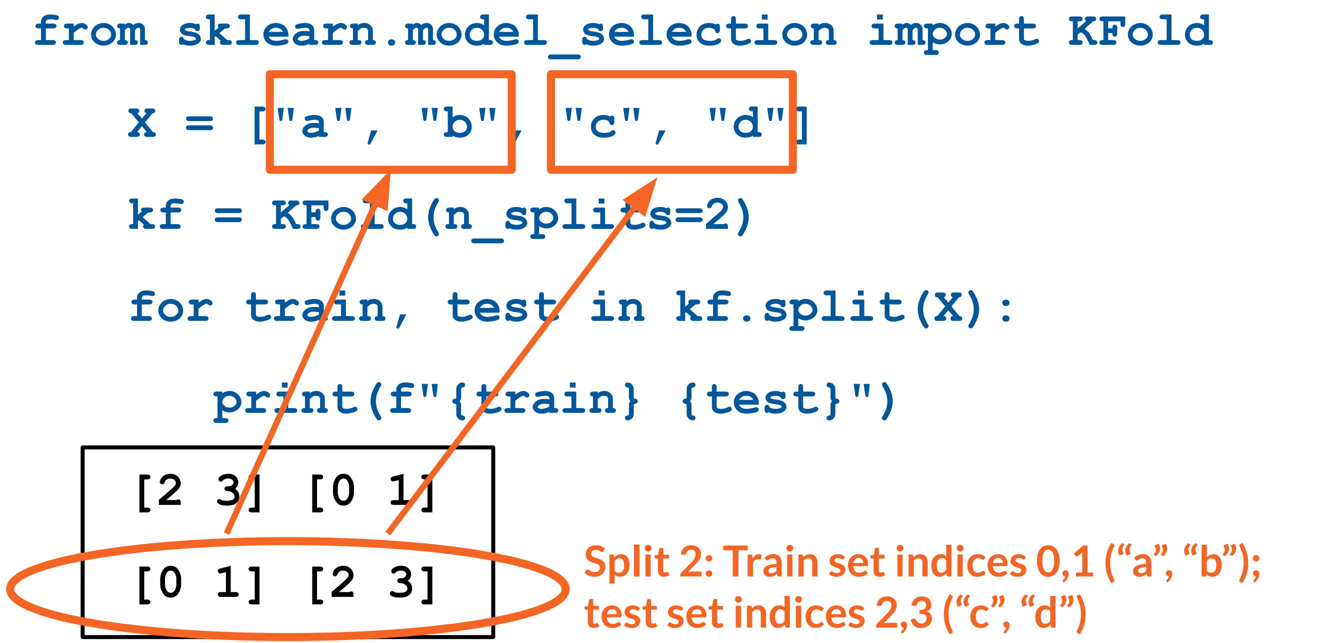
```
from sklearn.model_selection import KFold
    X = ["a", "b", "c", "d"]
    kf = KFold(n_splits=2)
    for train, test in kf.split(X):
        print(f"{train} {test}")
```

```
[2 3] [0 1]
[0 1] [2 3]
```

Split 1: Train set indices 2,3 (“c”, “d”); test set indices 0,1 (“a”, “b”)

https://scikit-learn.org/stable/modules/cross_validation.html

How to generate cross-validation splits



```
from sklearn.model_selection import KFold
    X = ["a", "b", "c", "d"]
    kf = KFold(n_splits=2)
    for train, test in kf.split(X):
       print(f"{train} {test}")
```

```
[2 3] [0 1]
[0 1] [2 3]
```

Split 2: Train set indices 0,1 (“a”, “b”);
test set indices 2,3 (“c”, “d”)

https://scikit-learn.org/stable/modules/cross_validation.html

How to generate cross-validation splits

```
from sklearn.model_selection import KFold
    X = ["a", "b", "c", "d"]
    kf = KFold(n_splits=2)
    for train, test in kf.split(X):
       print(f"{train} {test}")
```

```
[2 3] [0 1]
[0 1] [2 3]
```

