UNIVERSITY OF WARWICK

Lecture 7

Evaluating and Comparing Models: Binary Classification

Gabriele Pergola and Amina Asif

Machine Learning Algorithms and Practice
CS429

October 2026

Credits: Some slides adapted from Fayyaz Minhas

Evaluation

- We train **supervised learning** algorithms to learn models that will be able to make classifications or class predictions on **unseen data**.

- l We need a means of **evaluating** the efficacy of these models.
  - **How good** is my classifier?
  - How can I **quantify** its performance? Generalisation?
  - **Which** of my classifiers A and B does **better**?

Outline

- Dataset split

- Model evaluation metrics
  - Binary classifiers
  - Multi-class classifiers
  - Multi-label classifiers

- Model comparison

Supervised Classification

Supervised Text Classification

- **Assumption:** We have a manually labelled dataset, e.g.:

  $\mathbf{d_1}$: ‘That’s really good, I love it’ → **positive**

  $\mathbf{d_2}$: ‘It was boring, don’t recommend it’ → **negative**

  …

  $\mathbf{d_n}$: ‘I wouldn’t go again, awful’ → **negative**

- **If not**, we need to **find one** or **label one ourselves**.

Supervised Classification: **Decisions to Make**

1. **Split the dataset** into train/dev/test sets.
   - or often just train/test.

2. What **features** are we going to use to represent the documents?

3. What **classifier** are we going to use?
   - Choose **settings, parameters**, etc. for the classifier.

Evaluation on Small Datasets

- The **holdout method** reserves a certain amount for testing and uses the remainder for training.
  - Usually: one third for testing, the rest for training
- For small or **unbalanced datasets**, samples might not be **representative**.
  - Few or no instances of some classes
- **Stratified sample**: advanced version of balancing the data.
  - Make sure that each class is represented with approximately equal proportions in both subsets

1. Splitting the Dataset: The problem of Generalisation

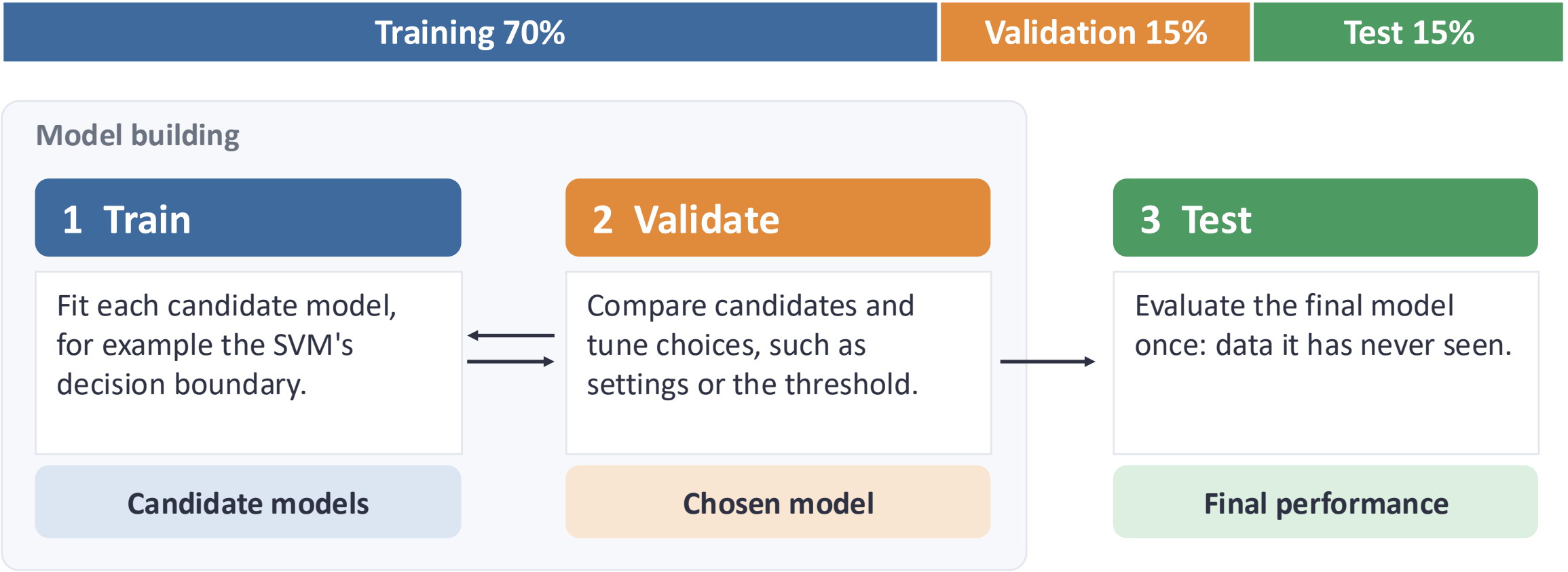
- We can **split the dataset into 3 parts**:
  - Training set → the largest set as we want proper training
  - Development set
  - Test set



  - **Tweak classifier based on the development set**, then test it on the test set.
    - **Tweaking and testing on the test set may lead to overfitting** (doing the right things specifically for that test set, not necessarily generalisable)

Training, validation and test sets

All labelled data (true answers known)

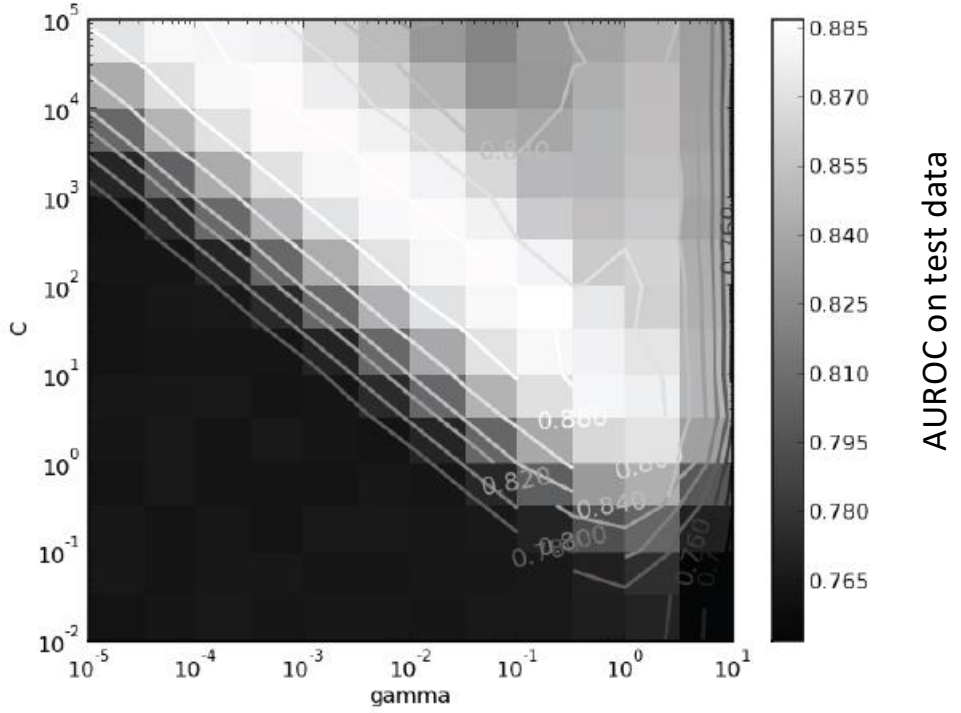


- The **test set** stays locked away until the end: using it to make choices would make the final scores over-optimistic.

Model Hyperparameter Selection

- **Grid Search**
  - Exhaustive Search through Cross-Validation
    - Recommended: Nested Cross-Validation or separate test set

- There can be a range of parameter values that yield optimal values and these equivalent points in the parameter space fall along a ridge



Ben-Hur, Asa, and Jason Weston. 2010. "A User's Guide to Support Vector Machines." In *Data Mining Techniques for the Life Sciences*, edited by Oliviero Carugo and Frank Eisenhaber, 223–39. Methods in Molecular Biology 609. Humana Press. http://dx.doi.org/10.1007/978-1-60327-241-4_13.

What is Overfitting?

- **Overly adjusting** our classifier and features **to a specific test set**.

- The classifier **may not generalise** to new instances beyond the current test set.



Pretty much like designing a mattress based on a single observation of sleeping position!

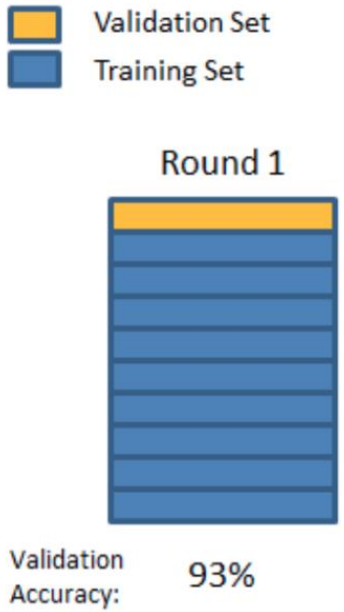
How to Avoid Overfitting?

- Having train/dev/test, **tweak our classifier for the dev set**.
  - Then apply to the **test set**, does it **generalise**?

- **Cross-validation**.

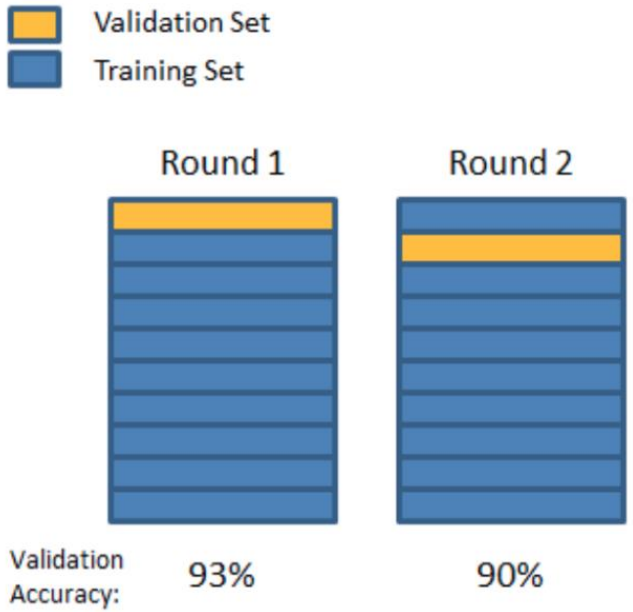
Cross-Validation

- **Cross-validation:** train and test on different “folds”
  - e.g.: 10-fold cross-validation, split the data into 10 parts.
  - each time 1 fold is used for testing, the other 9 for training.
  - after all 10 runs, compute the average performance.



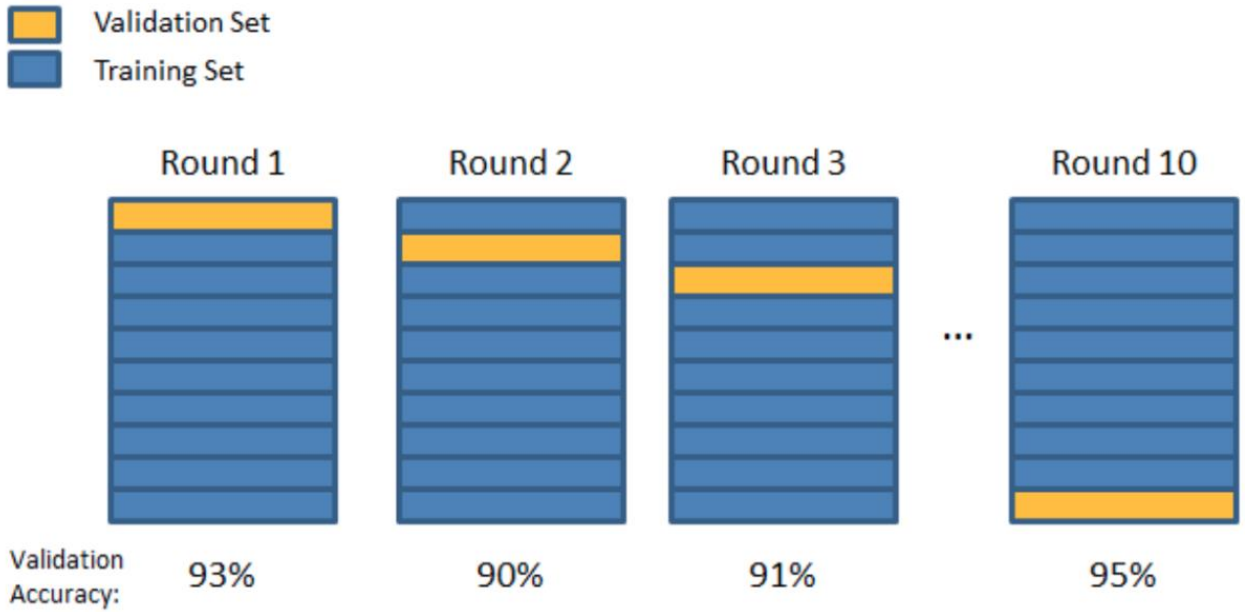
Cross-Validation

- **Cross-validation:** train and test on different “folds”
  - e.g.: 10-fold cross-validation, split the data into 10 parts.
  - each time 1 fold is used for testing, the other 9 for training.
  - after all 10 runs, compute the average performance.



Cross-Validation

- **Cross-validation:** train and test on different “folds”
  - e.g.: 10-fold cross-validation, split the data into 10 parts.
  - each time 1 fold is used for testing, the other 9 for training.
  - after all 10 runs, compute the average performance.



Making the Most of the Data

Once evaluation is complete, **all the data** can be used to build the final classifier.

- Generally, the **larger the training data** the better the classifier.
- The **larger the test** dataset the more accurate the error estimate.

3. Choosing a Classifier

- Many different classifiers exist, well-known classifiers include:
  - **Random Forest**
  - **Logistic Regression**
  - **Support Vector Machines** (SVM)

- Classifiers can be **binary** (k = 2) or **multiclass** (k > 2).

Evaluation of Binary Classification

Evaluation of Classification

Evaluation is different for binary and multiclass classification.

- **Binary**: we generally have a **positive** and a **negative class** (spam vs non-spam, medical test positive vs negative, exam pass vs fail).
  → Classification **errors can only go to the other class**.
- **Multiclass**: multiple categories, may have different level of importance.
  → Classification **errors can go to any other class**.

→ That is why we need to have a look at how to conduct proper evaluation.

Accuracy

- Two-Class Classification
  - **Accuracy: Percentage of Correct Predictions**

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

- Assumption
  - The data set is balanced
  - Misclassification of any class is equally bad
  - The threshold used for classification is optimal

Thought Experiment

- Consider the data
  - Assume that the data is balanced (equal number of positive and negative examples)
  - Consider a random classifier
    - This classifier will generate a random score of any example given as input
    - **What will be its accuracy?**
  - Consider a classifier which generates a score of +1 (positive class) for all inputs
    - **What will be its accuracy?**

Thought Experiment

- Consider the data
  - Assume that the data is imbalanced (#Neg>>#Pos)
  - Consider a random classifier
    - This classifier will generate a random score of any example given as input
    - **What will be its accuracy?**
  - Consider a classifier which generates a score of +1 (positive class) for all inputs
    - **What will be its accuracy?**

Evaluation of Binary Classification

- 2-by-2 contingency table:

	Actually positive	**Actually negative**
Classified as positive	True Positive (TP)	False Positive (FP)
Classified as negative	False Negative (FN)	True Negative (TN)

Evaluation of Binary Classification

- 2-by-2 contingency table:

	Actually positive	**Actually negative**
Classified as positive	True Positive (TP)	Type I error (FP)
Classified as negative	Type II error (FN)	True Negative (TN)

Evaluation of Binary Classification

- 2-by-2 contingency table:

	Actually positive	Actually negative
Classified as positive	True Positive (TP)	False Positive (FP)
Classified as negative	False Negative (FN)	True Negative (TN)

- **Precision:** ratio of items classified as positive that are correct.

$$\text{Precision} = \frac{tp}{tp + fp}$$

- **Recall:** ratio of actual positive items that are classified as positive.

$$\text{Recall} = \frac{tp}{tp + fn}$$

Evaluation of Binary Classification

- We want to optimise for both precision and recall:

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$ (harmonic mean of precision and recall)

- Equation as follows, however generally ß = 1:

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}}$$

Sensitivity, Specificity and ROC Curves

Reading a confusion matrix and choosing a classification threshold

Our running example: spotting fractures in X-rays



An AI system analyses each X-ray and predicts whether it shows a **Fracture** or **No Fracture**.

We test it on X-rays whose true diagnosis a radiologist has already confirmed, and measure how often it is right for each kind of image.

Throughout, **Fracture is the positive class.**

Fracture (positive)

The X-ray shows a broken bone

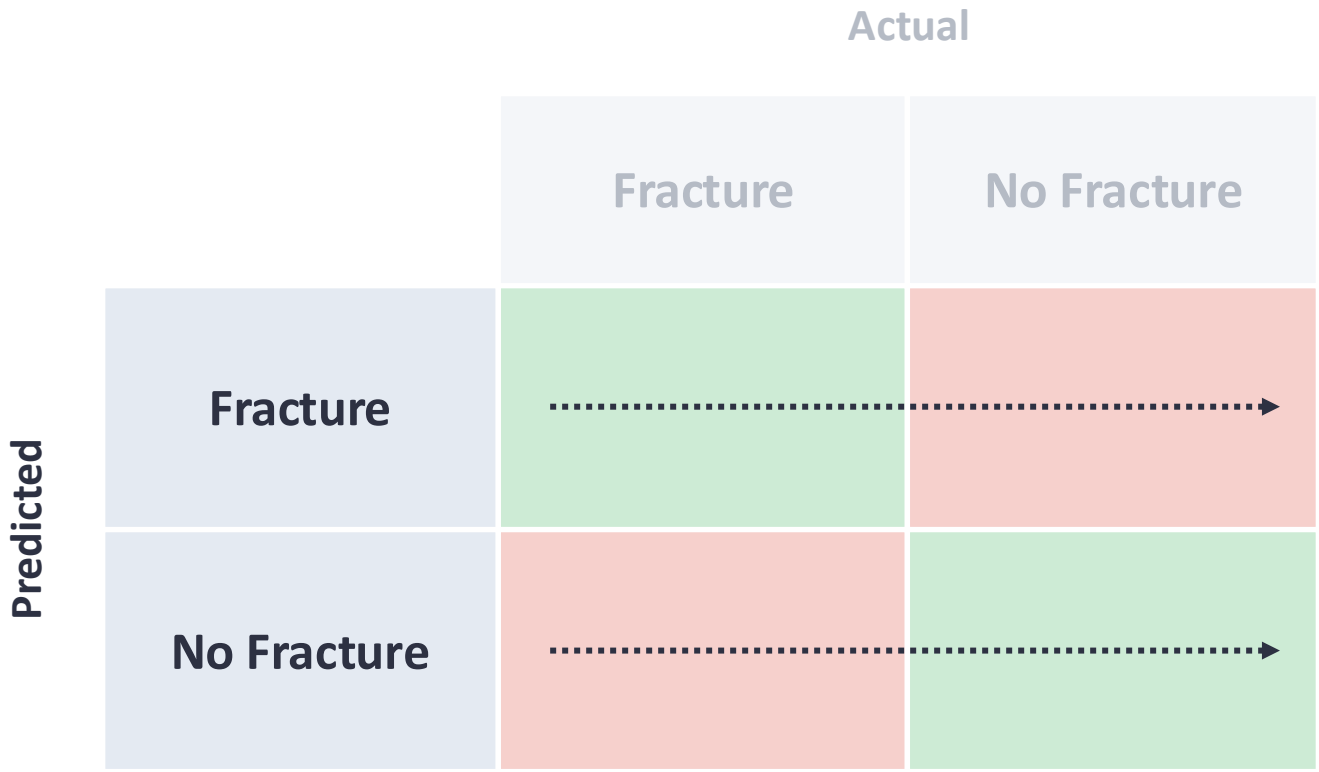
No Fracture (negative)

The bone in the X-ray is intact

Rows show what was predicted

Each row collects the X-rays the AI system assigned to one class.

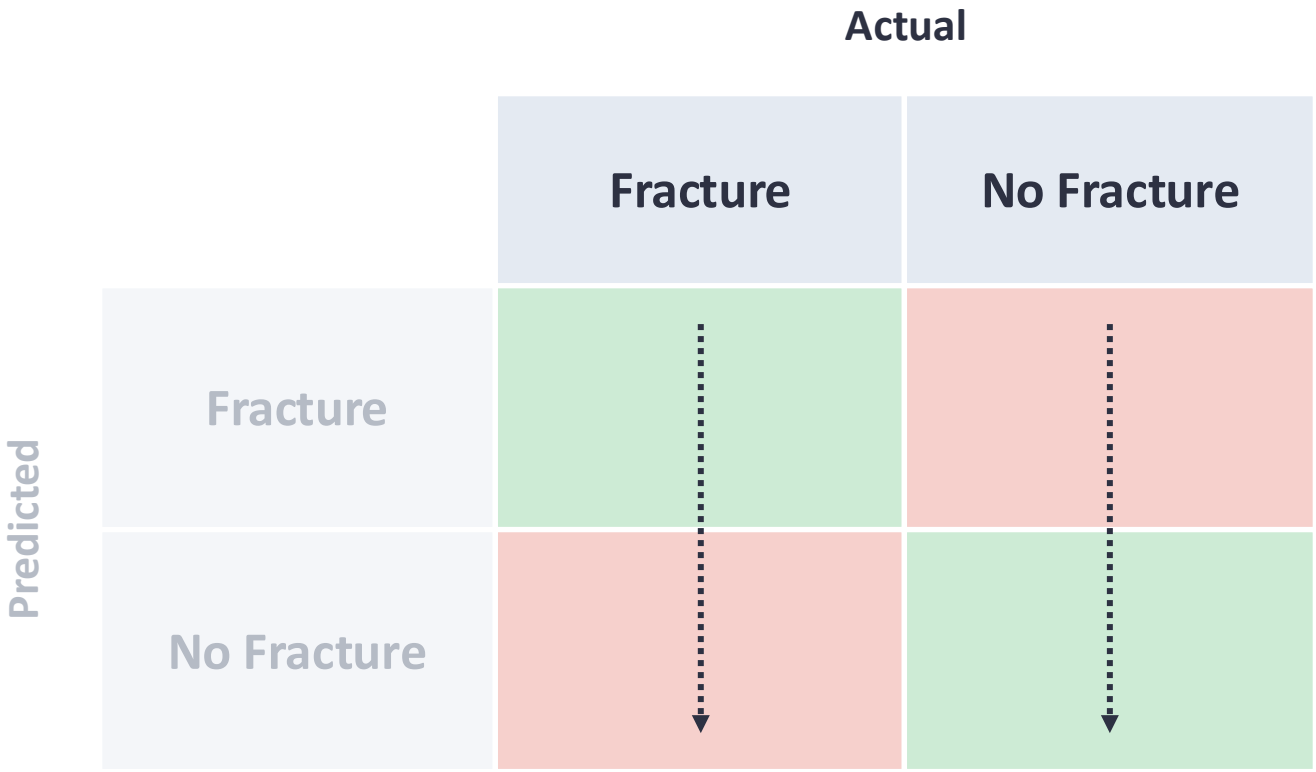
Reading along a row tells us what happened to every image the system labelled Fracture, or labelled No Fracture.