Calculate evaluation metrics on split 1

Calculate evaluation metrics on split 2

https://scikit-learn.org/stable/modules/cross_validation.html

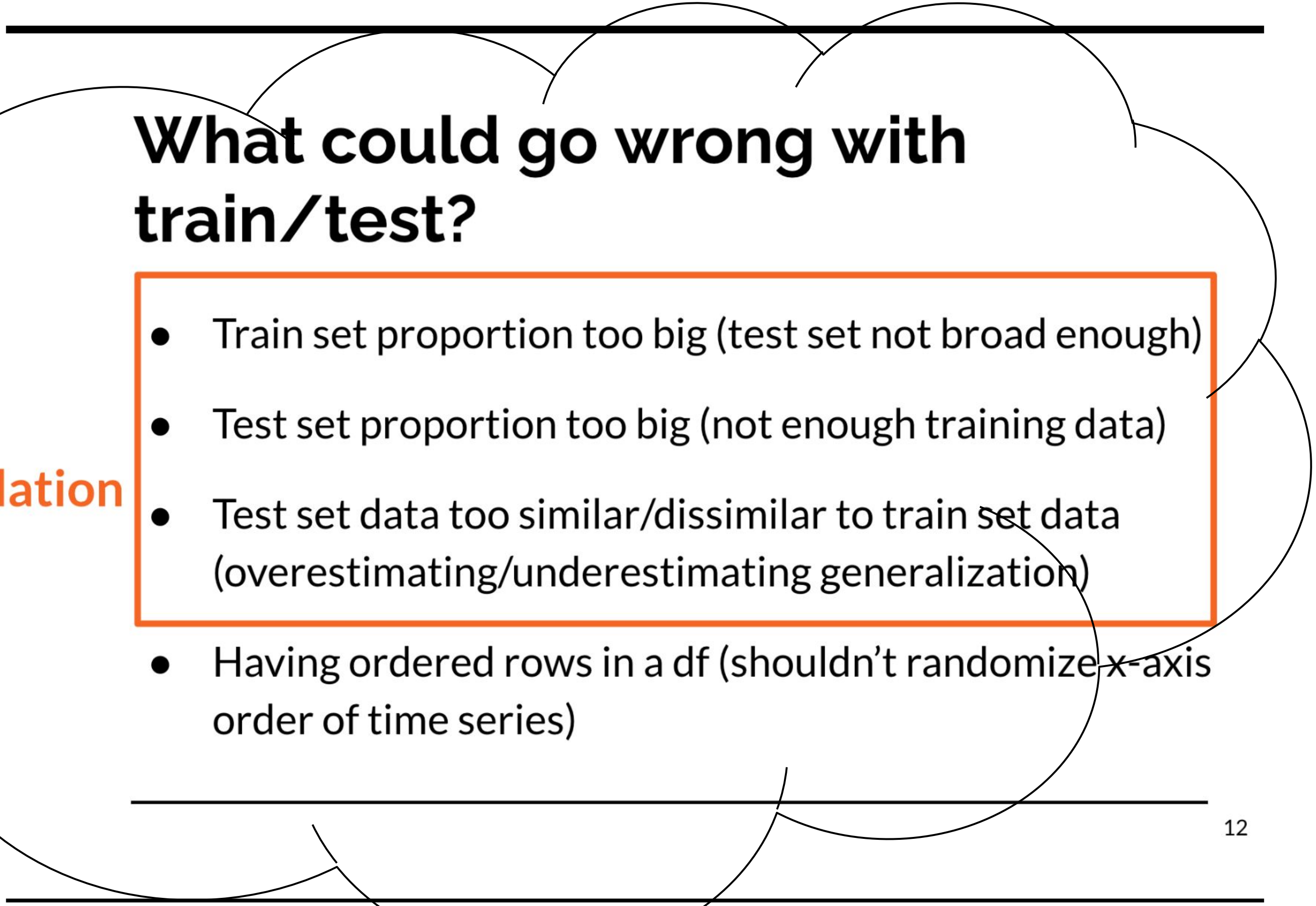
How to generate cross-validation splits

```
from sklearn.model_selection import KFold
   X = ["a", "b", "c", "d"]
   kf = KFold(n_splits=2)
   for train, test in kf.split(X):
      print(f"{train} {test}")
```

```
[2 3] [0 1]
[0 1] [2 3]
```