Columns show the known truth

Each column collects the X-rays that truly belong to one class, as confirmed by a radiologist.

Reading down a column tells us how the system treated every X-ray that really showed a fracture, or really did not.

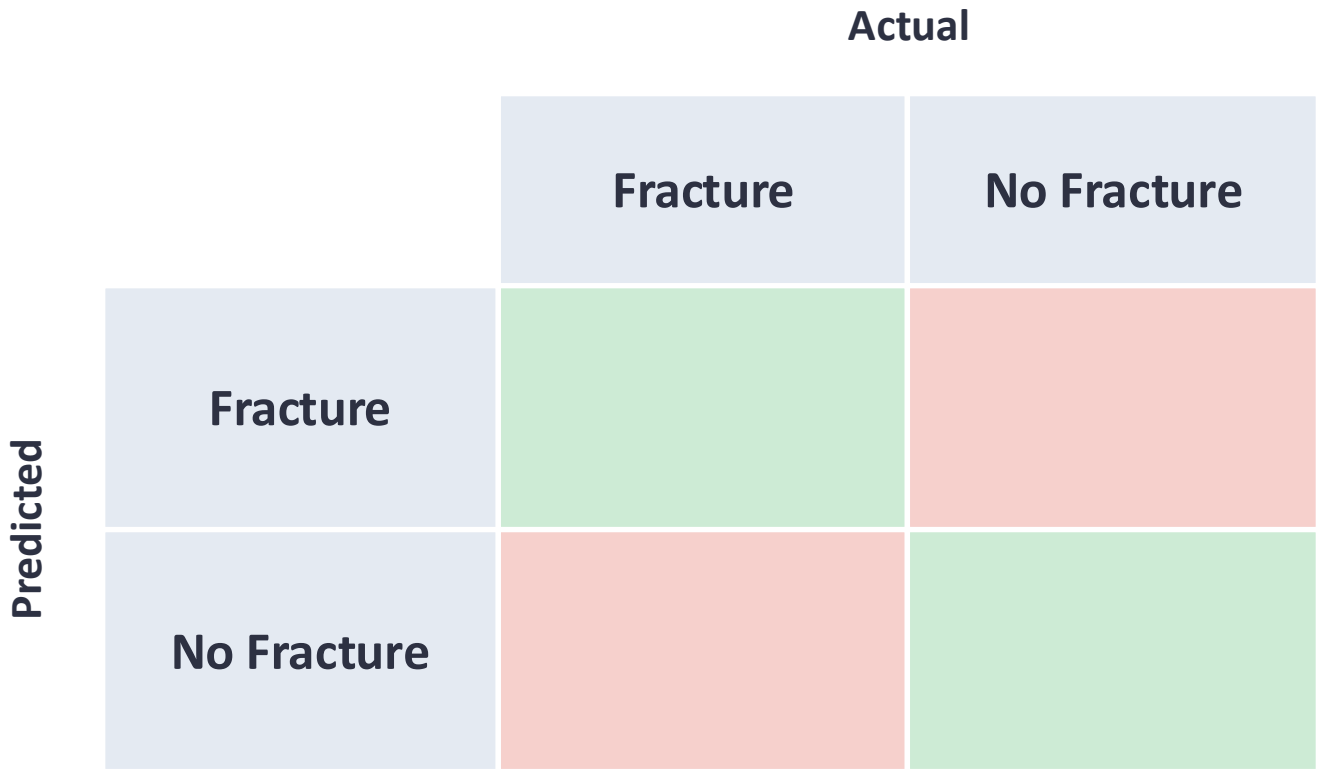


With two classes, the matrix has four cells

Every X-ray in the test set falls into exactly one cell.

Green cells on the diagonal are correct decisions.

Red cells off the diagonal are mistakes.

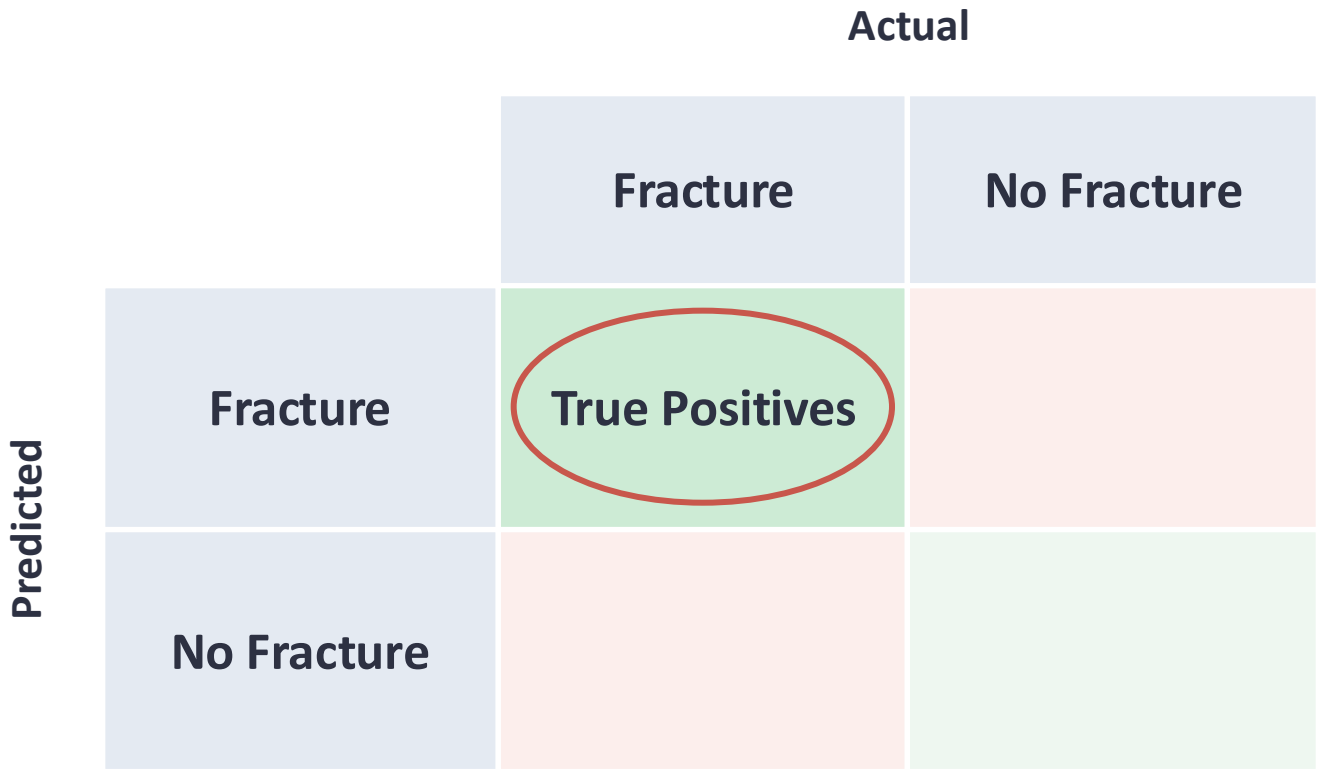


True Positives: top-left

True Positives

X-rays that really show a fracture, and that the AI system correctly flagged as Fracture.

Example: a broken wrist is flagged, and the patient is treated promptly.

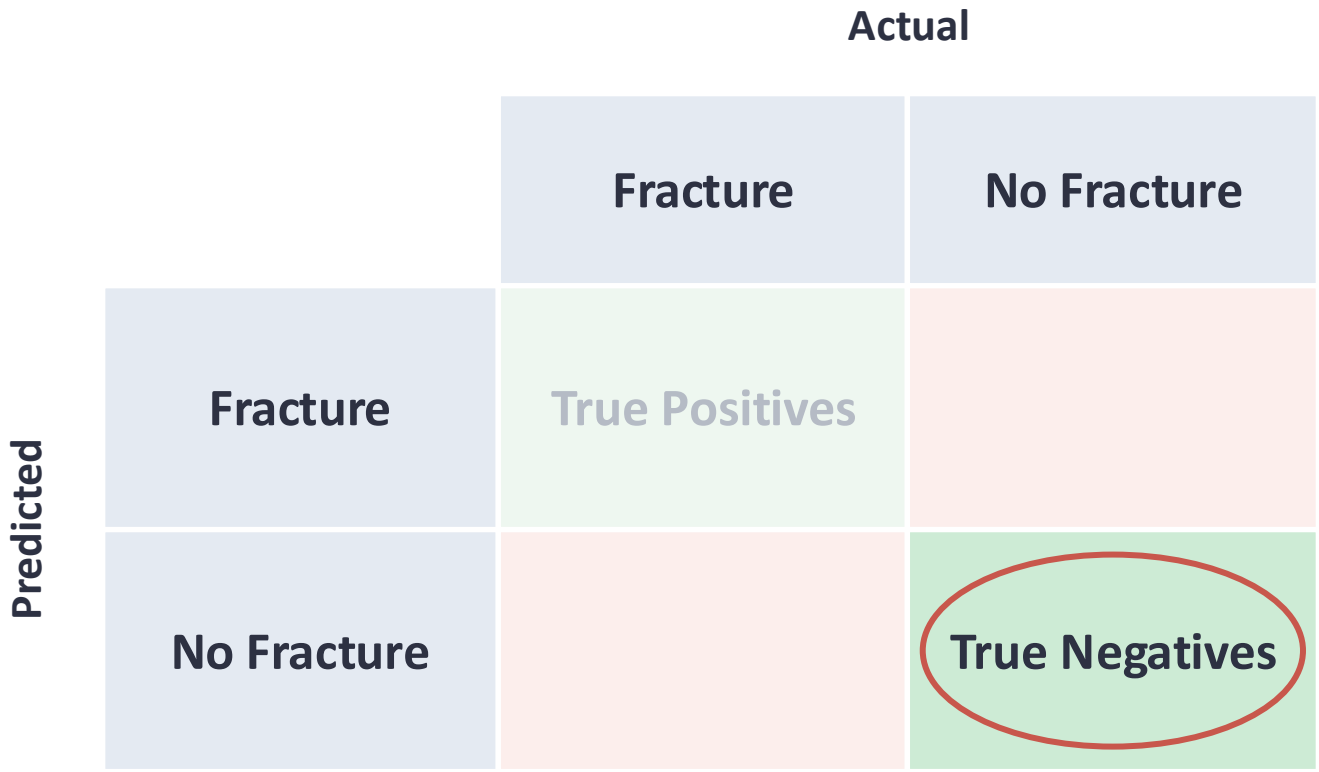


True Negatives: bottom-right

True Negatives

X-rays of intact bones that the AI system correctly cleared as No Fracture.

Example: a sprained ankle with no break is cleared, and the patient goes home.

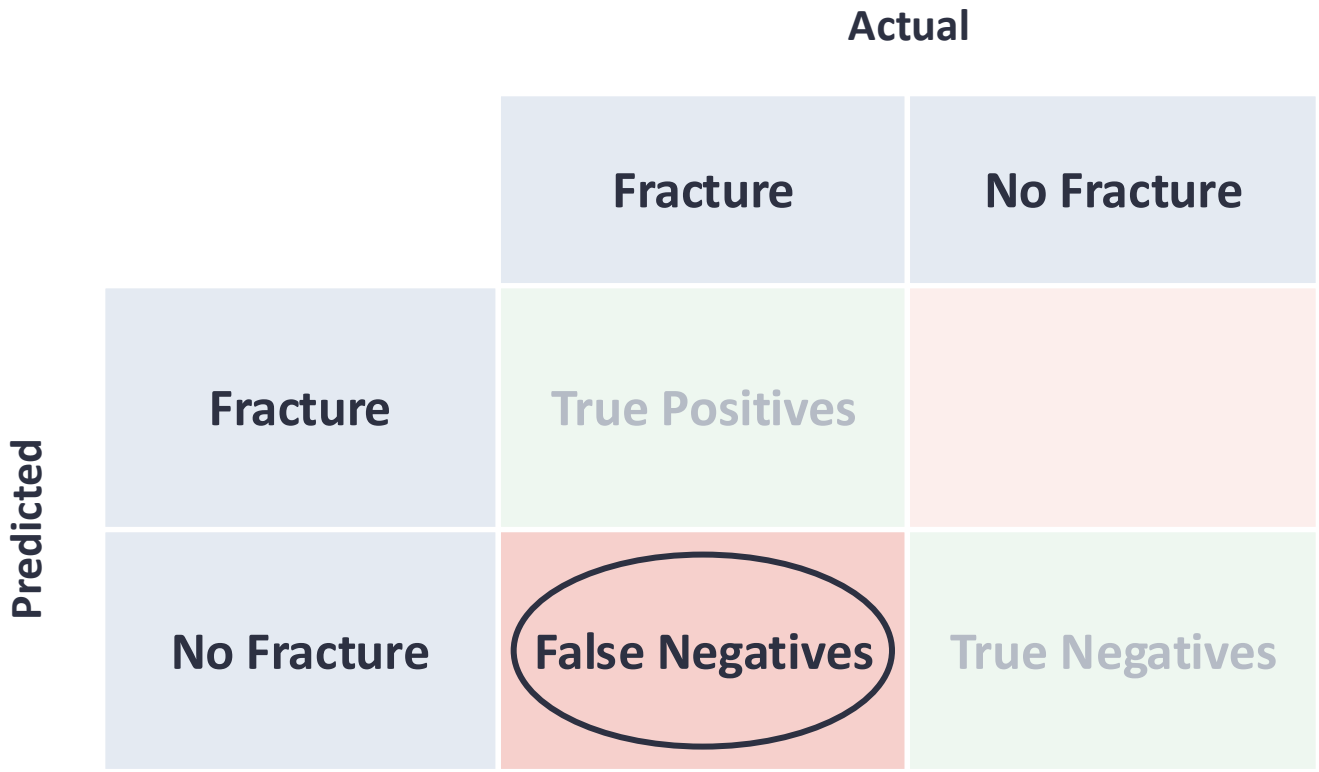


False Negatives: bottom-left

False Negatives

X-rays that really show a fracture, but that the AI system wrongly cleared as No Fracture.

Example: a hairline fracture is missed, and the patient is sent home untreated.



False Positives: top-right

False Positives

X-rays of intact bones that the AI system wrongly flagged as Fracture.

Example: a healthy bone is flagged, and the patient waits for an unnecessary review.

		Actual: Fracture	Actual: No Fracture
Predicted	Fracture	True Positives	False Positives
Predicted	No Fracture	False Negatives	True Negatives

The complete confusion matrix

Once the four cells are counted, we can compute two useful rates:

Sensitivity
how well we detect fractures

Specificity
how well we clear intact bones

		Actual: Fracture	Actual: No Fracture
Predicted	Fracture	True Positives	False Positives
Predicted	No Fracture	False Negatives	True Negatives

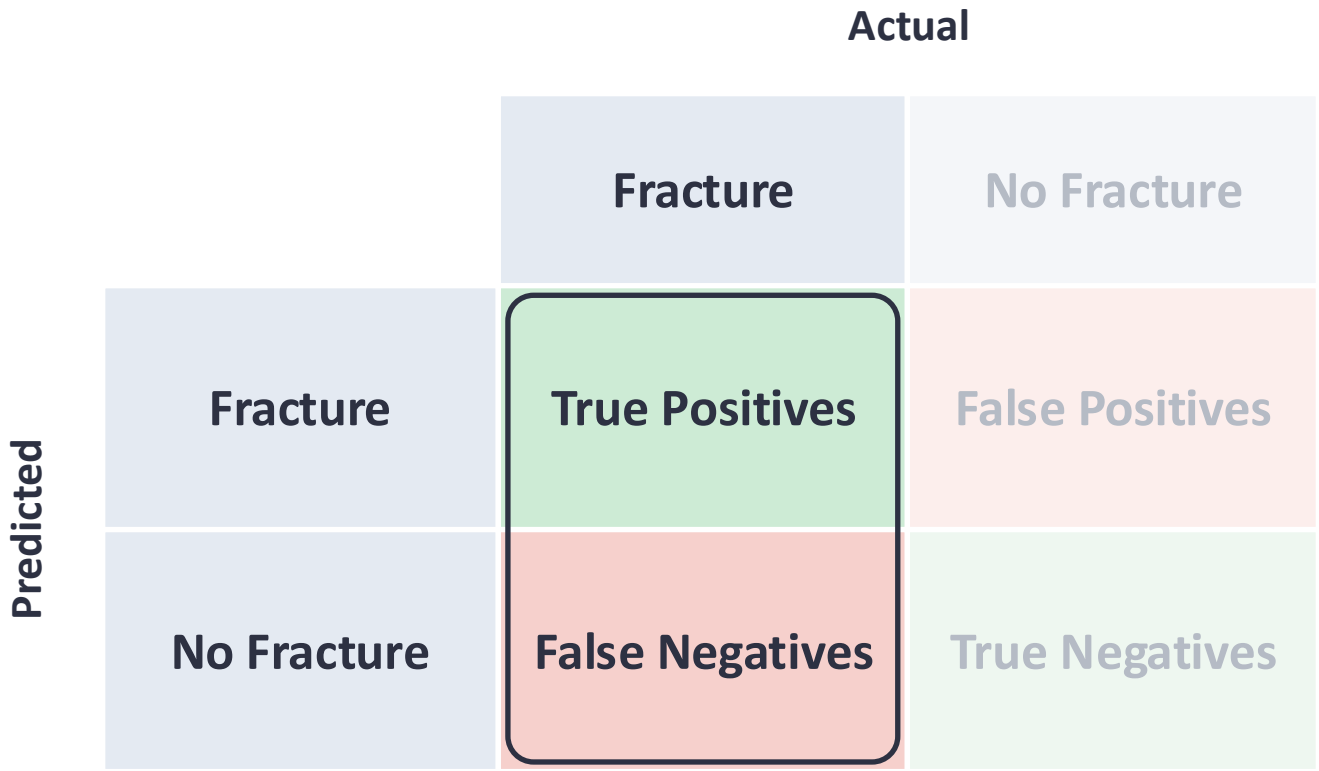
Sensitivity: how many fractures do we detect?

Of all the X-rays that really show a fracture, what proportion did the AI system correctly flag?

It uses only the Fracture column.

Also called **Recall** or the **true positive rate (TPR).**

$$\textbf{Sensitivity} = \frac{TP}{TP + FN}$$

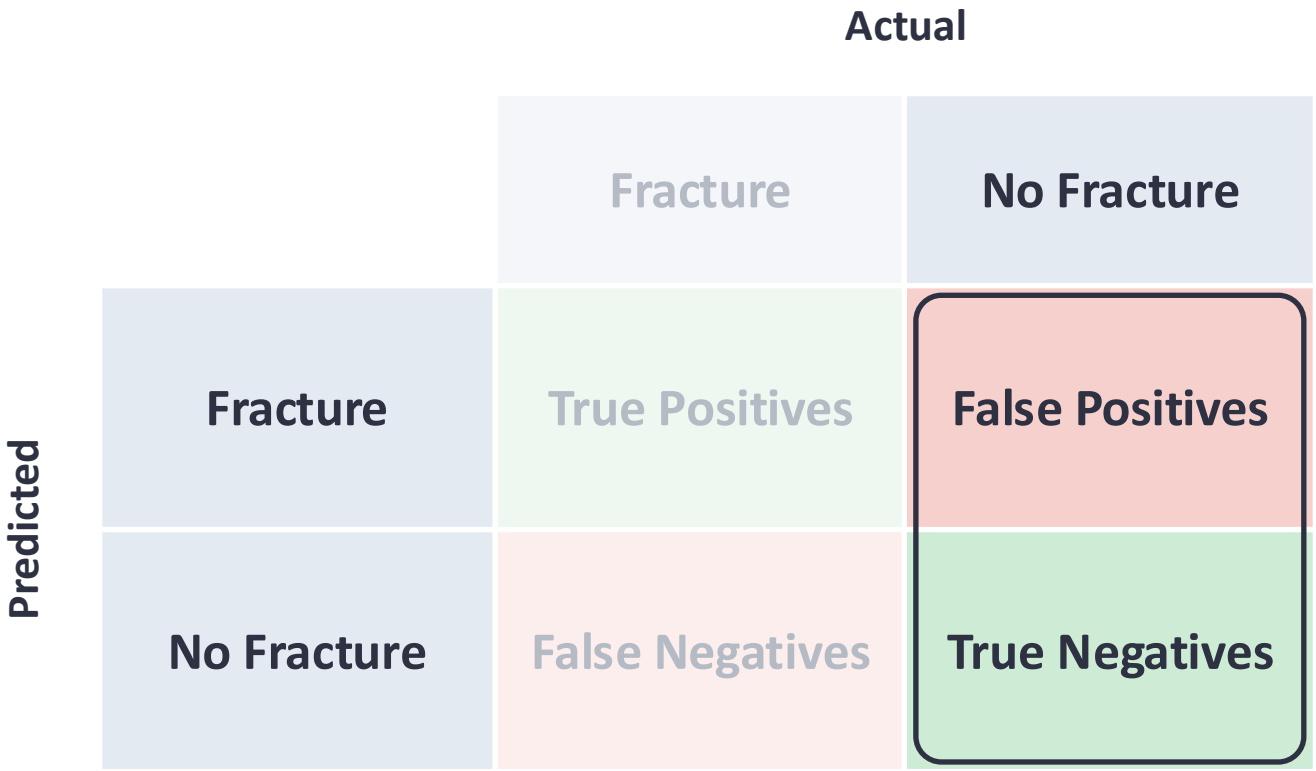


Specificity: how many intact bones do we clear?

Of all the X-rays of intact bones, what proportion did the AI system correctly clear?

It uses only the No Fracture column.

Also called the **true negative rate**.



$$\textbf{Specificity} = \frac{TN}{TN + FP}$$

Model A: support vector machine on the test set

Support vector machine (SVM)

We apply a support vector machine (SVM) classifier to a test set of **303 X-rays** and count the outcomes.

171 X-rays truly show a fracture (left column) and **132** show intact bones (right column).

		Actual	
		Fracture	**No Fracture**
Predicted	**Fracture**	**139**	**20**
	No Fracture	**32**	**112**

Model A: sensitivity

Support vector machine (SVM)

$$\text{Sensitivity} = \frac{139}{139 + 32} = 0.81$$

81% of the X-rays showing a fracture were correctly flagged by the SVM.

		Actual: Fracture	Actual: No Fracture
Predicted	Fracture	139	20
	No Fracture	32	112

Model A: specificity

Support vector machine (SVM)

$$\text{Specificity} = \frac{112}{112 + 20} = 0.85$$

85% of the X-rays of intact bones were correctly cleared by the SVM.

		Actual: Fracture	Actual: No Fracture
Predicted	Fracture	139	20
	No Fracture	32	112

Model B: random forest on the same test set

Random Forest

$$\text{Sensitivity} = \frac{142}{142 + 29} = 0.83$$

$$\text{Specificity} = \frac{110}{110 + 22} = 0.83$$

		Actual: Fracture	Actual: No Fracture
Predicted	Fracture	142	22
Predicted	No Fracture	29	110

Comparing the two models

Support vector machine

Sensitivity **0.81**

Specificity **0.85**

Random forest

Sensitivity **0.83**

Specificity **0.83**

Random forest is slightly better at detecting fractures (positives).
The SVM is slightly better at clearing intact bones (negatives).

?

Which model should we choose?

Choose the SVM

if avoiding unnecessary reviews matters most: higher specificity means fewer healthy patients flagged.

Choose random forest

if missing a fracture is the greater risk: higher sensitivity means fewer broken bones sent home untreated.

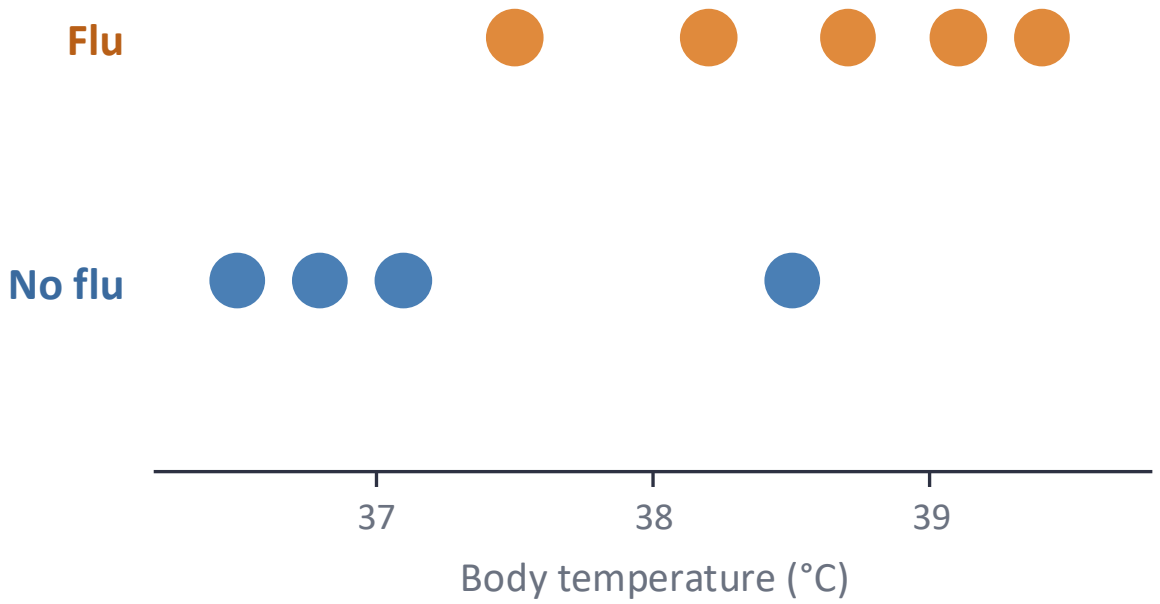
- The right choice depends on which mistake is more costly in the clinical setting.

ROC Curves

Choosing a classification threshold, illustrated with flu and body temperature

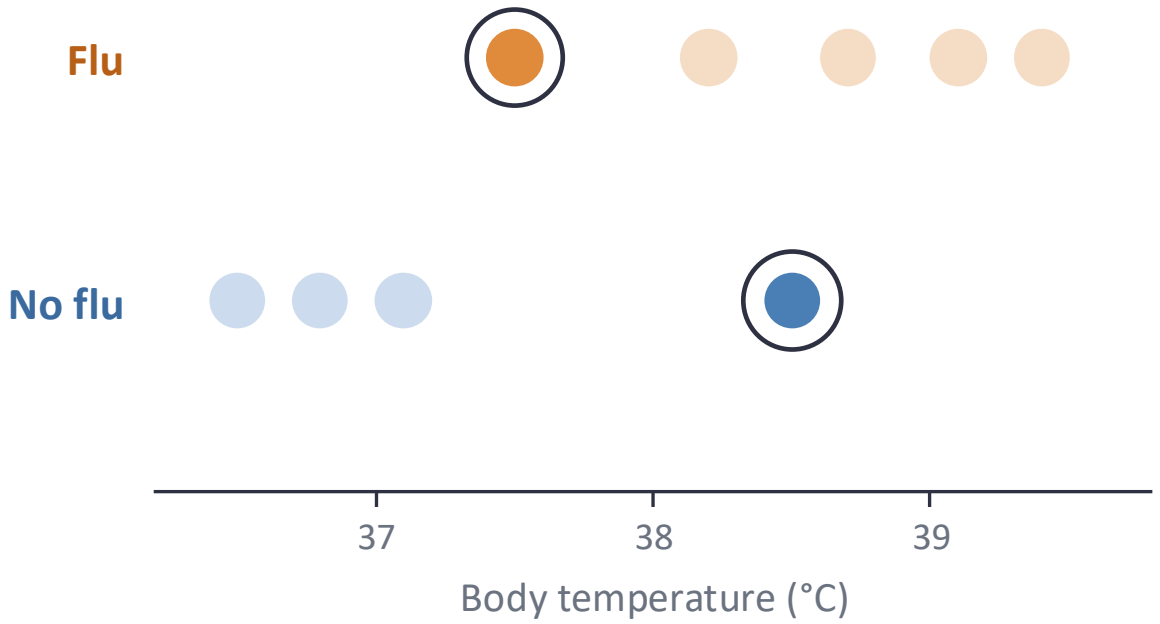
A new example: flu and body temperature

- We measure the body temperature of patients and know, from a laboratory test, whether each one had flu.
- **Flu patients** (top row) tend to run warmer than **patients without flu** (bottom row).
- **Flu is the positive class.**



Real data is not perfectly tidy

- **Flu without a high fever:** some people with flu have only a mild temperature, for example early in the illness.
- **Fever without flu:** a high temperature can have other causes, such as another infection.
- Temperature alone cannot separate the two groups perfectly.

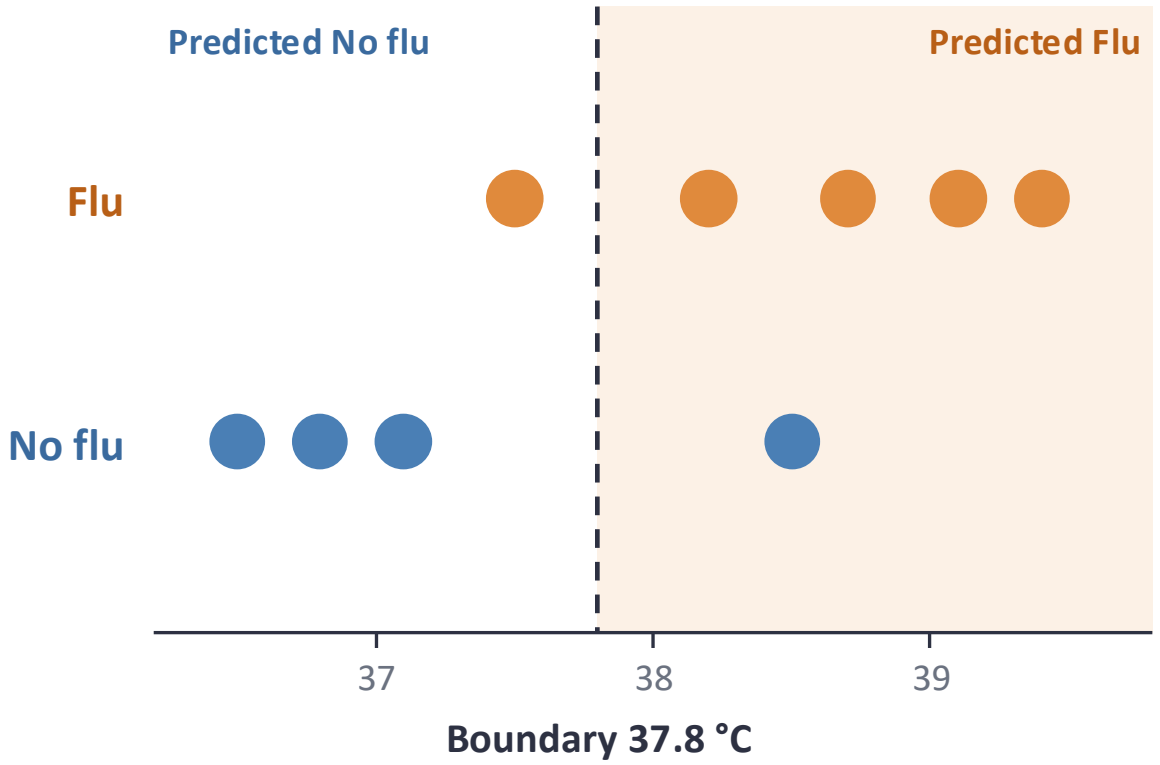


Fitting a support vector machine

An **SVM** learns a **decision boundary** that separates the two classes as widely as possible.

With one feature, the boundary is simply a cut-off temperature: here, 37.8 °C.

It tolerates a few mistakes, such as the two unusual patients.

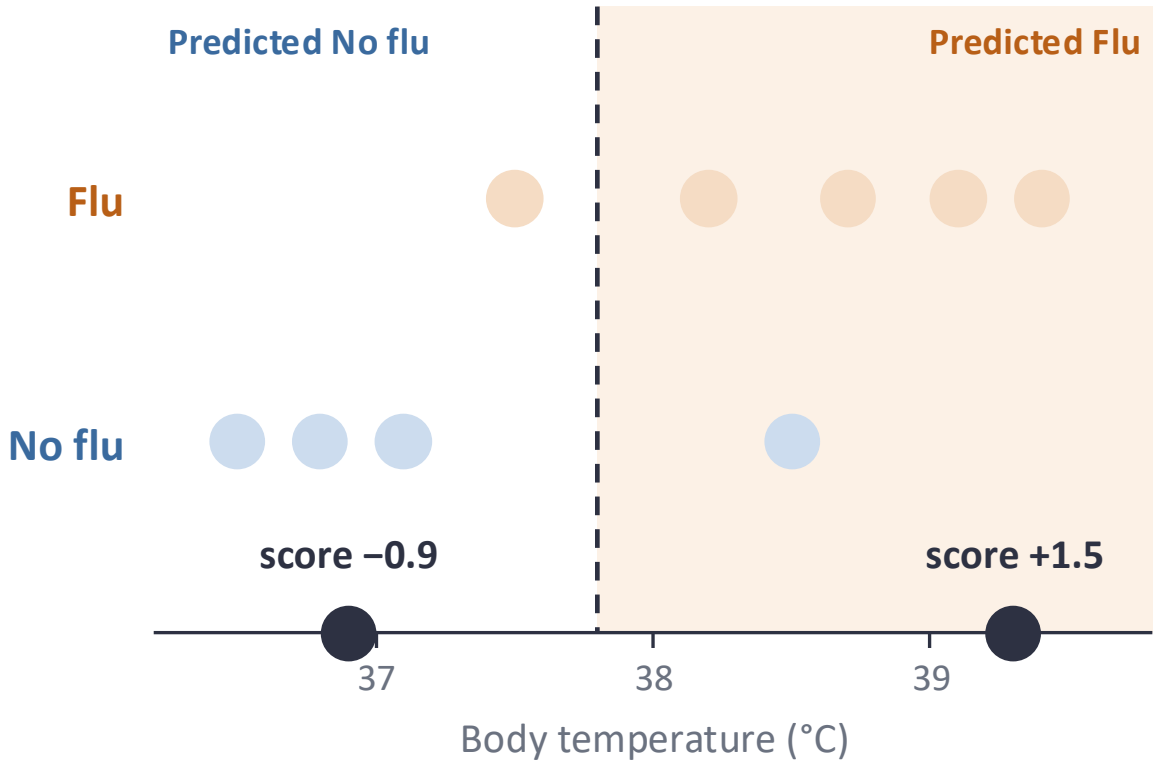


The SVM gives every patient a score

For a new patient, the SVM computes a **score**: how far, and on which side of the boundary, the patient lies?

A patient at 39.3 °C gets a large positive score: very likely flu.

A patient at 36.9 °C gets a negative score: very likely no flu.

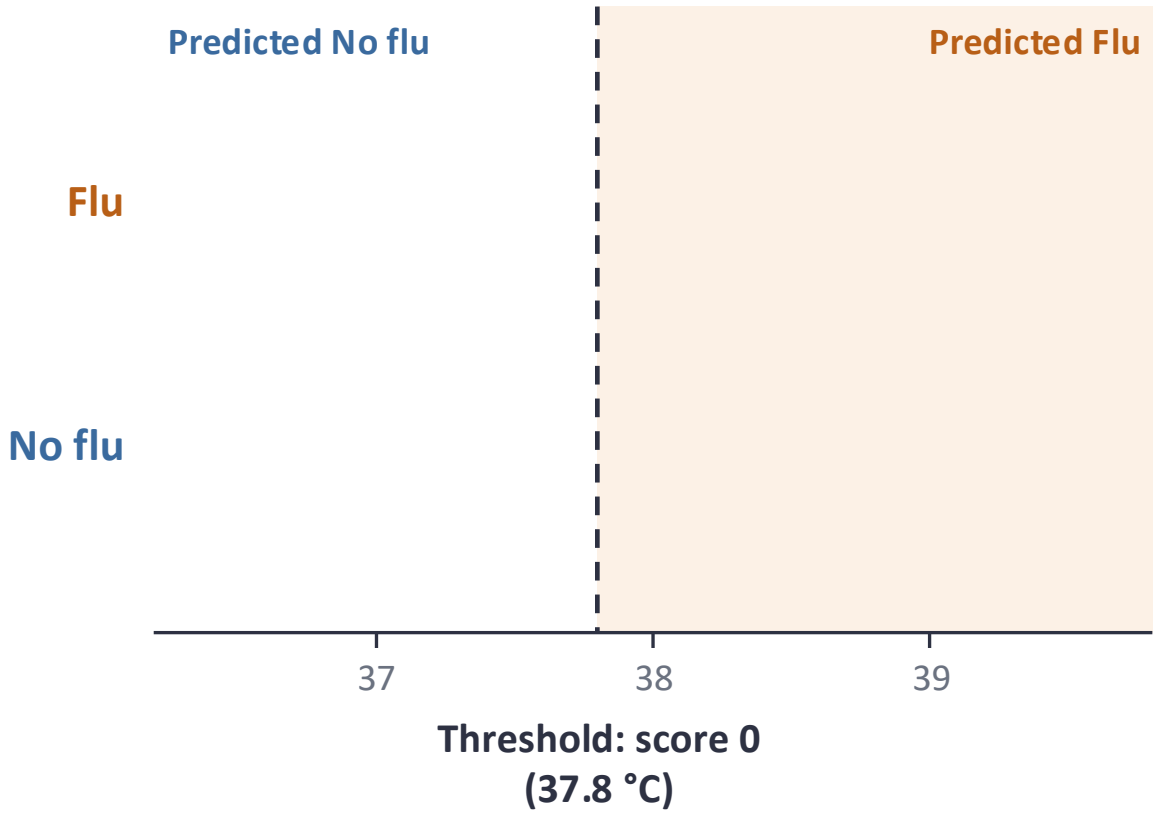


From scores to decisions: the threshold

To turn scores into decisions, we need a **threshold**.

By default: predict **Flu** if the score is above 0, otherwise **No flu**.

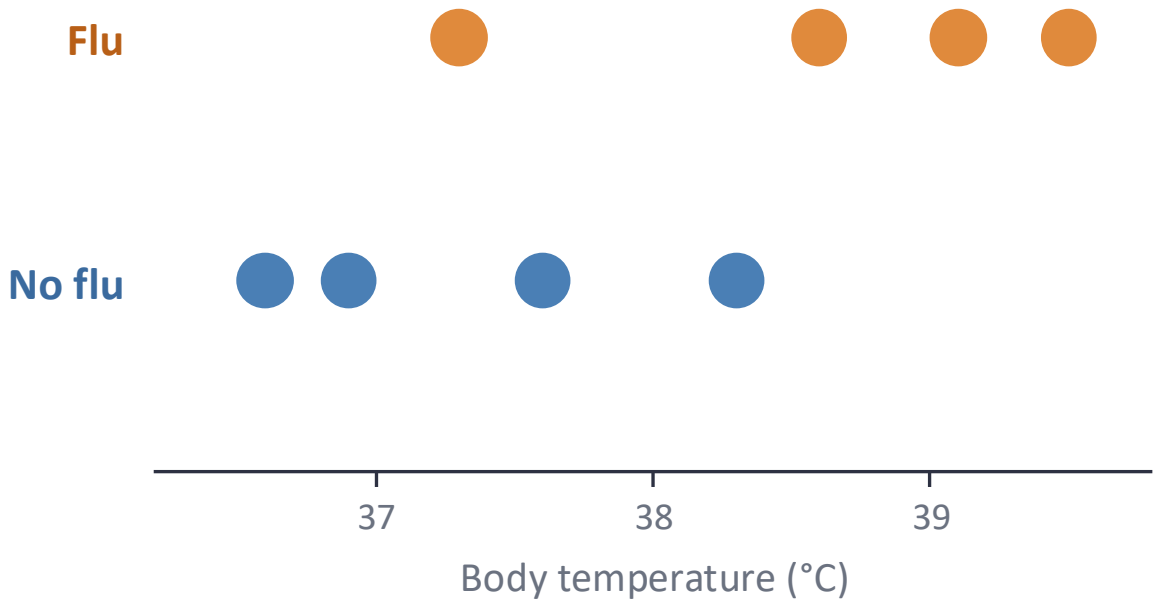
Here each threshold matches a cut-off temperature, so we can picture it as a line we slide left or right.



Testing on new patients

To evaluate the classifier we use **eight new patients** whose diagnosis we already know:

four with flu
four without flu

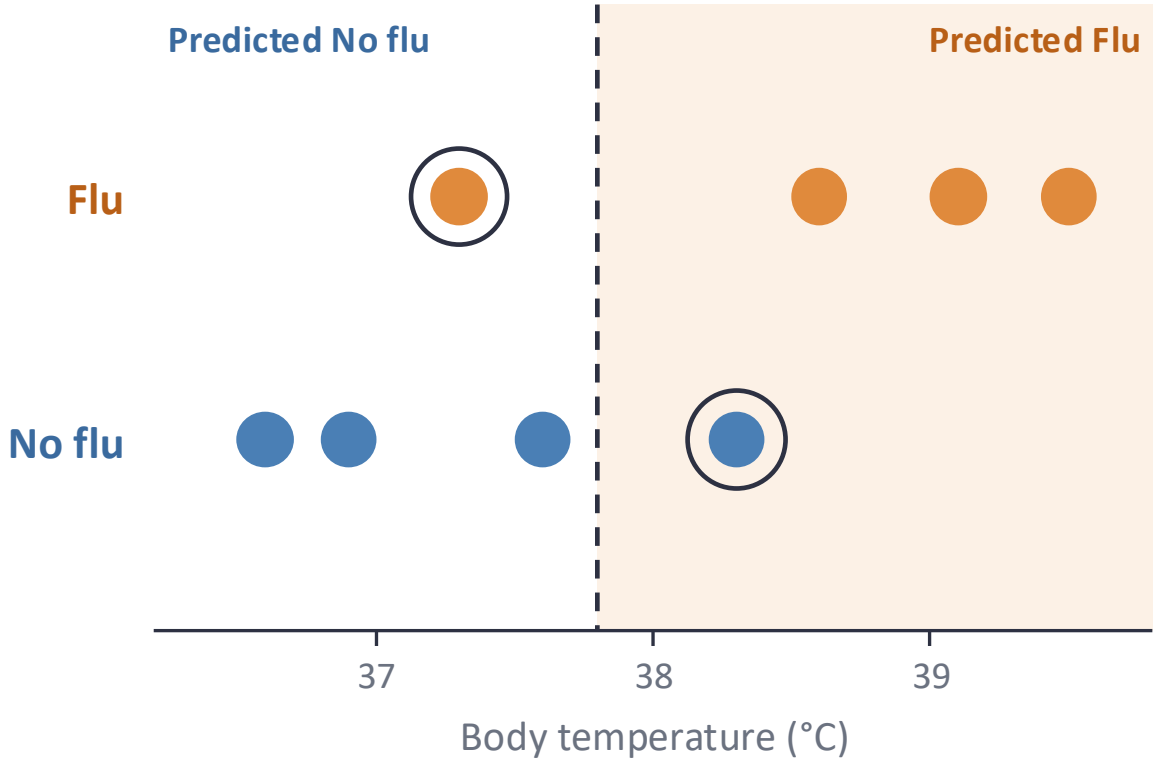


Classifying the test patients

With the default threshold, six of the eight patients are classified correctly.