Calculate the mean / stdev of evaluation metrics across splits

https://scikit-learn.org/stable/modules/cross_validation.html



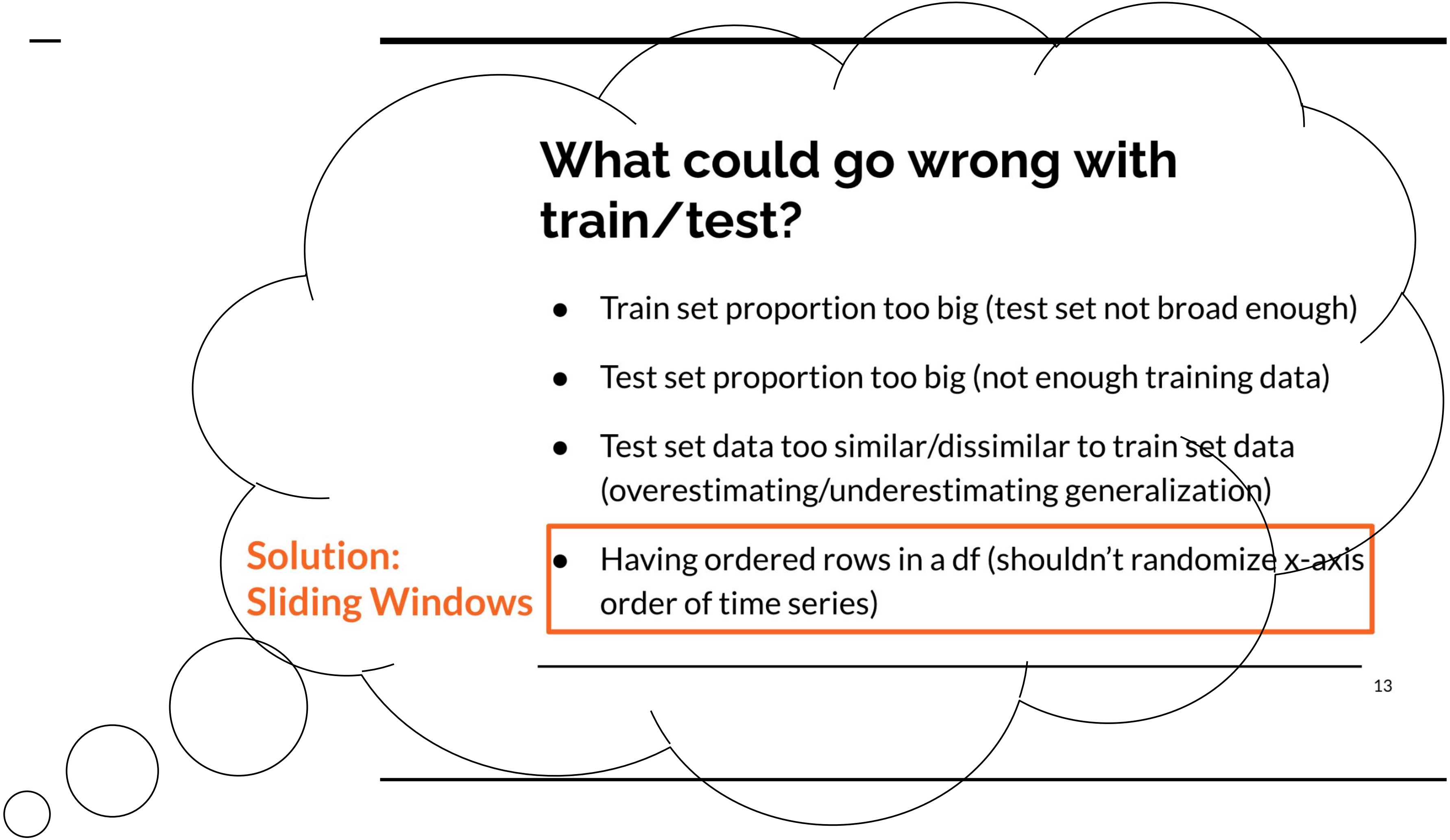
What could go wrong with train/test?

- Train set proportion too big (test set not broad enough)
- Test set proportion too big (not enough training data)
- Test set data too similar/dissimilar to train set data (overestimating/underestimating generalization)
- Having ordered rows in a df (shouldn't randomize x-axis order of time series)

Solution: Cross-Validation

When you realize k-fold cross validation can only validate your models, not yourself...

https://pub.towardsai.net/importance-of-k-fold-cross-validation-in-machine-learning-a0d76f49493e



What could go wrong with train/test?