Missed: a flu patient at only 37.3 °C (false negative).

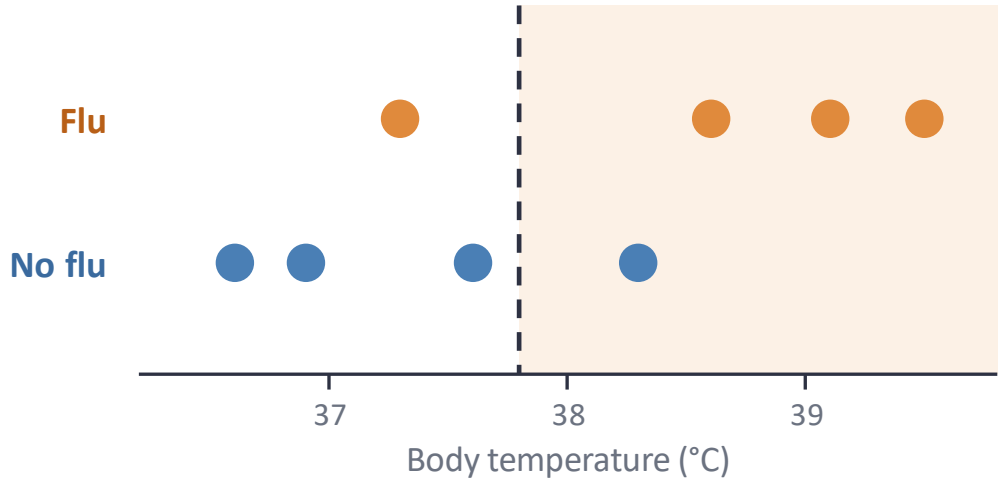
False alarm: a patient at 38.3 °C who does not have flu (false positive).



Summarising the results in a confusion matrix

With the default threshold we get one confusion matrix, and from it sensitivity and specificity.

But the threshold is our choice. What happens if we move it?



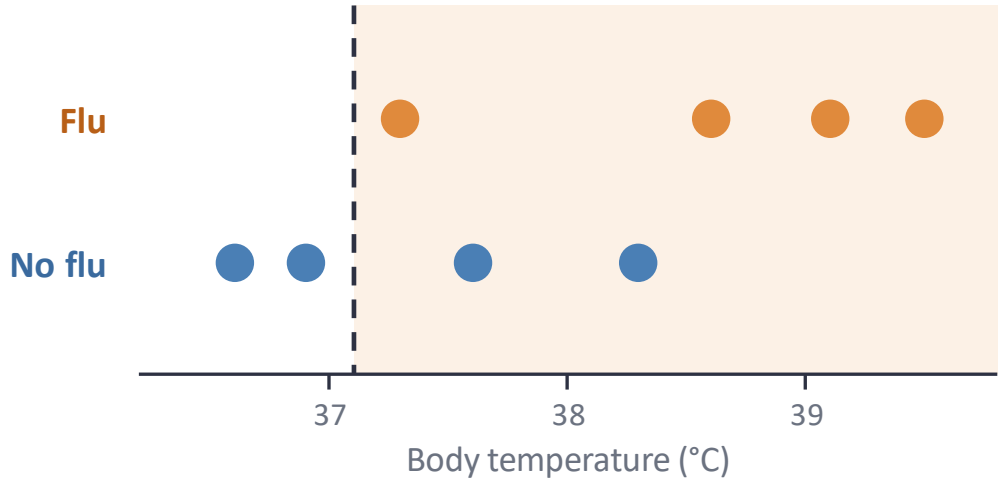
		Actual: Flu	Actual: No flu
Predicted	Flu	3	1
Predicted	No flu	1	3

Sensitivity = 3/4 = 0.75
Specificity = 3/4 = 0.75

Lowering the threshold

If missing a flu case is costly, for example during an outbreak in a care home, we can **lower the threshold**.

Every flu patient is now caught, but there are more false alarms.



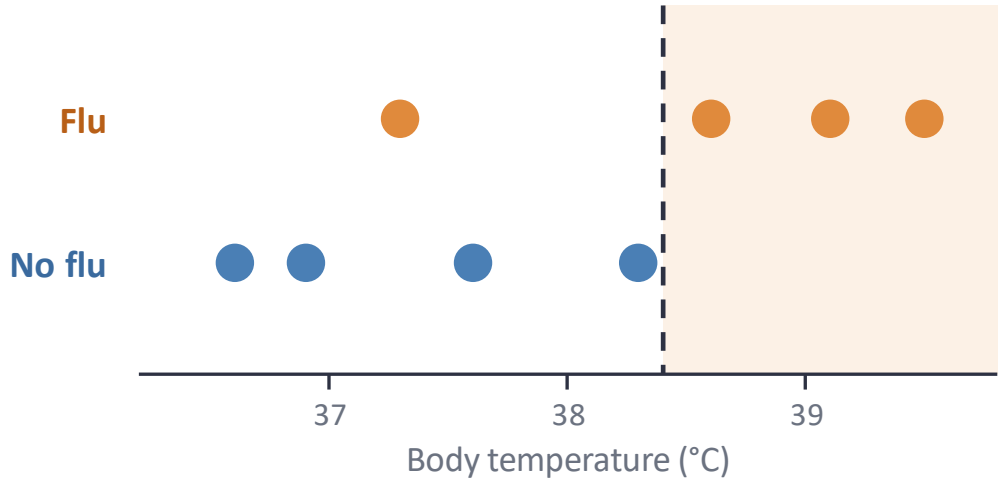
Predicted \ Actual	Flu	No flu
Flu	4	2
No flu	0	2

Sensitivity = 4/4 = 1.00
Specificity = 2/4 = 0.50

Raising the threshold

If false alarms are costly, for example unnecessary medication or time off work, we can **raise the threshold**.

No false alarms now, but one flu case is still missed.

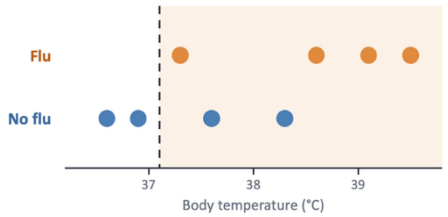


Predicted \ Actual	Flu	No flu
Flu	3	0
No flu	1	4

Sensitivity = 3/4 = 0.75
Specificity = 4/4 = 1.00

Too many confusion matrices

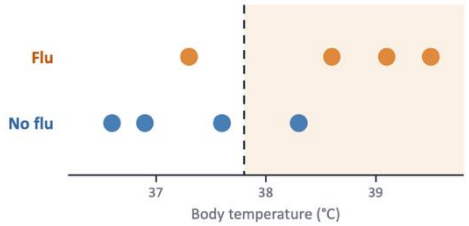
- Every threshold gives a different confusion matrix.
- Thresholds between the same two patients give identical results, but even so, comparing many tables side by side quickly becomes confusing.



Cut-off 37.1 °C

	Flu	No flu
Flu	4	2
No flu	0	2

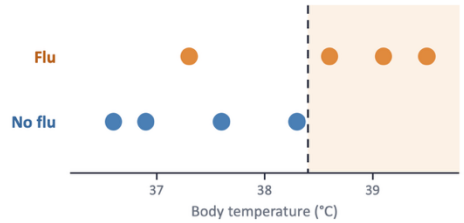
…



Cut-off 37.8 °C

	Flu	No flu
Flu	3	1
No flu	1	3

…



Cut-off 38.4 °C

	Flu	No flu
Flu	3	0
No flu	1	4

… …

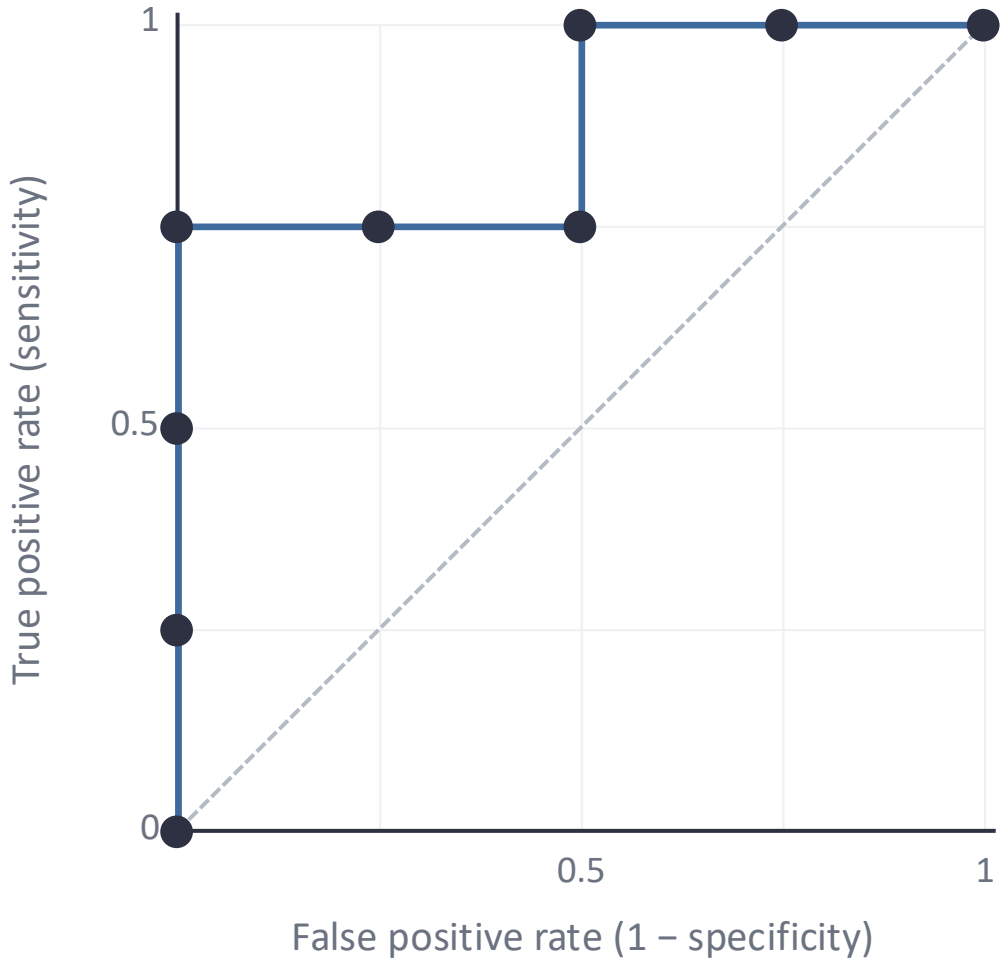
…and many more.

ROC curves summarise every threshold

A **Receiver Operating Characteristic (ROC)** curve replaces the pile of tables with one picture.

Each point is one threshold.

The y-axis shows how many flu cases we catch; the x-axis how many healthy patients we wrongly flag.

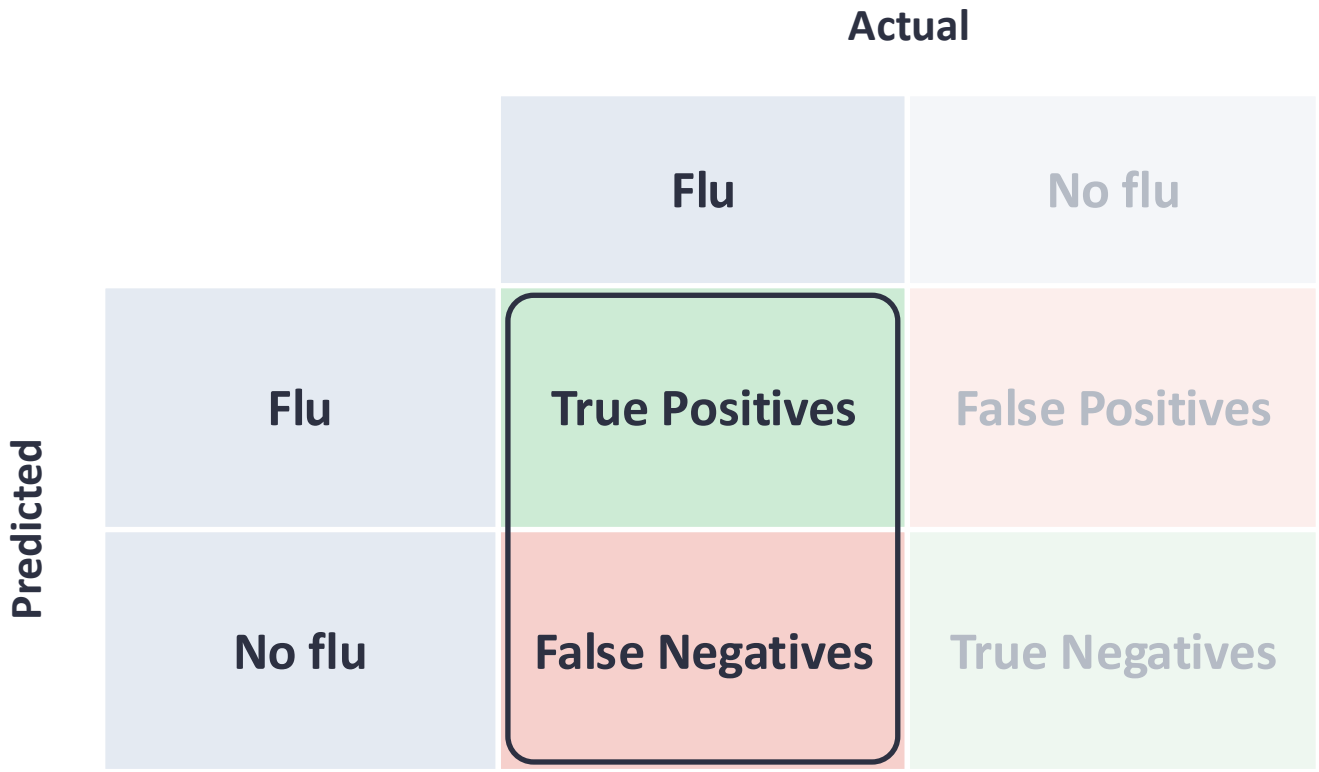


The y-axis: true positive rate

The true positive rate is the same as sensitivity: the proportion of patients with flu who are correctly flagged.

It uses only the Flu column.

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



The x-axis: false positive rate

The false positive rate is 1 − specificity: the proportion of patients without flu who are wrongly flagged.

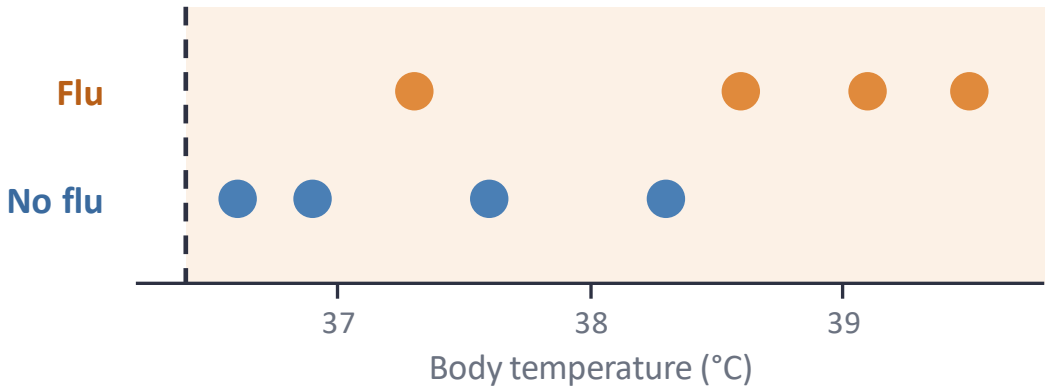
It uses only the No flu column.

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

		Actual: Flu	Actual: No flu
Predicted	Flu	True Positives	False Positives
Predicted	No flu	False Negatives	True Negatives

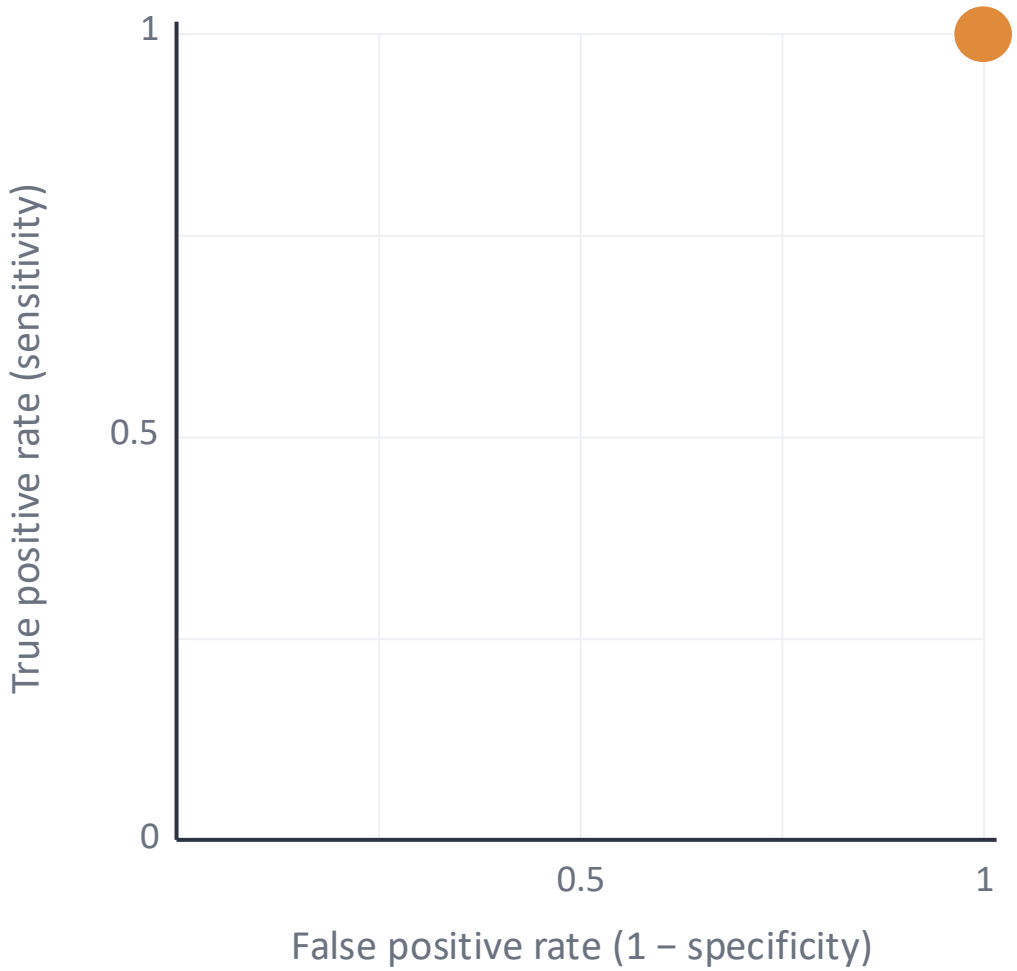
Building the ROC curve: step 1

- Start with the lowest cut-off: every patient is predicted to have flu.
- All flu cases caught, but every healthy patient is a false alarm.



	Flu	No flu
Flu	4	4
No flu	0	0

TPR = 4/4 = 1.00
FPR = 4/4 = 1.00



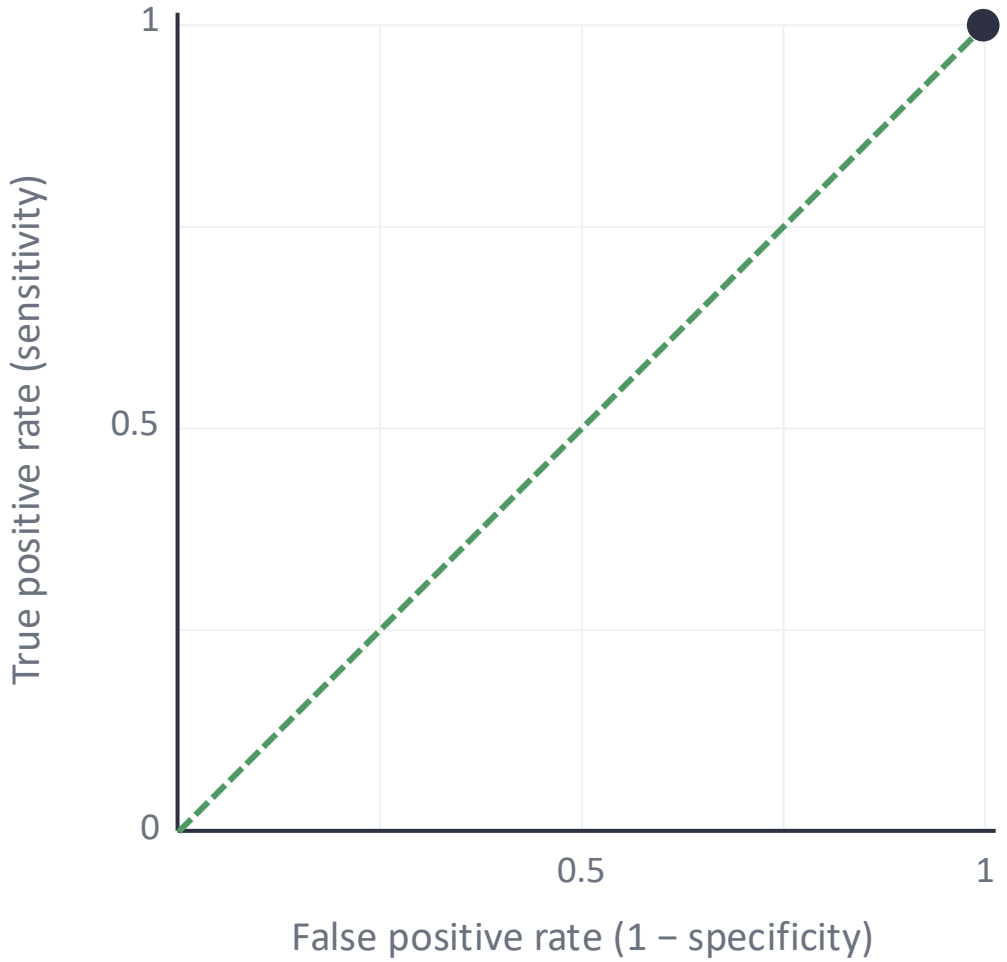
The diagonal line: no better than chance

The point (1, 1) catches every flu case but flags every healthy patient too.

On the **diagonal**, the true positive rate equals the false positive rate: the classifier is **no better than chance**.

Any point on this line means that the proportion of correctly classified patients with flu is the same as the proportion of incorrectly classified samples that have no flu.

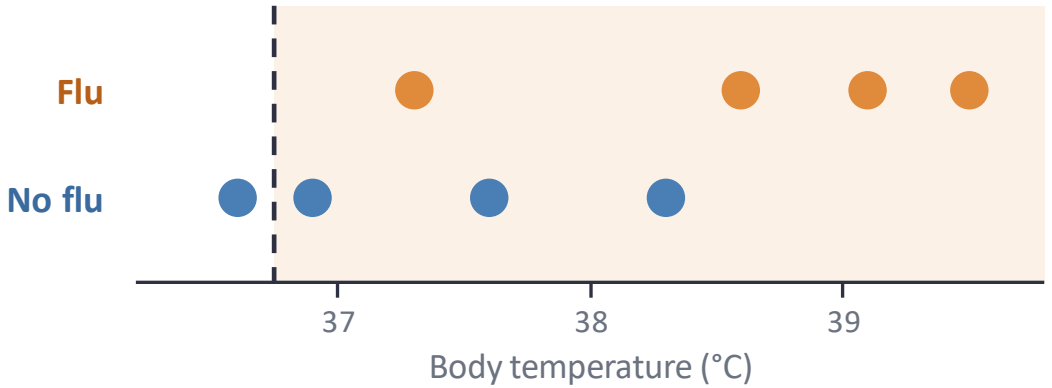
Good thresholds sit above it, towards the top-left.



Building the ROC curve: step 2

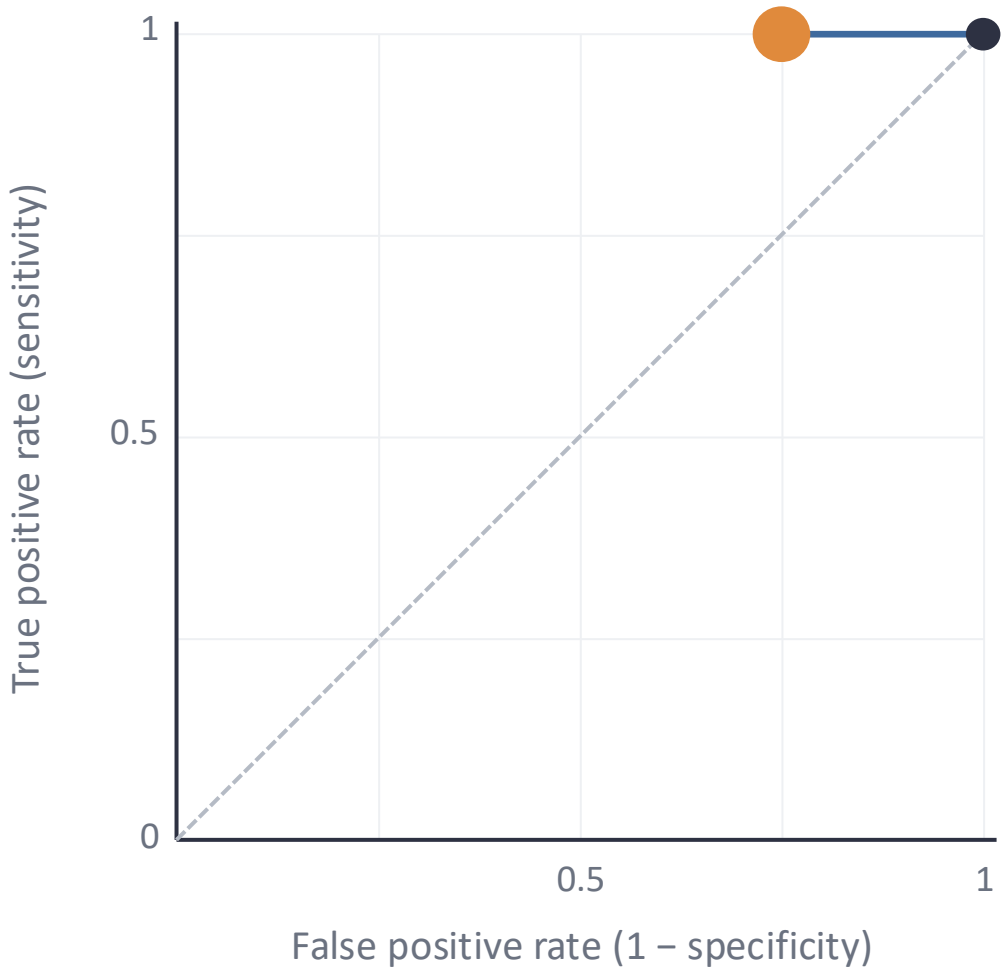
Move the cut-off past the coolest patient, who has no flu.

One fewer false alarm, no flu case lost: the point moves left.



	Flu	No flu
Flu	4	3
No flu	0	1

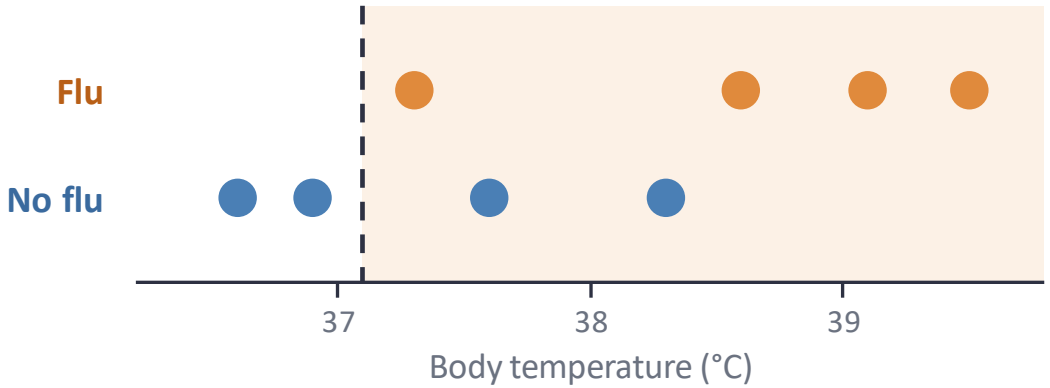
TPR = 4/4 = 1.00
FPR = 3/4 = 0.75



Building the ROC curve: step 3

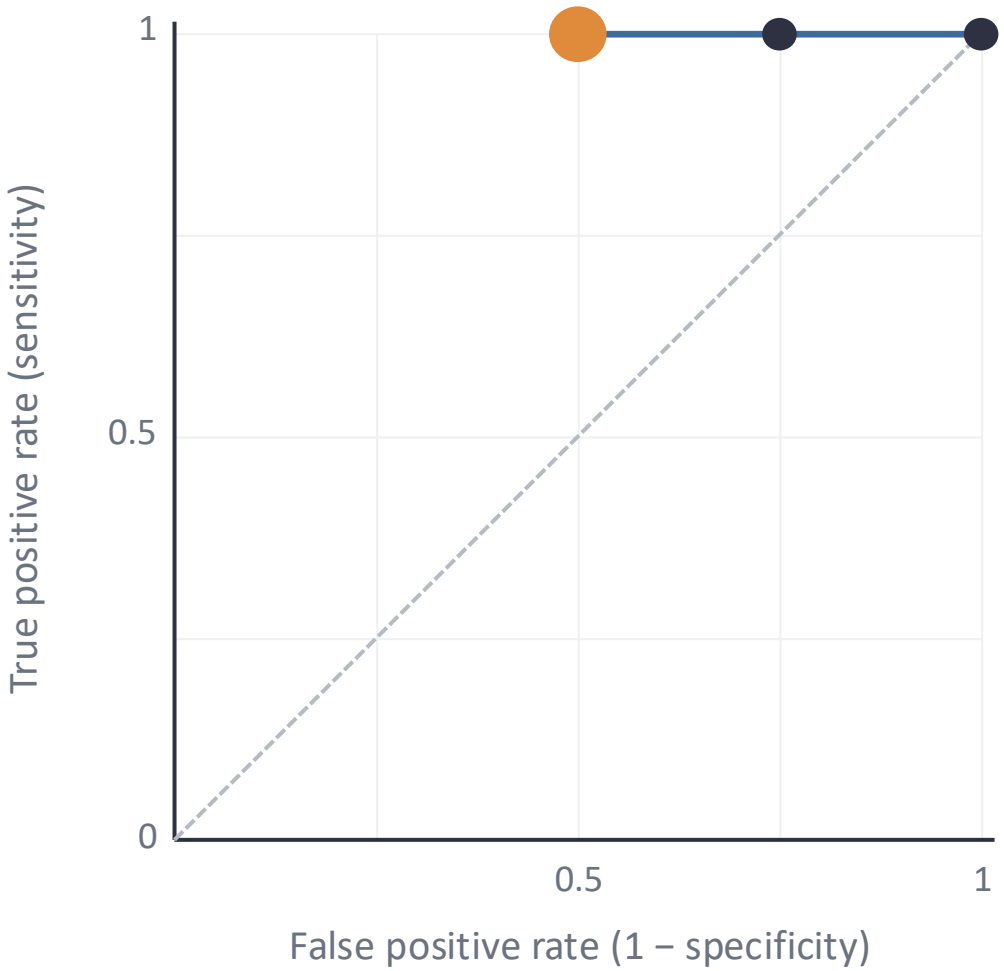
Another healthy patient is cleared correctly.
Still every flu case caught: the point moves further left.

Best one so far!



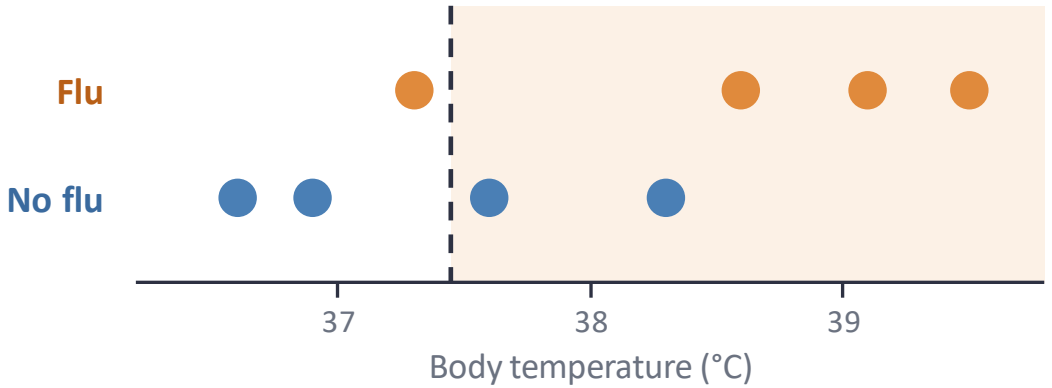
	Flu	No flu
Flu	4	2
No flu	0	2

TPR = 4/4 = 1.00
FPR = 2/4 = 0.50



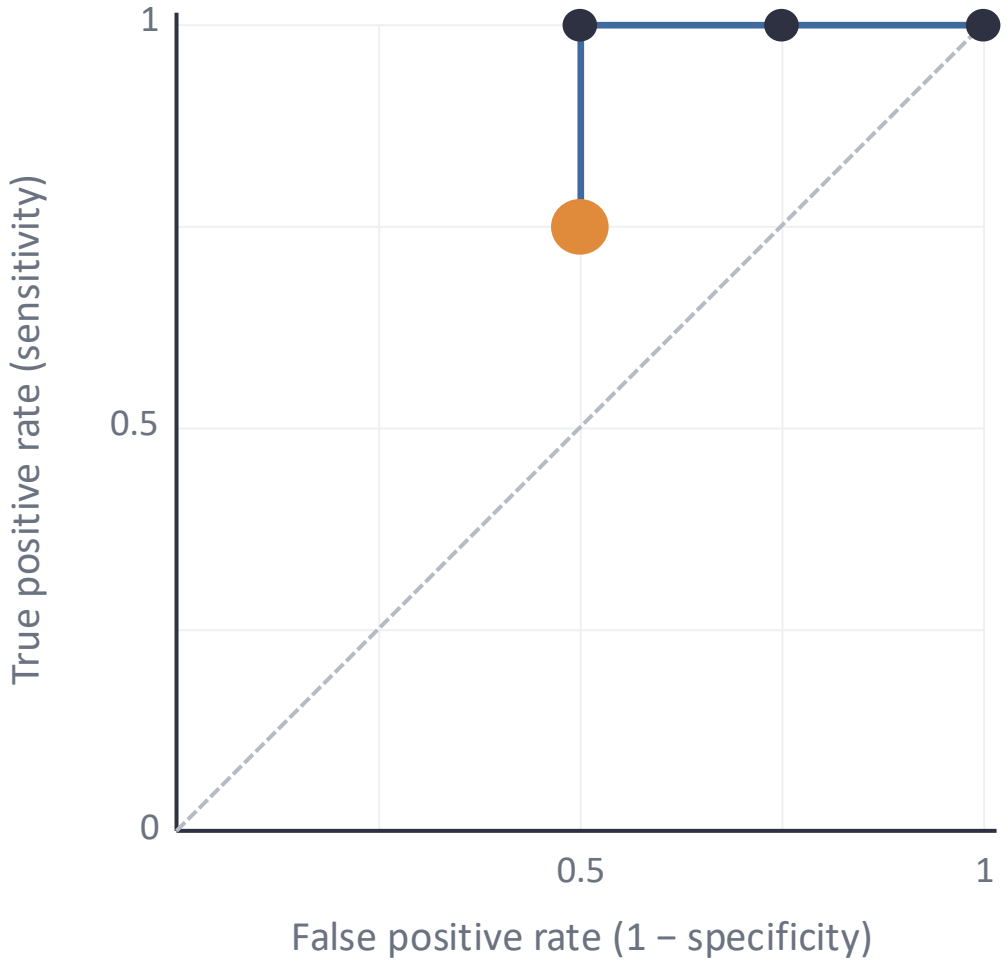
Building the ROC curve: step 4

Now the flu patient with only 37.3 °C is missed.
The true positive rate drops: the point moves down.



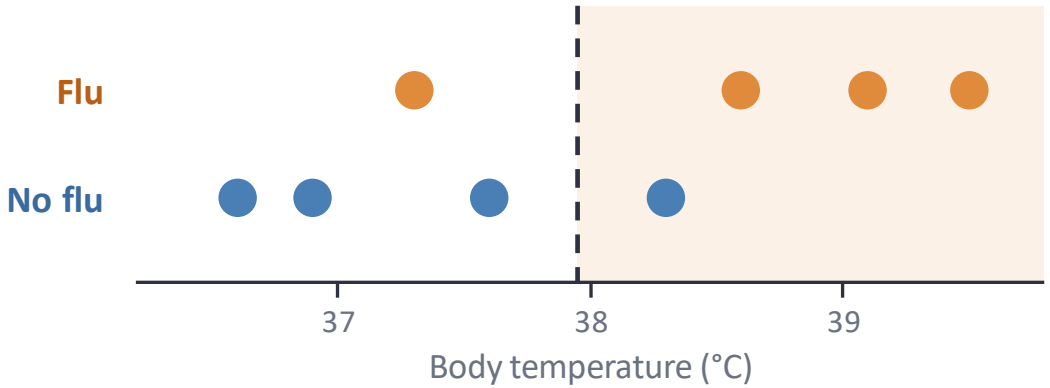
	Flu	No flu
Flu	3	2
No flu	1	2

TPR = 3/4 = 0.75
FPR = 2/4 = 0.50



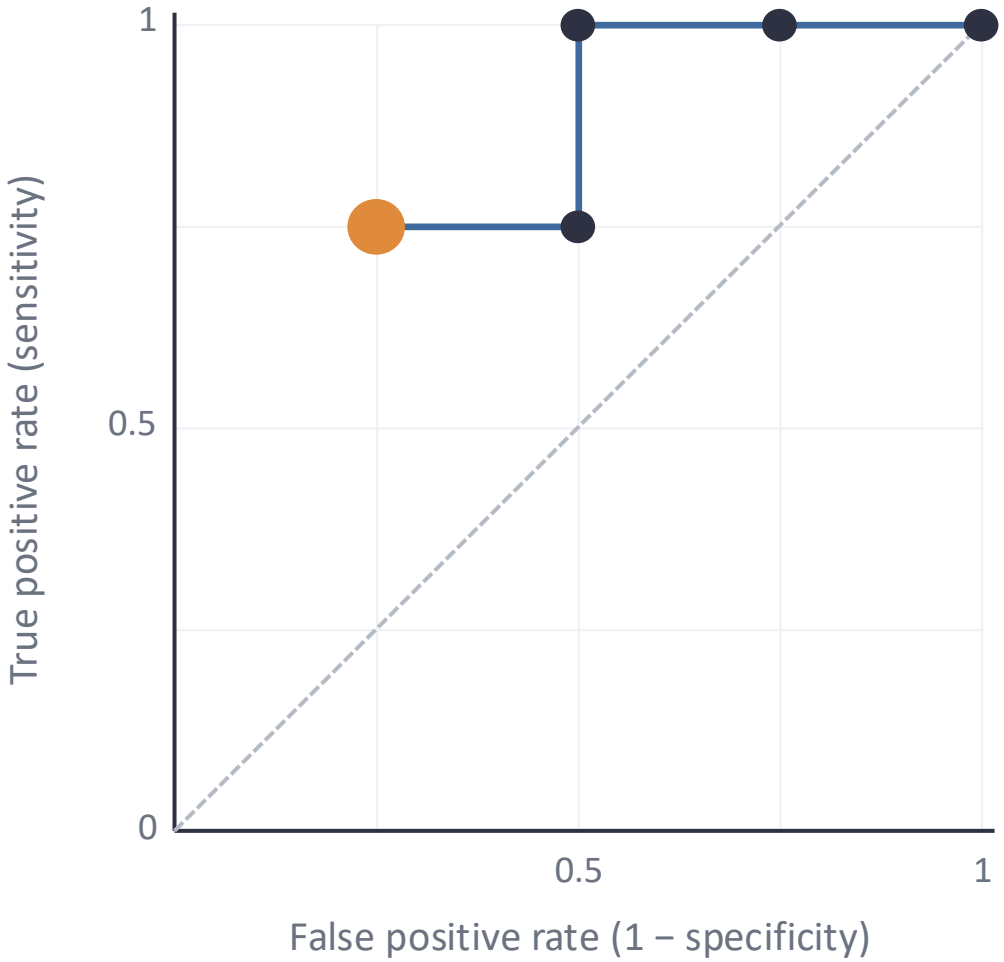
Building the ROC curve: step 5

The healthy patient at 37.6 °C is cleared.
This is the same result as the SVM's default threshold.



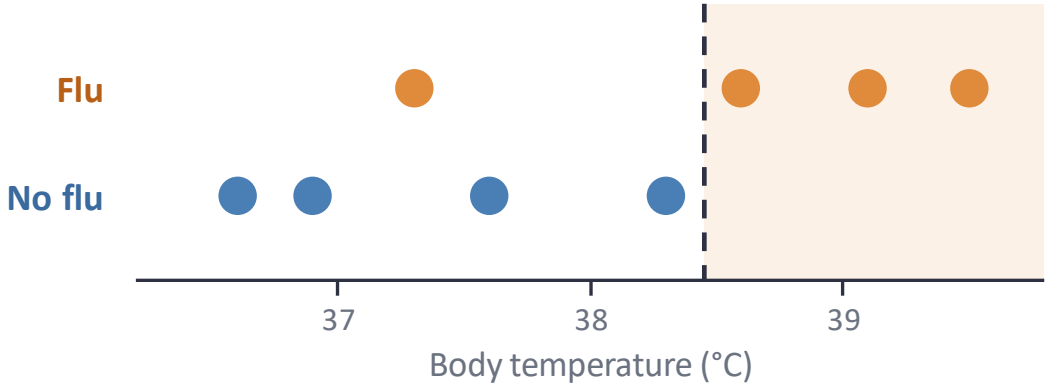
	Flu	No flu
Flu	3	1
No flu	1	3

TPR = 3/4 = 0.75
FPR = 1/4 = 0.25



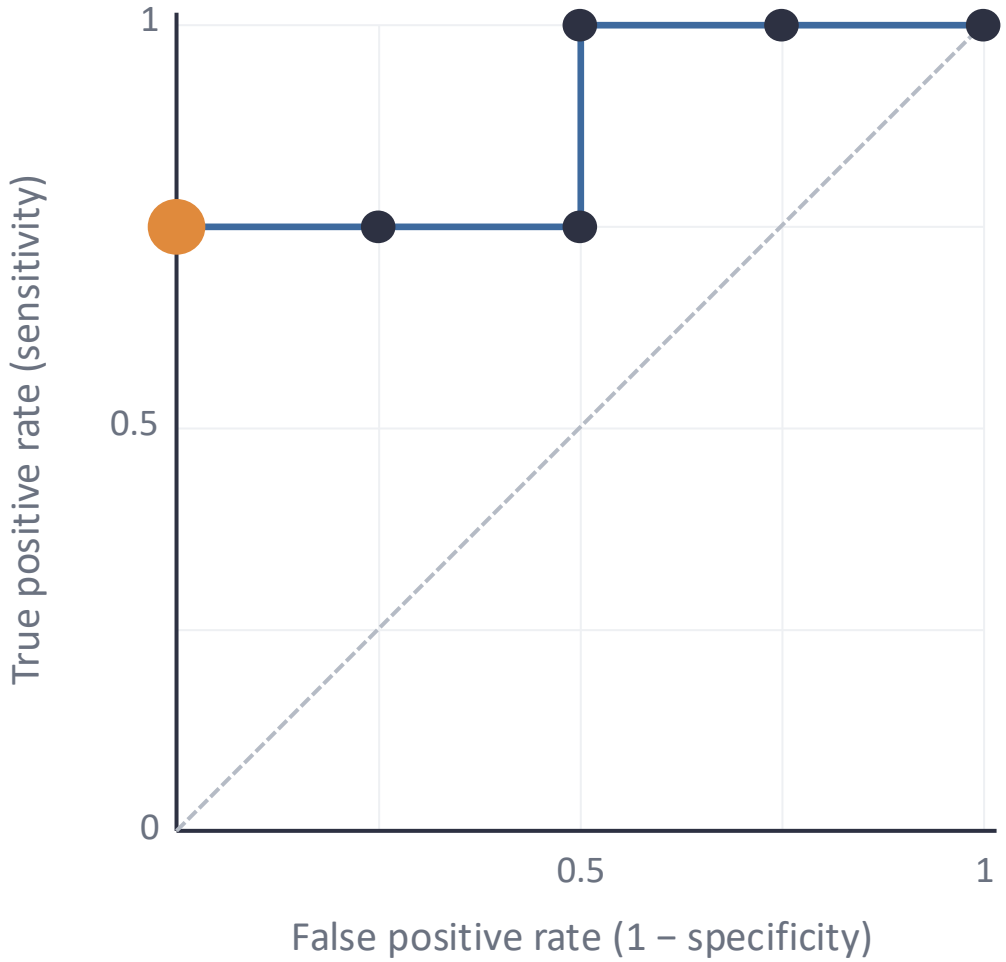
Building the ROC curve: step 6

The healthy patient with a fever is cleared: no false alarms at all.
Three of the four flu cases are still caught.