- Train set proportion too big (test set not broad enough)
- Test set proportion too big (not enough training data)
- Test set data too similar/dissimilar to train set data (overestimating/underestimating generalization)
- Having ordered rows in a df (shouldn't randomize x-axis order of time series)

Solution:
Sliding Windows

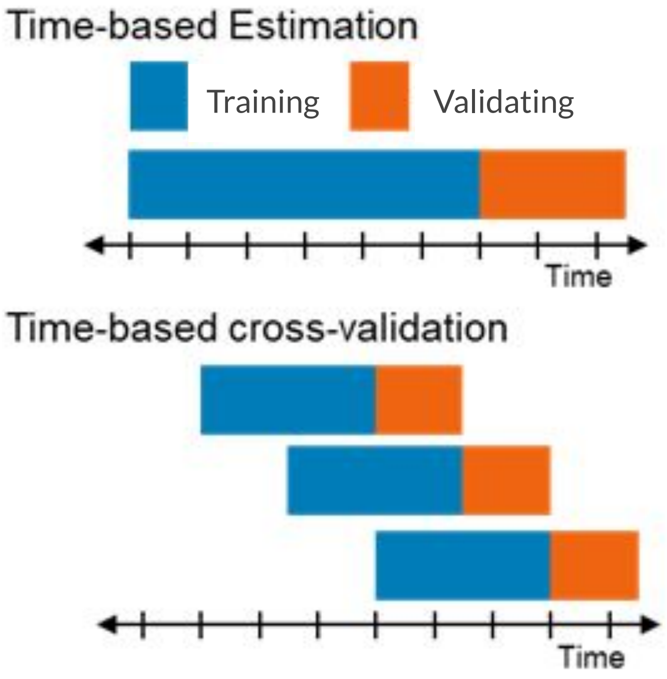
Sliding Windows

- For time series: take the concept of cross validation, but tweak it so that you can’t use the **future** to model the **past**
  - This is cheating (“data leakage”) and will give you overly-good results!

Sliding Windows

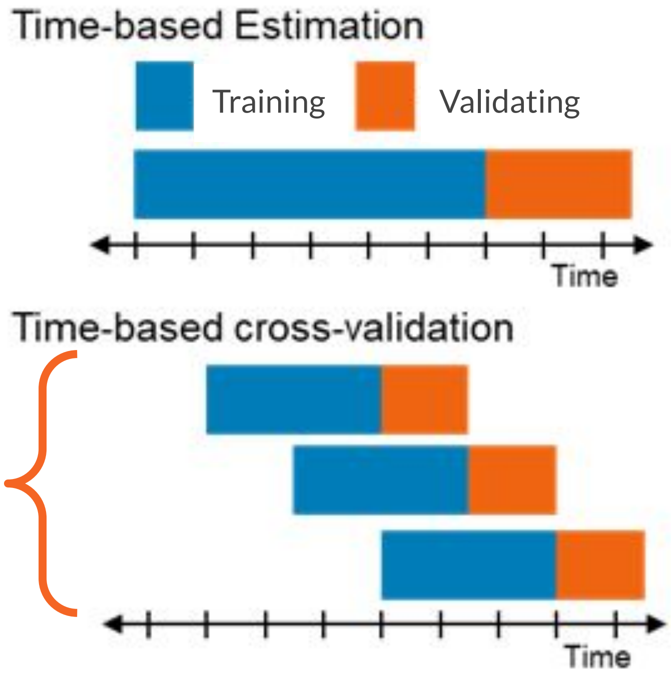
- For time series: take the concept of cross validation, but tweak it so that you can't use the **future** to model the **past**
  - This is cheating ("data leakage") and will give you overly-good results!
- Instead, **slide** your test/validation sets forward by a "time **window**" unit