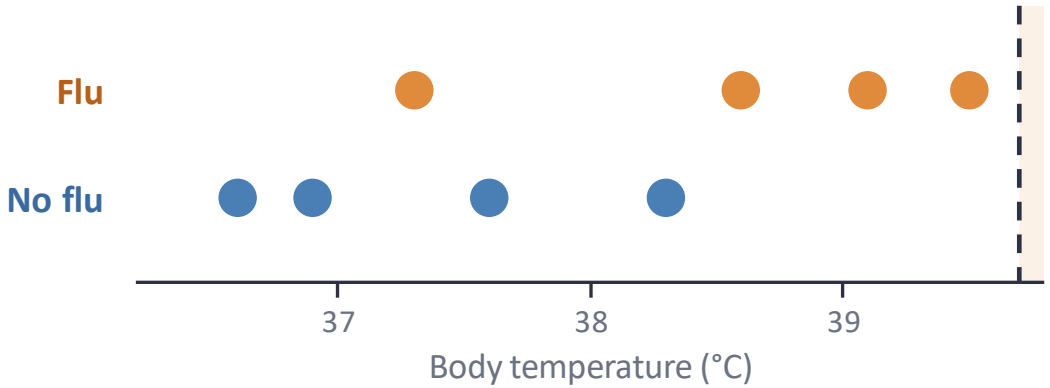
	Flu	No flu
Flu	3	0
No flu	1	4

TPR = 3/4 = 0.75
FPR = 0/4 = 0.00



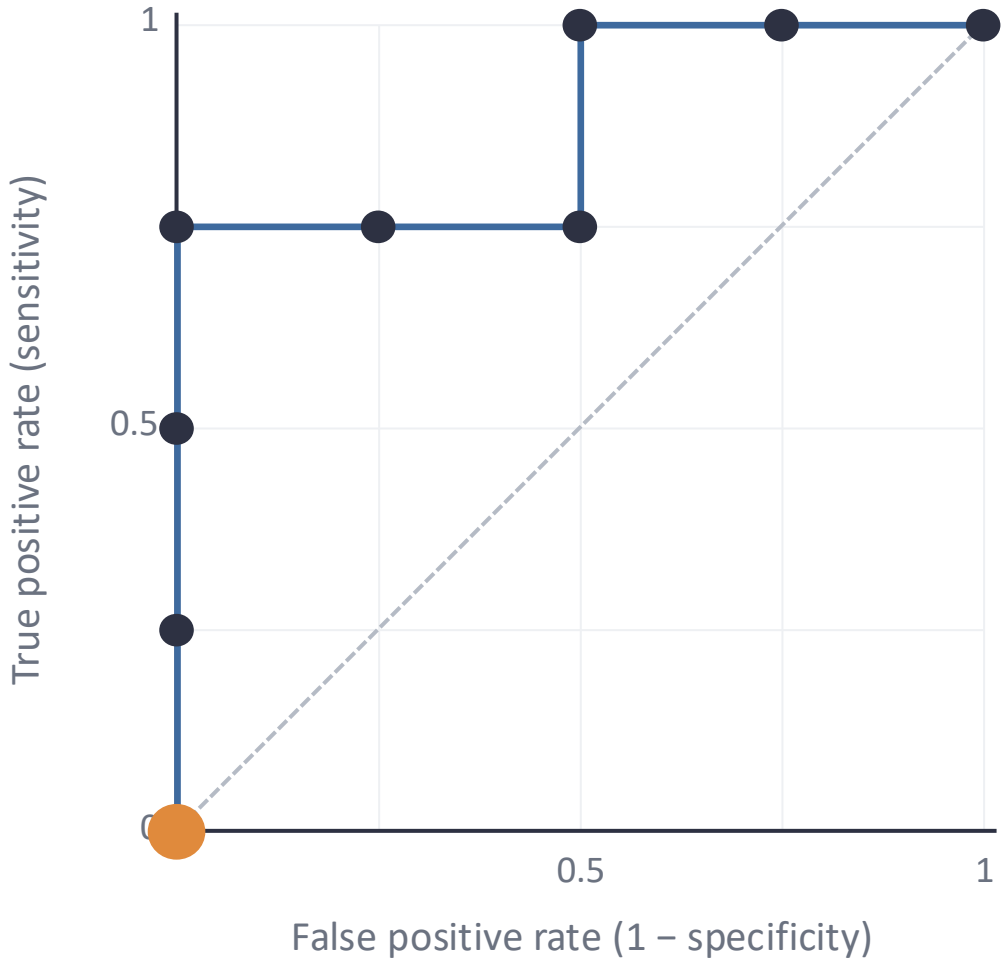
Building the ROC curve: step 7

Raising it further only loses flu cases, until nobody is flagged: (0, 0).
Joining the points gives the ROC curve.



	Flu	No flu
Flu	0	0
No flu	4	4

TPR = 0/4 = 0.00
FPR = 0/4 = 0.00

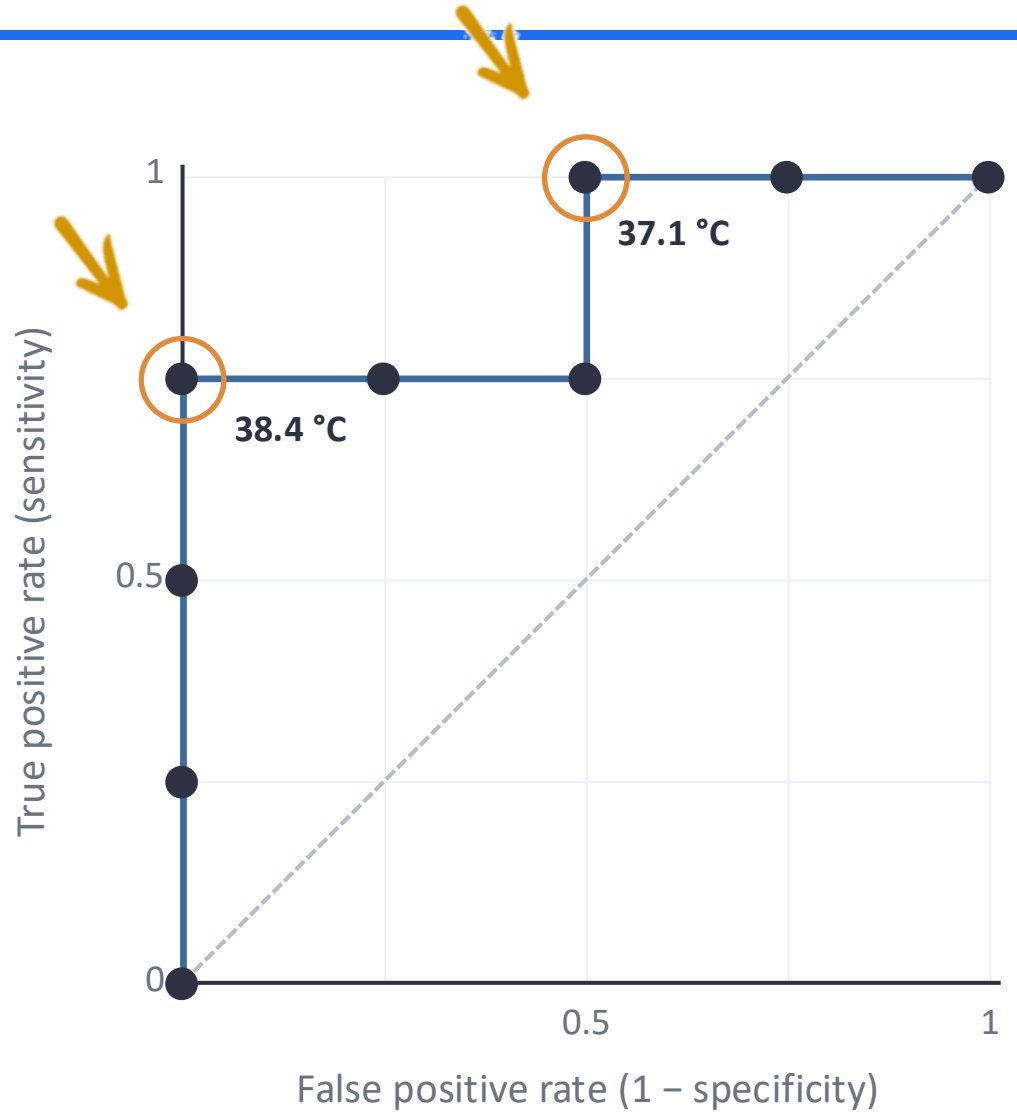


Reading the ROC curve

Points towards the **top-left** are better:
many flu cases caught, few false alarms.

38.4 °C: no false alarms, three of four cases caught.
37.1 °C: every case caught, half the healthy patients flagged.

The best threshold depends on which mistake costs more.



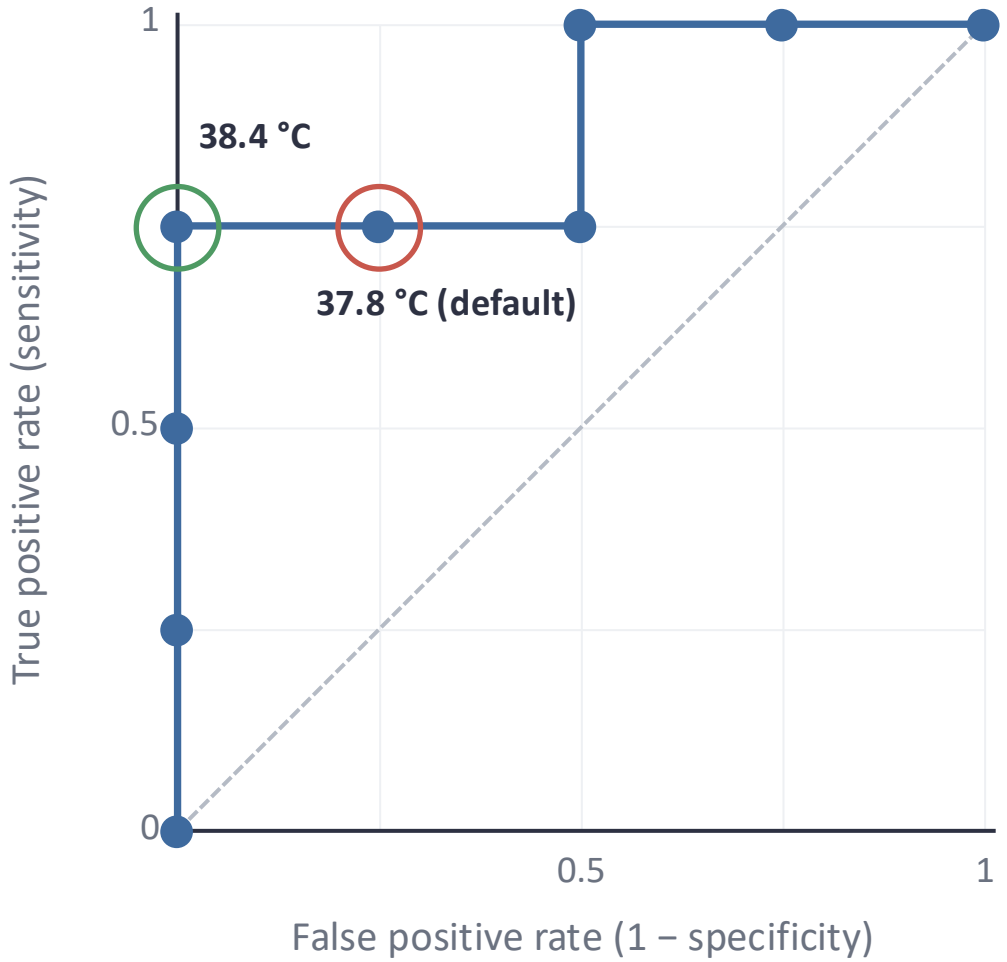
Some thresholds are never worth using

Without opening a single confusion matrix, the curve shows that some thresholds are simply worse than others.

37.8 °C and 38.4 °C both catch three of the four flu cases, but 37.8 °C also (wrongly) flags a healthy patient.

Same height, further left: 38.4 °C is better.

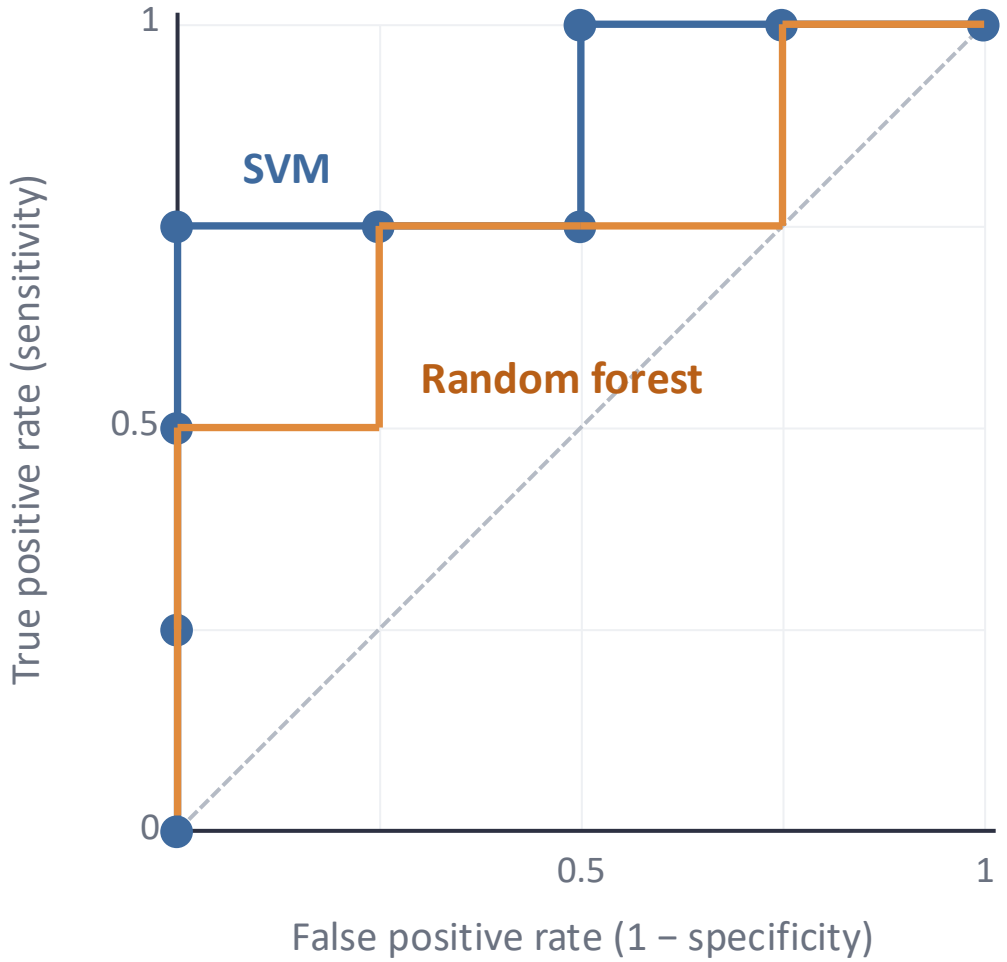
Note that 37.8 °C was the SVM's default threshold: the default is not always the best choice.



What ROC is better?

What model has the best ROC?

Let's compare the areas!



AUC and Precision

Summarising a whole ROC curve in one number

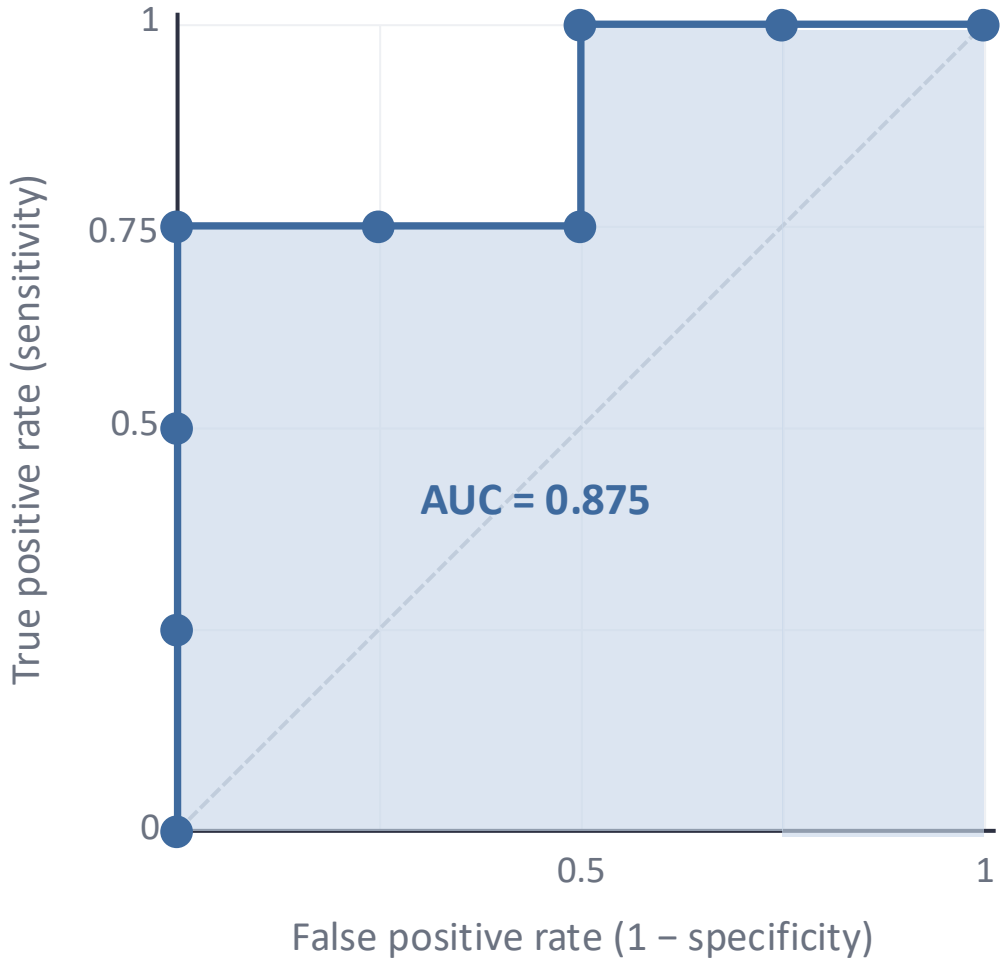
The Area Under the Curve (AUC)

The **AUC** is the area beneath the ROC curve.

It summarises how well the model performs across every possible threshold, in a single number between 0 and 1.

For our SVM:

AUC = 0.75 × 0.5 + 1 × 0.5 = 0.875



How to read an AUC value

AUC		
1.0	**Perfect**	Every flu patient scores higher than every healthy patient.
0.875	**Our SVM**	Good, but some patients are ranked in the wrong order.
0.5	**Chance**	The diagonal: no better than flipping a coin.

- Another way to read it: **the AUC is the chance that a random flu patient gets a higher score than a random healthy patient**.
- The SVM orders 14 of the 16 possible pairs correctly, and 14/16 = 0.875.

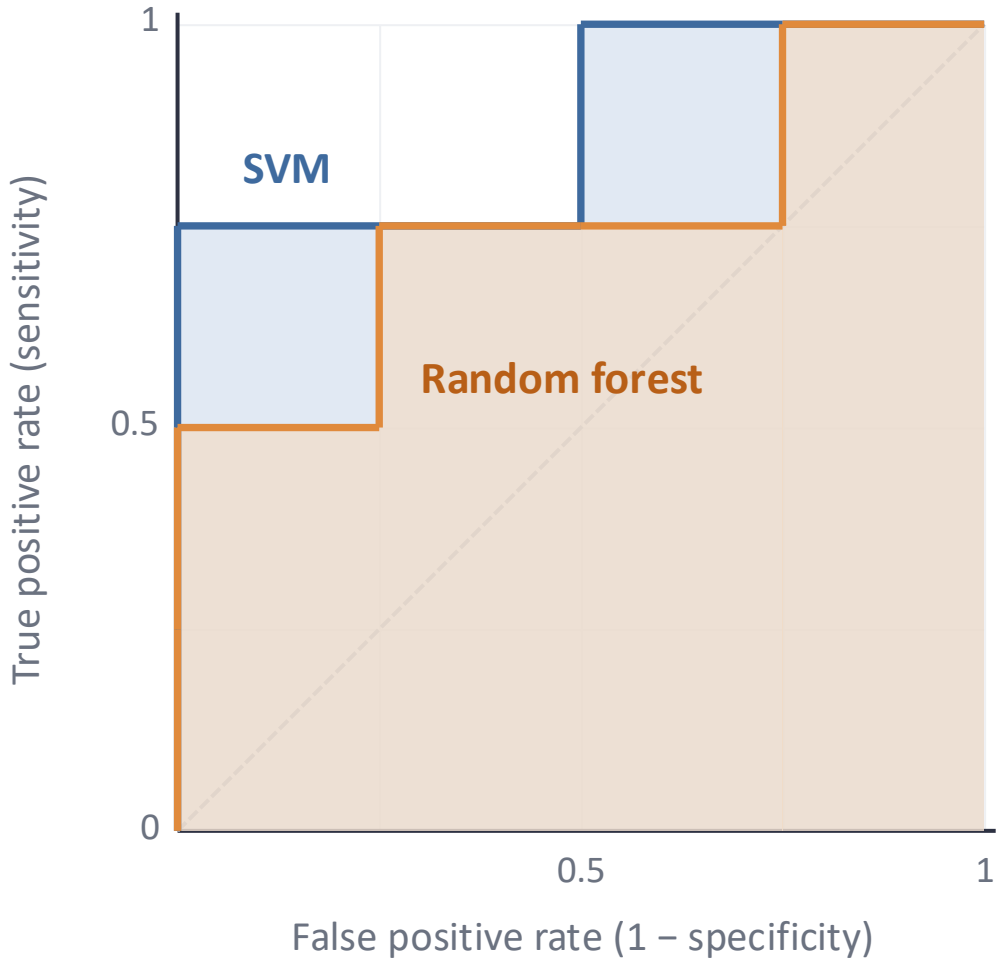
Comparing models with the AUC

Suppose we also train a **random forest** and draw its ROC curve on the same patients.