Sliding Windows



https://towardsdatascience.com/time-based-cross-validation-d259b13d42b8

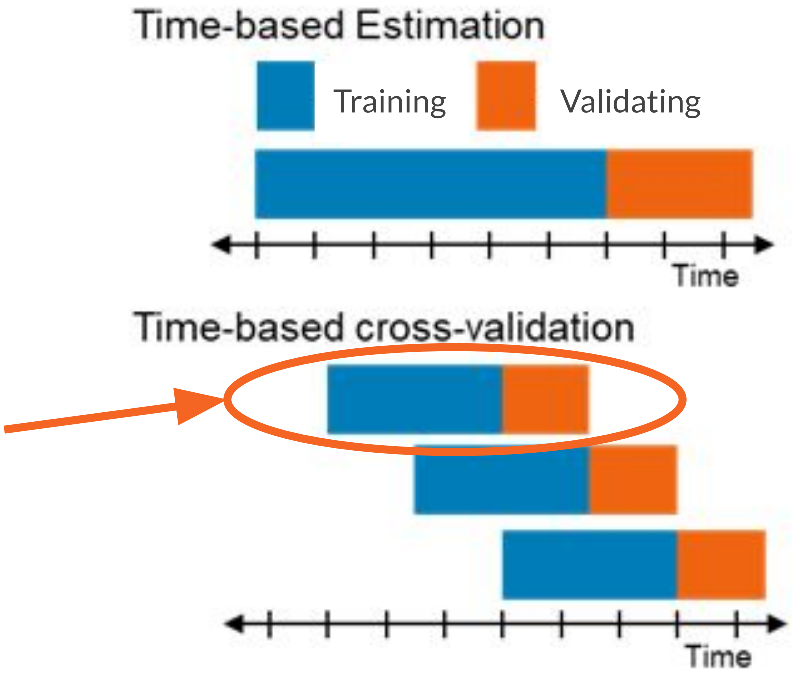
Sliding Windows



Multiple splits, each one with a different train and validation set

https://towardsdatascience.com/time-based-cross-validation-d259b13d42b8

Sliding Windows



Never peeking at "future" test data when training the model

https://towardsdatascience.com/time-based-cross-validation-d259b13d42b8

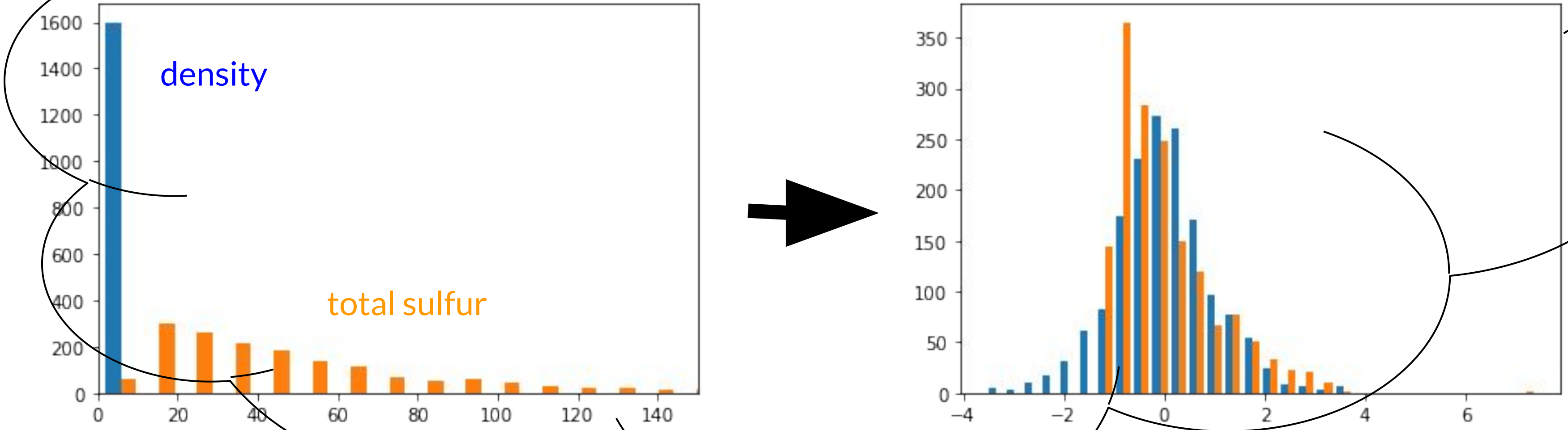
Data Leakage

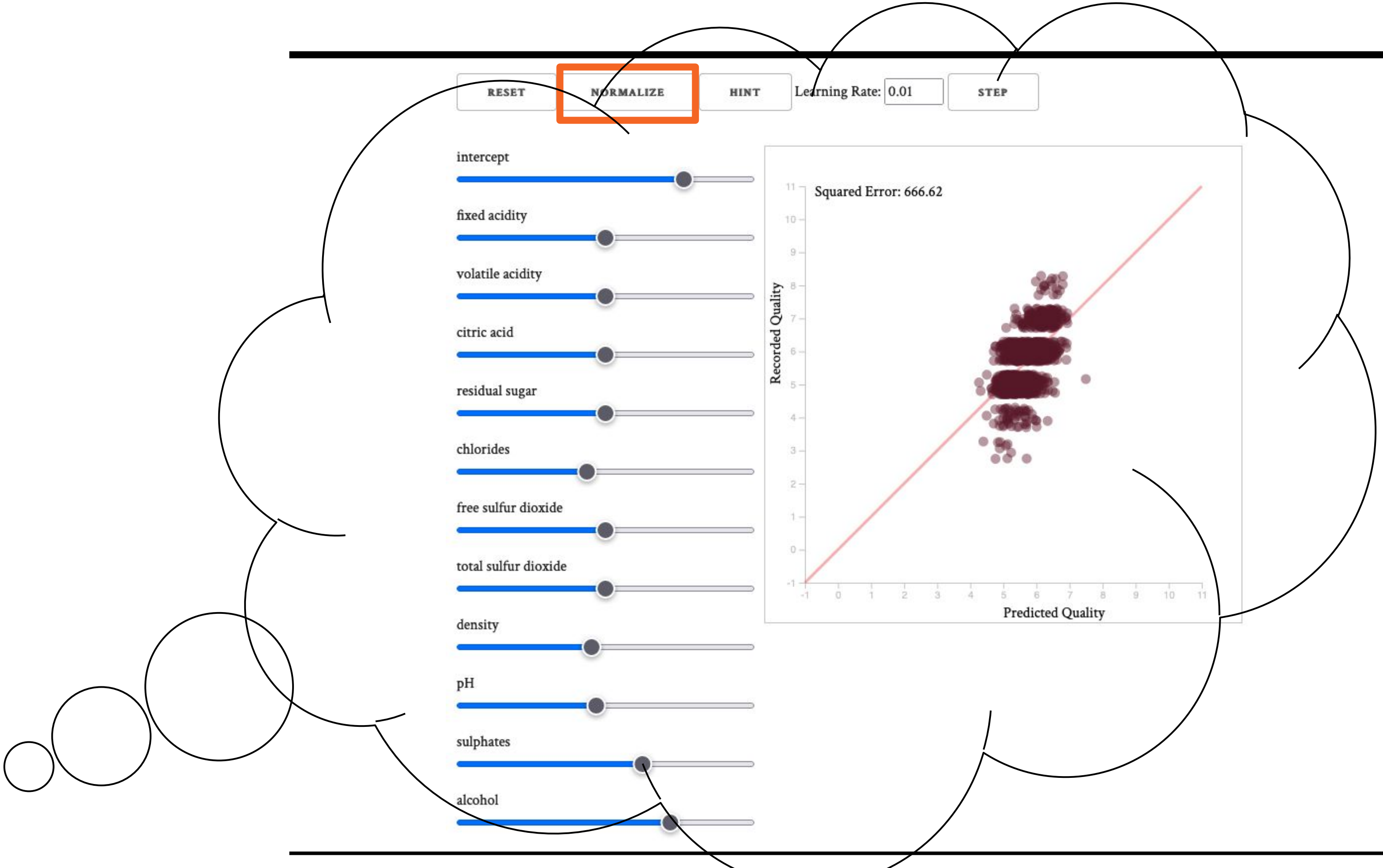
- We don’t want to “leak” information between the train / validation / test sets, in order to keep the val / test set evaluations **fully separate** from the train set evaluations
  - “Leaking” via time

Data Leakage

- We don’t want to “leak” information between the train / validation / test sets, in order to keep the val / test set evaluations **fully separate** from the train set evaluations
    - “Leaking” via time
    - **“Leaking” via normalization**

Input variables are on different scales: *normalize* with z-scores





Learning Rate: 0.01

Squared Error: 666.62

https://mimno.infosci.cornell.edu/info2950/interactive/wine.html

Normalizing → data leakage?

- Which of the following would prevent data leakage when normalizing data?

A: First generate train/test splits, then normalize within train set, and separately normalize within test set

B: First normalize all the data, then generate train/test splits

Normalizing → data leakage?

- Which of the following would prevent data leakage when normalizing data?

A: First generate train/test splits, then normalize within train set, and separately normalize within test set

If you instead normalize across all the data, the mean and stdev you calculate will incorporate data across both train/test sets, effectively “leaking” information

How to do this CV stuff in Python?

Check the Scikit Learn documentation: https://scikit-learn.org/stable/modules/cross_validation.html