SVM: AUC = 0.875
Random forest: AUC = 0.75

The larger area suggests the SVM is the better model here.

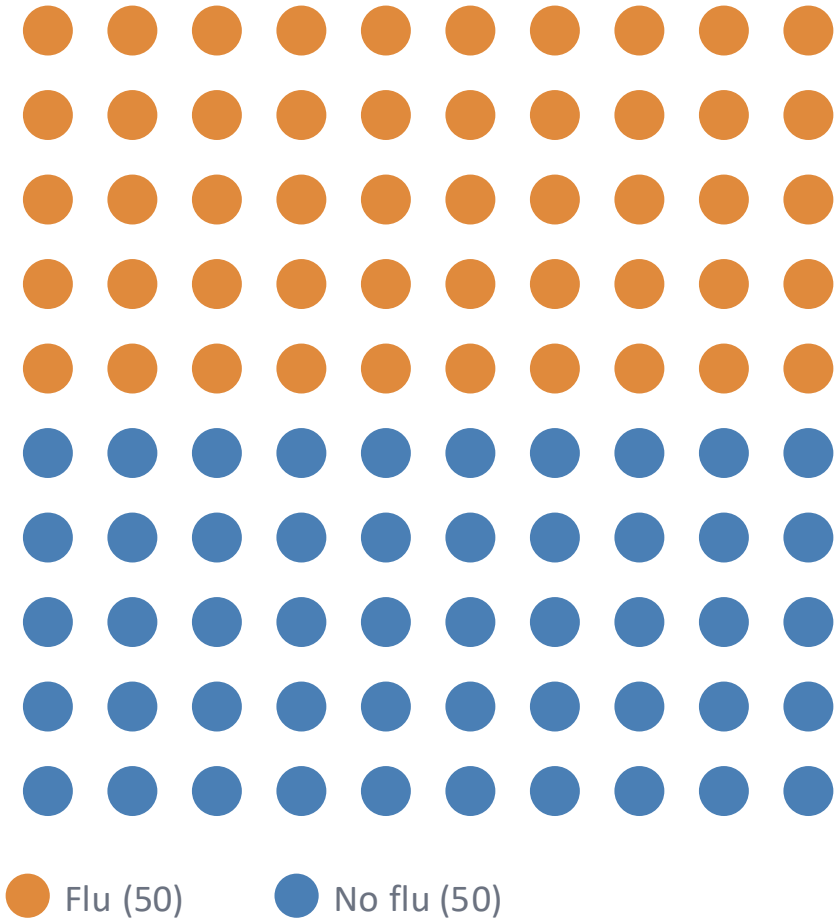
- *The AUC compares models*
- *The ROC curve then helps choose a threshold*



Thought Experiment 1

- Consider the data
  - Assume that the data is balanced (equal number of positive and negative examples)
  - Consider a random classifier
    - This classifier will generate a random score of any example given as input
    - **What will be its accuracy?**
  - Consider a classifier which generates a score of +1 (positive class) for all inputs
    - **What will be its accuracy?**

Thought experiment 1: balanced data



The data: 100 patients, 50 with flu and 50 without (balanced).

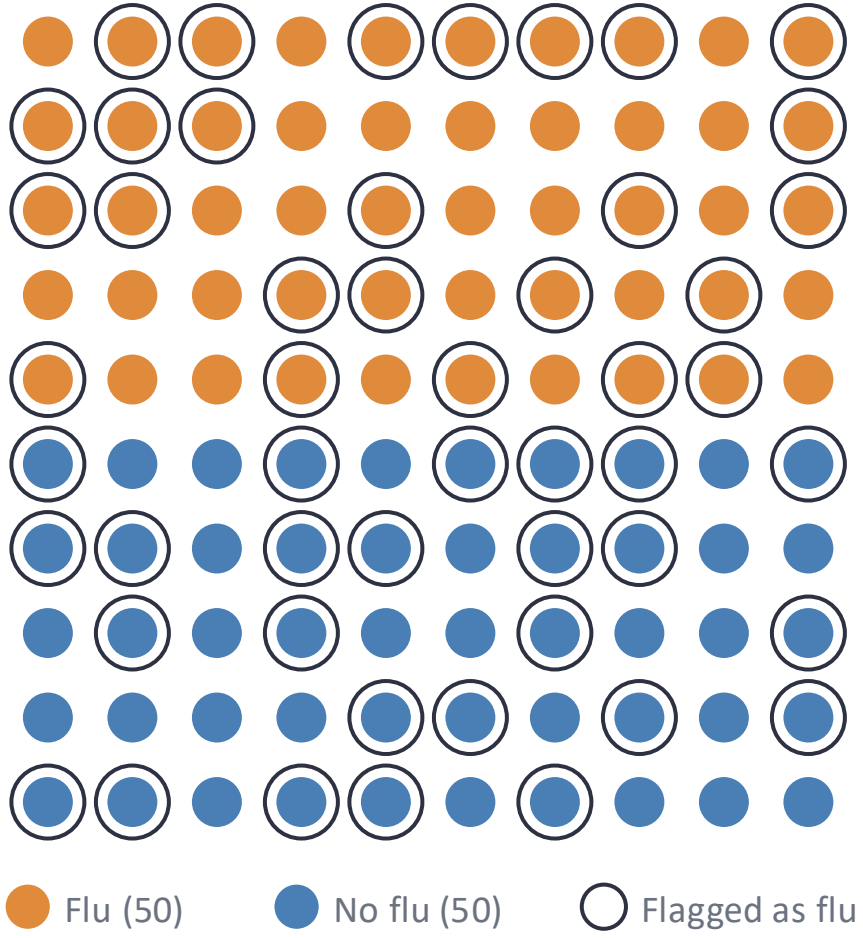
- **A random classifier**

Gives each patient a random score, ignoring their temperature.

- **An always-positive classifier**

Predicts Flu (+1) for every patient.

Balanced data: the **random classifier**

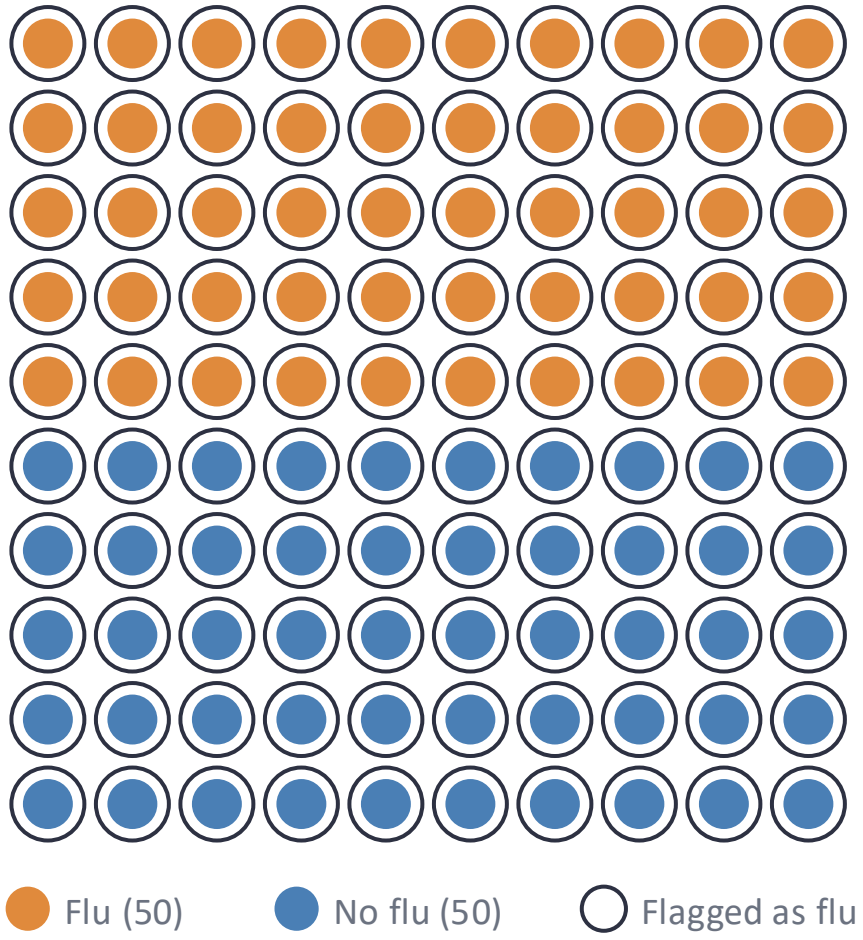


		Actual: Flu	Actual: No flu
Predicted	Flu	25	25
	No flu	25	25

Accuracy = (25 + 25) / 100 = 50%

No better than tossing a coin.

Balanced data: **always predicting Flu**



		Actual: Flu	Actual: No flu
Predicted	Flu	50	50
	No flu	0	0

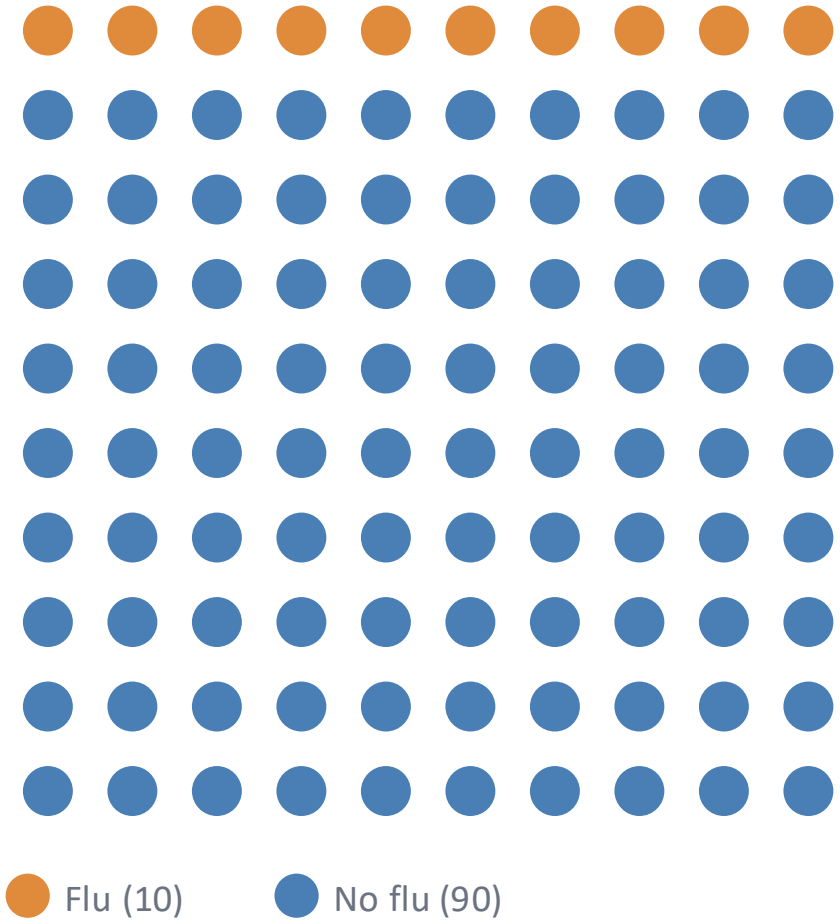
Accuracy = (50 + 0) / 100 = 50%

- Every flu case is caught, but every healthy patient is a false alarm.

Thought Experiment 2

- Consider the data
  - Assume that the data is imbalanced (#Neg>>#Pos)
  - Consider a random classifier
    - This classifier will generate a random score of any example given as input
    - **What will be its accuracy?**
  - Consider a classifier which generates a score of +1 (positive class) for all inputs
    - **What will be its accuracy?**

Thought experiment 2: **imbalanced data**



The data: 100 patients, only 10 with flu and 90 without (#Neg ≫ #Pos).

- **A random classifier**

  Gives each patient a random score, ignoring their temperature.

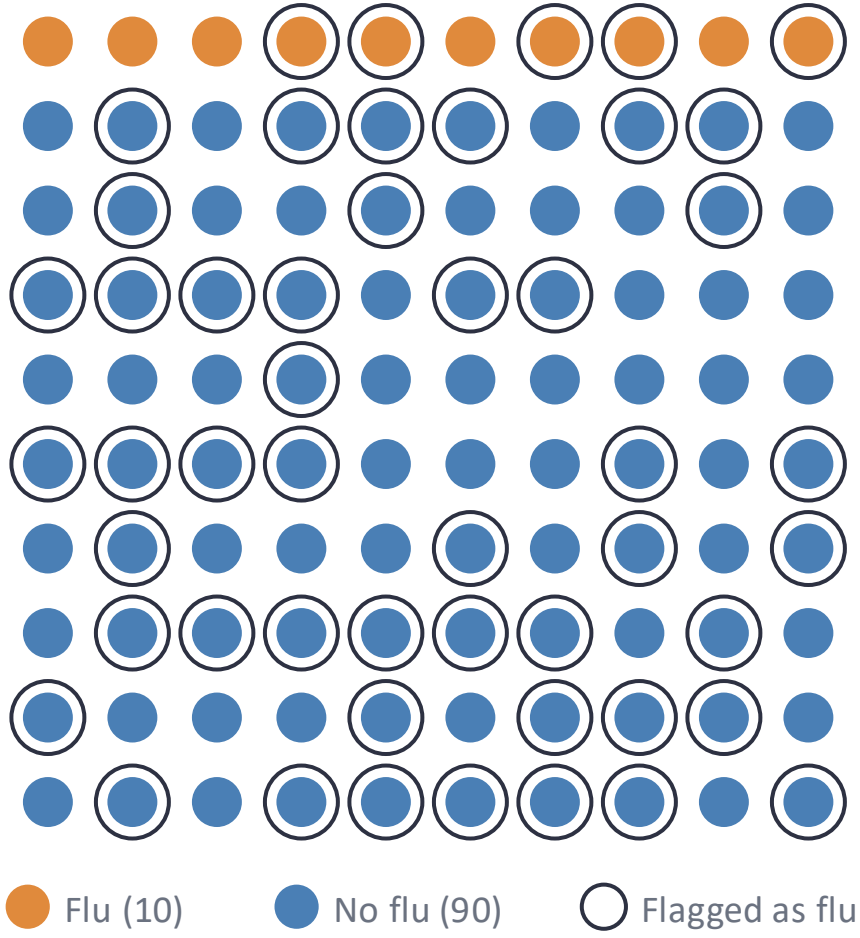
- **An always-positive classifier**

  Predicts Flu (+1) for every patient.

What will the accuracy of each be?

Accuracy = (TP + TN) / all patients

Imbalanced data: **the random classifier**

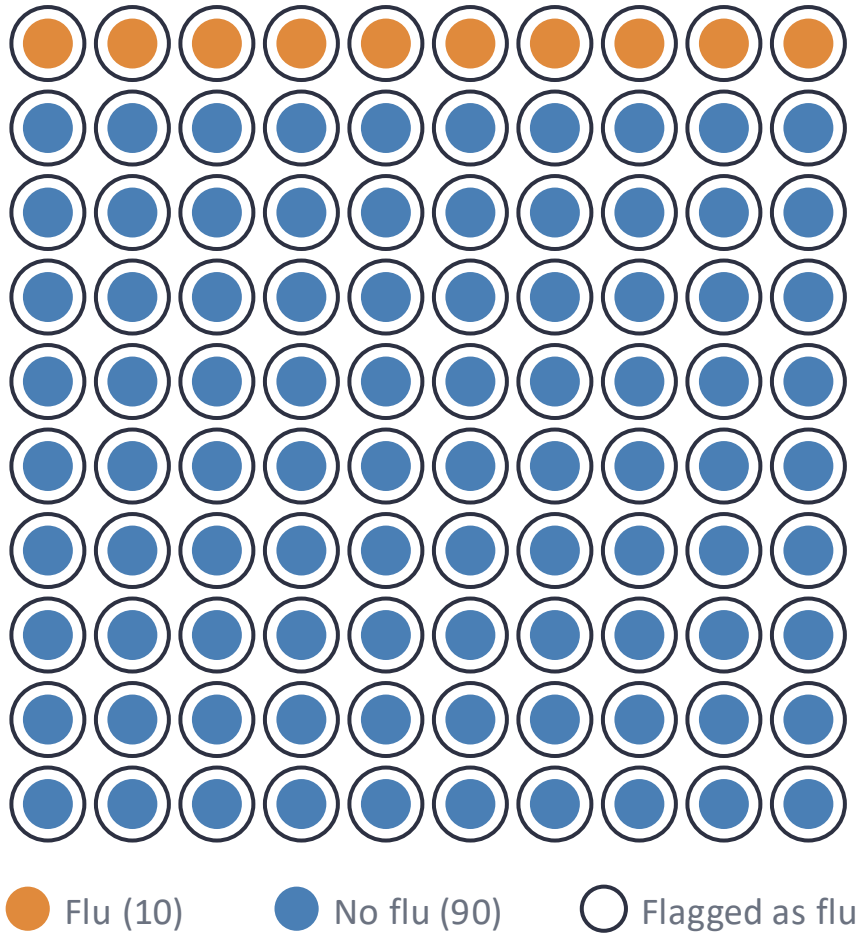


		Actual: Flu	Actual: No flu
Predicted	Flu	5	45
	No flu	5	45

Accuracy = (5 + 45) / 100 = 50%

It ignores the patients, so the class balance makes no difference.

Imbalanced data: **always predicting Flu**

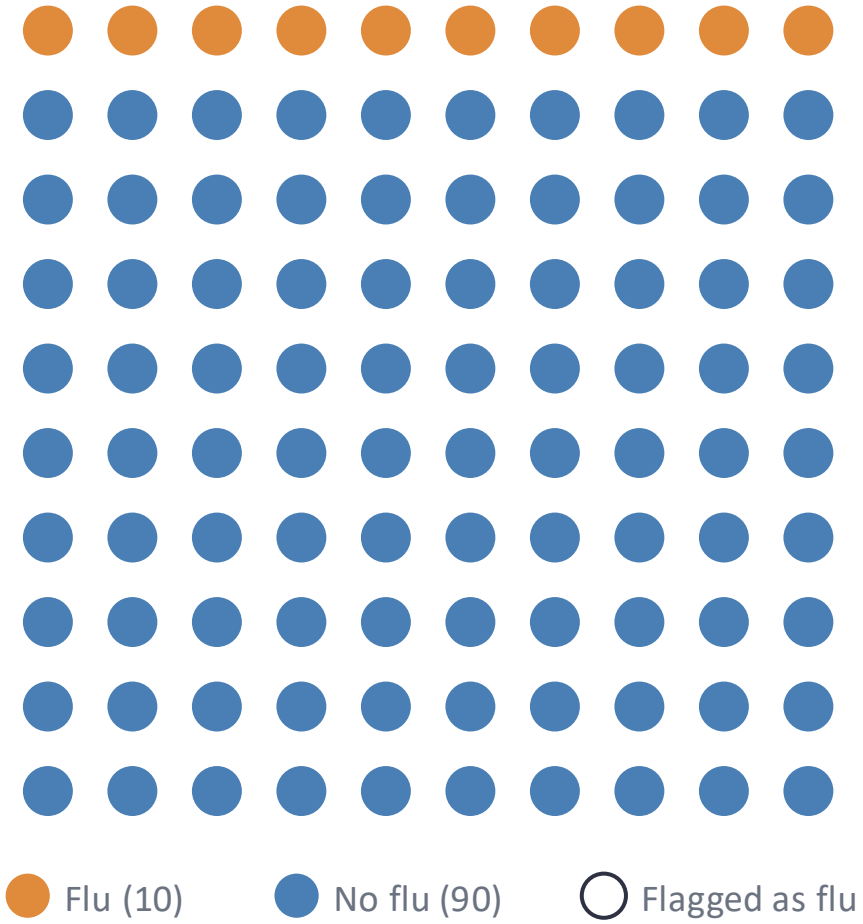


Predicted \ Actual	Flu	No flu
Flu	10	90
No flu	0	0

Accuracy = (10 + 0) / 100 = 10%

All 10 flu cases are caught, but 90 healthy patients are false alarms. Accuracy drops to 10%.

The twist: always predicting No flu



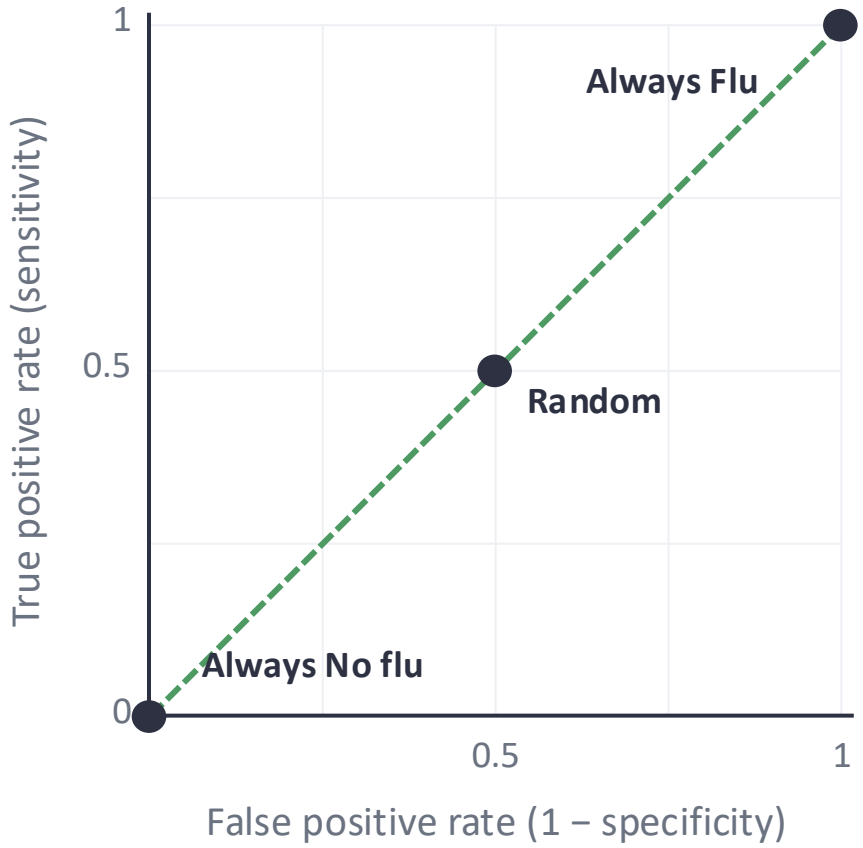
		Actual: Flu	Actual: No flu
Predicted	Flu	0	0
	No flu	10	90

Accuracy = (0 + 90) / 100 = 90%

- 90% accuracy, yet it never detects a single case of flu (sensitivity 0).
  With imbalanced data, high accuracy can be meaningless.

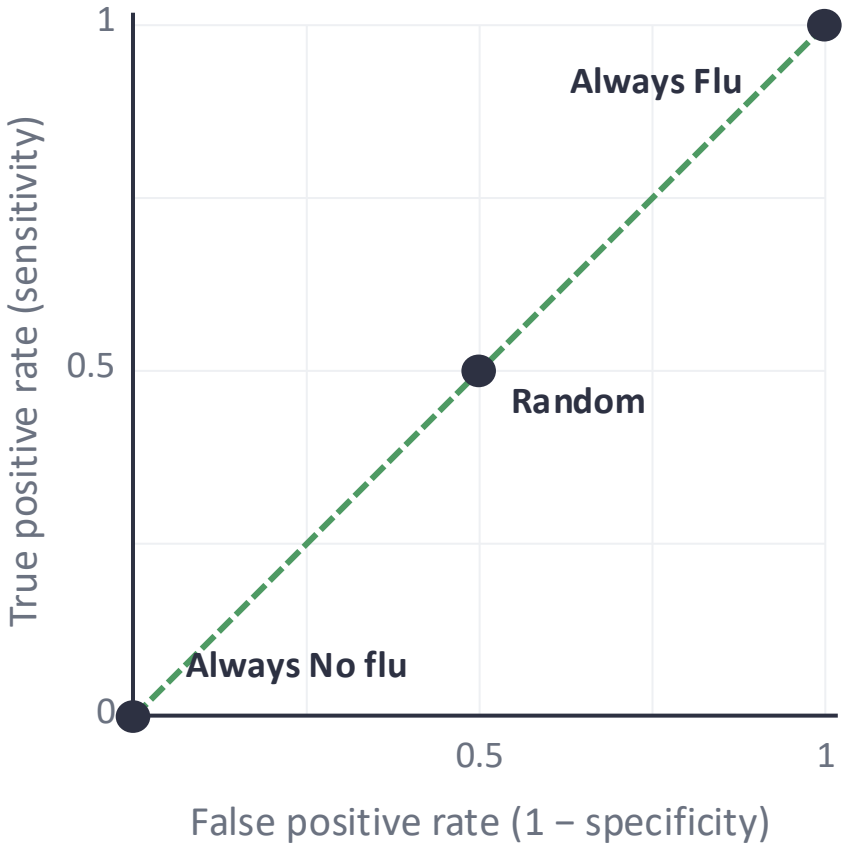
What the thought experiments show

Data	Classifier	Accuracy	Sensitivity	Specificity
Balanced	Random	**50%**	0.5	0.5
Balanced	Always Flu	**50%**	1	0
Imbalanced	Random	**50%**	0.5	0.5
Imbalanced	Always Flu	**10%**	1	0
Imbalanced	Always No flu	**90%**	**0**	1



What the thought experiments show

Data	Classifier	Accuracy	Sensitivity	Specificity
Balanced	Random	**50%**	0.5	0.5
Balanced	Always Flu	**50%**	**1**	**0**
Imbalanced	Random	**50%**	0.5	0.5
Imbalanced	Always Flu	**10%**	**1**	**0**
Imbalanced	Always No flu	**90%**	**0**	1



Accuracy changes with the class balance; sensitivity and specificity do not.

Every trivial classifier sits on the ROC diagonal (AUC 0.5), however the data are balanced.

Another measure: Precision

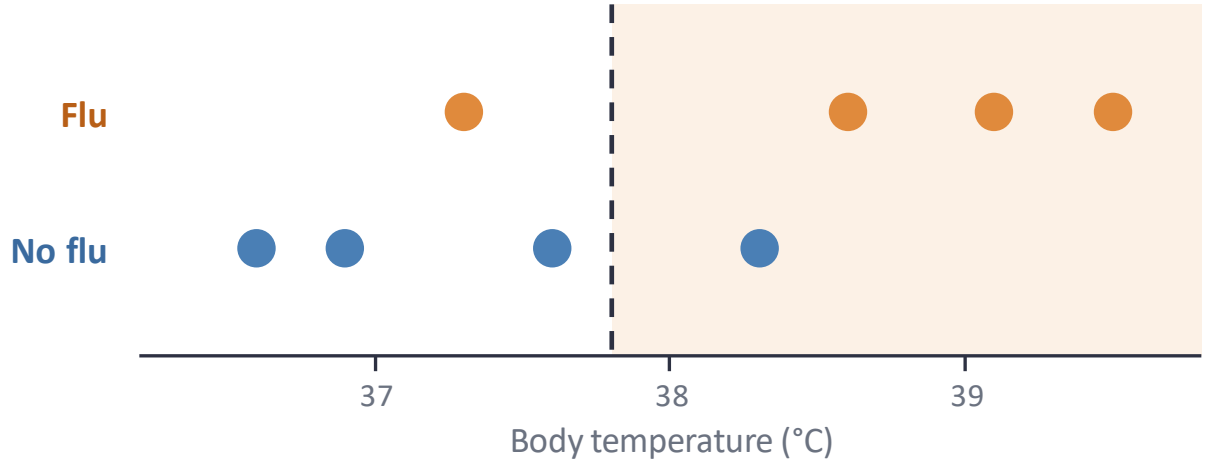
Precision asks: of all the patients we flag as having flu, what proportion really have it?

It uses only the Predicted Flu row.

At the default threshold, four patients are flagged: three have flu, one does not.

		Actual: Flu	Actual: No flu
Predicted	Flu	3	1
	No flu	1	3

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{3}{3 + 1} = 0.75$$

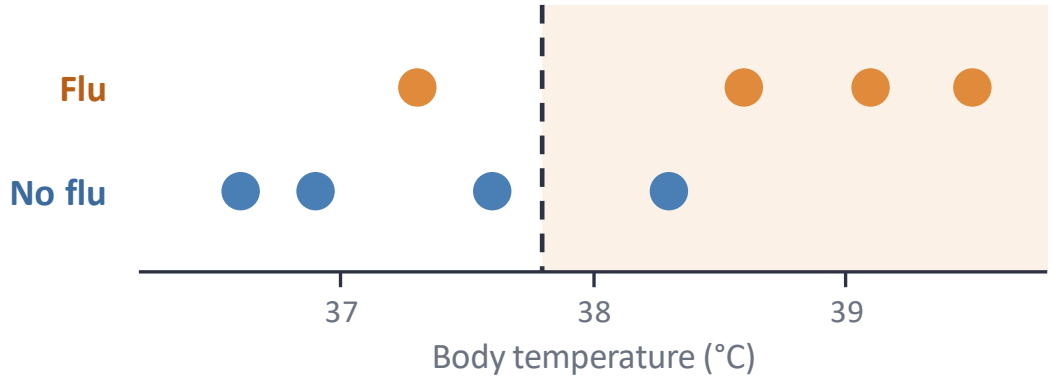


When flu is rare, precision tells us more

- Same sensitivity (PTR) and false positive rate (FTR), but very different precision (P).

In flu season

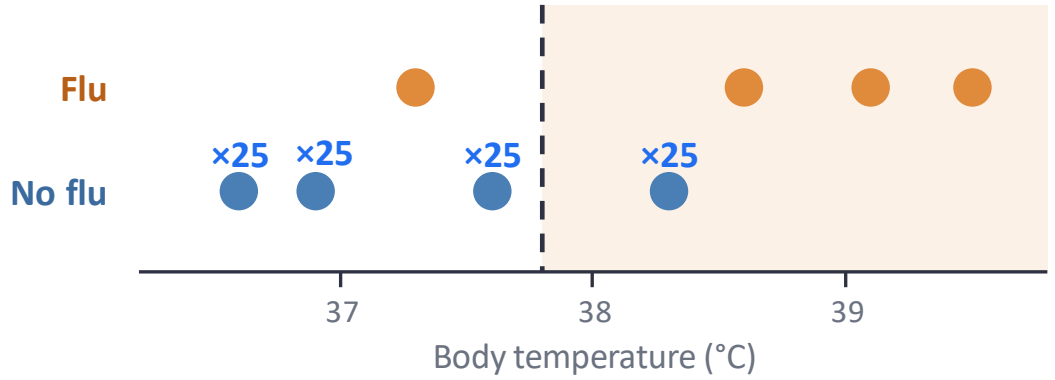
4 flu patients, 4 healthy (prevalence 50%)



Sensitivity = 3/4 = 0.75
False positive rate = 1/4 = 0.25
Precision = 3/(3 + 1) = **0.75**

Out of season

4 flu patients, 100 healthy: each blue dot stands for 25



Sensitivity = 3/4 = 0.75
False positive rate = 25/100 = 0.25
Precision = 3/(3 + 25) = **0.11**

- Same point on the ROC curve, yet out of season **almost 9 in 10 flagged patients do not have flu.**

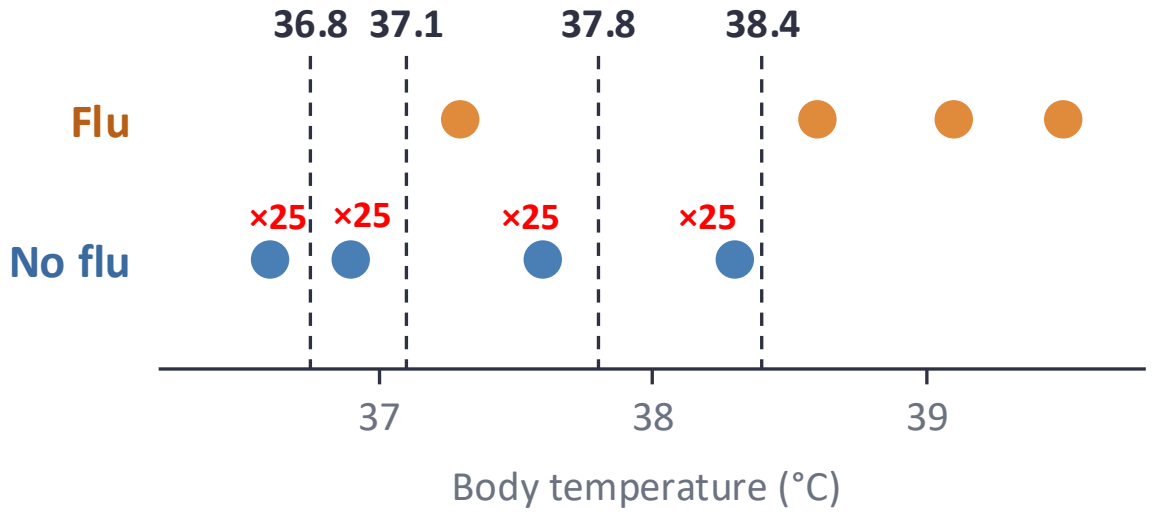
Same ROC curve, different precision

Cut-off (°C)	Sensitivity (TPR)	FPR	Precision in season	Precision out of season
36.8	1.00	0.75	4/7 = 0.57	4/79 = 0.05
37.1	1.00	0.50	4/6 = 0.67	4/54 = 0.07
37.8 (default)	0.75	0.25	3/4 = 0.75	3/28 = 0.11
38.4	0.75	0.00	3/3 = 1.00	3/3 = 1.00

TPR and FPR are identical in both settings, so the ROC curve and the AUC do not change.

Precision falls at every cut-off with a healthy patient above it; only 38.4 °C keeps a precision of 1.

Out of season, we would choose a stricter threshold, even though the ROC curve looks exactly the same.



Precision–recall curves show the difference

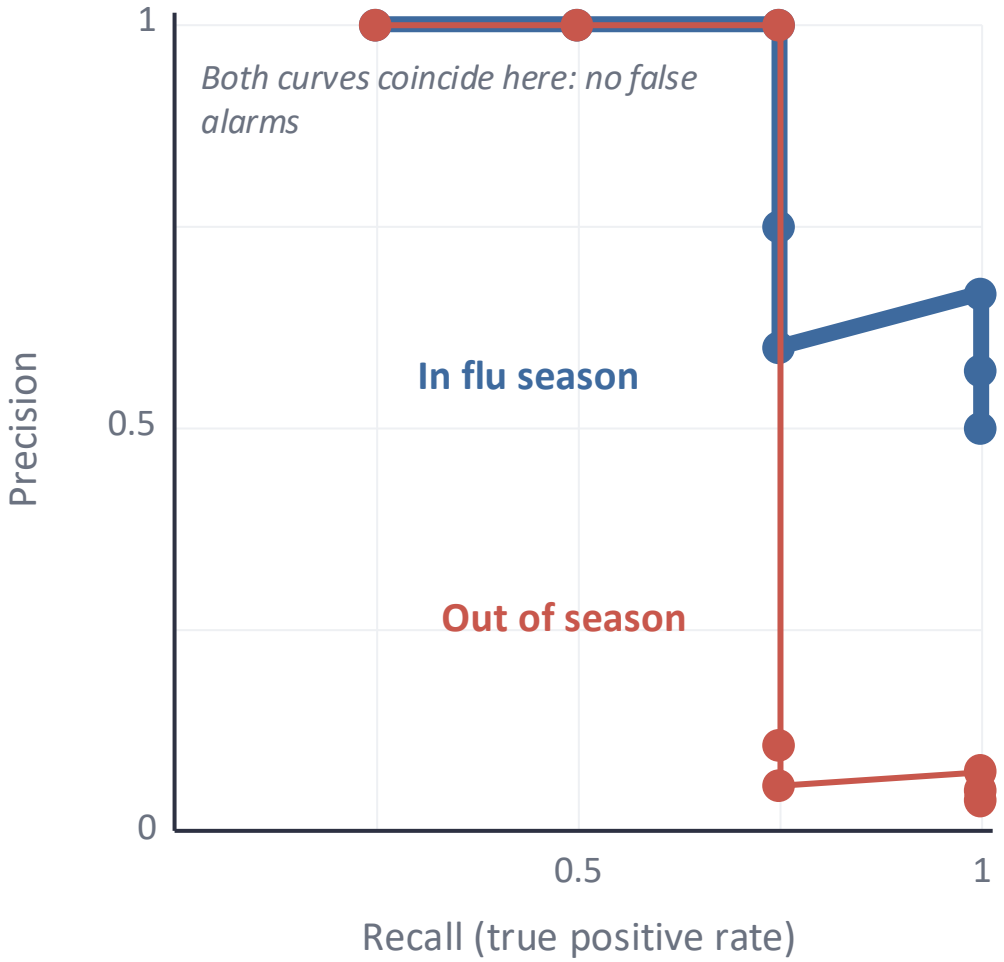
A **precision–recall curve** plots precision against recall (True Positive Rate) for every threshold.

Both curves come from the same classifier, with the same ROC curve.

The same classifier:

- In **flu season**, precision stays high.
- **Out of season**, it collapses wherever there are false alarms.

Precision depends on prevalence.



ROC or Precision–Recall?

ROC curve

- Measures how well the model separates the two classes
- Does not change with prevalence
- Good for comparing models (AUC)

Precision–recall curve

- Measures how trustworthy a positive prediction is
- Changes with prevalence
- Preferable when positives are rare and false alarms are costly

Two classifiers, or one classifier in two populations, can have identical ROC performance but very different practical usefulness.

Key points

Confusion matrix	Counts the four outcomes: true and false positives, true and false negatives.
Sensitivity and specificity	How well we catch positives, and how well we clear negatives.
ROC curve	Shows the trade-off between the two for every threshold; better points sit top-left.
AUC	Summarises the curve in one number, to compare models. 0.5 is chance, 1 is perfect.
Precision	Of everything flagged, how much is right. Valuable when the positive class is rare.

ROC: Review Questions

- What will be ROC curve for a perfect classifier?
- What will the ROC Curve of a random classifier look like?
- What will the ROC curve of a classifier that always predicts the positive class look like?
- What are the underlying assumptions of the ROC curve?
- What part of the ROC curve is the most important?

ROC and PR Curves in Scikit-Learn

- http://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
- http://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html#example-model-selection-plot-roc-py

```
from sklearn.metrics import *
P,R = precision_recall_curve(Y,Z)
AUCPR = average_precision_score(Y,Z)
```

Very Important Exercise and associated Questions

- See notebook provided in the module website

Confusion Matrix

		True condition			
	Total population	Condition positive	Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**	**False positive** (Type I error)	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Test outcome positive}}$	False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Test outcome positive}}$
	Predicted condition negative	**False negative** (Type II error)	**True negative**	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Test outcome negative}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Test outcome negative}}$
	Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	True positive rate (TPR), Sensitivity, Recall $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	True negative rate (TNR), Specificity (SPC) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$	

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

Metrics

Classification		
'accuracy'	metrics.accuracy_score	
'balanced_accuracy'	metrics.balanced_accuracy_score	
'top_k_accuracy'	metrics.top_k_accuracy_score	
'average_precision'	metrics.average_precision_score	
'neg_brier_score'	metrics.brier_score_loss	
'f1'	metrics.f1_score	for binary targets
'f1_micro'	metrics.f1_score	micro-averaged
'f1_macro'	metrics.f1_score	macro-averaged
'f1_weighted'	metrics.f1_score	weighted average
'f1_samples'	metrics.f1_score	by multilabel sample
'neg_log_loss'	metrics.log_loss	requires predict_proba support
'precision' etc.	metrics.precision_score	suffixes apply as with 'f1'
'recall' etc.	metrics.recall_score	suffixes apply as with 'f1'
'jaccard' etc.	metrics.jaccard_score	suffixes apply as with 'f1'
'roc_auc'	metrics.roc_auc_score	
'roc_auc_ovr'	metrics.roc_auc_score	
'roc_auc_ovo'	metrics.roc_auc_score	
'roc_auc_ovr_weighted'	metrics.roc_auc_score	
Matthews correlation coefficient	metrics.matthews_corrcoef	
'roc_auc_ovo_weighted'	metrics.roc_auc_score	

You are expected to know at least those in **bold**. Also the difference between micro and macro averaging.
https://scikit-learn.org/stable/modules/model_evaluation.html

Try understanding why "MCC is better than F-measures":
https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-019-6413-7

Regression	
'explained_variance'	metrics.explained_variance_score
'max_error'	metrics.max_error
'neg_mean_absolute_error'	metrics.mean_absolute_error
'neg_mean_squared_error'	metrics.mean_squared_error
'neg_root_mean_squared_error'	metrics.root_mean_squared_error
'neg_mean_squared_log_error'	metrics.mean_squared_log_error
'neg_root_mean_squared_log_error'	metrics.root_mean_squared_log_error
'neg_median_absolute_error'	metrics.median_absolute_error
'r2'	metrics.r2_score
'neg_mean_poisson_deviance'	metrics.mean_poisson_deviance
'neg_mean_gamma_deviance'	metrics.mean_gamma_deviance
'neg_mean_absolute_percentage_error'	metrics.mean_absolute_percentage_error
'd2_absolute_error_score'	metrics.d2_absolute_error_score
'd2_pinball_score'	metrics.d2_pinball_score
'd2_tweedie_score'	metrics.d2_tweedie_score
Spearman, Pearson Corr and Kendall tau	scipy.stats.pearsonr scipy.stats.spearmanr scipy.stats.kendalltau

Clustering	
'adjusted_mutual_info_score'	metrics.adjusted_mutual_info_score
'adjusted_rand_score'	metrics.adjusted_rand_score
'completeness_score'	metrics.completeness_score
'fowlkes_mallows_score'	metrics.fowlkes_mallows_score
'homogeneity_score'	metrics.homogeneity_score
'mutual_info_score'	metrics.mutual_info_score
'normalized_mutual_info_score'	metrics.normalized_mutual_info_score
'rand_score'	metrics.rand_score
'v_measure_score'	metrics.v_measure_score

Reading

James, Witten, Hastie, Tibshirani and Taylor (2023). "***An Introduction to Statistical Learning with Applications in Python (ISLP)***".

- 2.2.3 The classification setting: training and test error
- 4.4.2 Confusion matrix, sensitivity, specificity, thresholds and the ROC curve
- 5.1.1 The validation set approach
- 9.3.3 Comparing models with ROC curves

Hastie, Tibshirani and Friedman (2009). "***The Elements of Statistical Learning (ESL), 2nd edition.***"

- 7.2 Training, validation and test sets
- 7.10 Cross-validation
- 9.2.5 Sensitivity, specificity, ROC curve and AUC