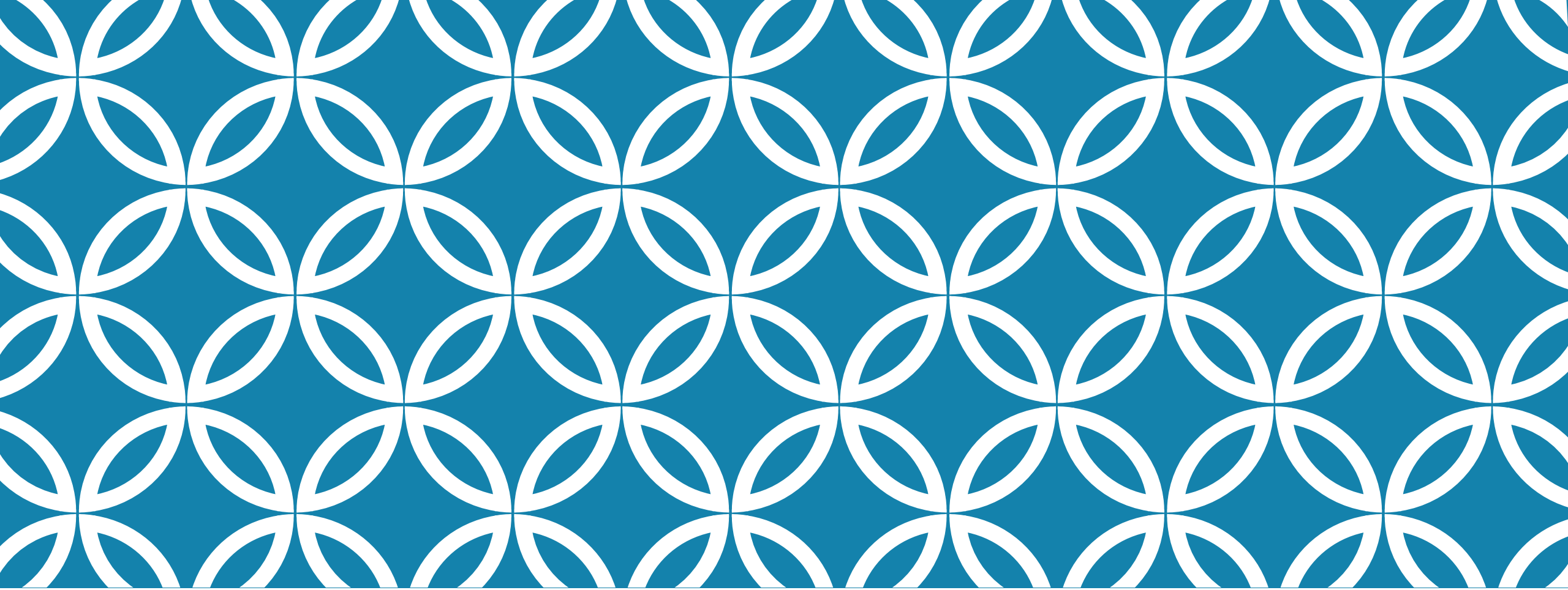


CS5228 LECTURE 5: CLASSIFICATION AND REGRESSION

Bryan Hooi

School of Computing

National University of Singapore

Slide credits: Wang Wei, Ng See Kiong, Wynne Hsu

ANNOUNCEMENTS

- No lecture next week (recess week).
- If you filled in the 1st iteration of the project group matching survey, you should have received your matching results. The 2nd iteration survey has been released; if you still would like to be matched, please fill it out (ending 22 Feb).
- Once your group is decided, please form the group in Canvas > People > Project by 5 March. If you have difficulties, you can email the course staff.

Week	Date	Topics	Tutorials	Important Dates
1	13 Jan	Introduction		
2	20 Jan	No class (public holiday)		
3	27 Jan	Clustering I	Tutorial 1	
4	3 Feb	Clustering II		Release A1 + project
5	10 Feb	Association Rules	Tutorial 2	
6	17 Feb	Regression & Classification I		
Recess		No class		
7	3 Mar	Regression & Classification II	Tutorial 3	A1 due (Sunday 11.59pm), release A2
8	10 Mar	Regression & Classification III		
9	17 Mar	Recommender Systems	Tutorial 4	
10	24 Mar	Graph Mining		
11	31 Mar	Data Stream Mining	Tutorial 5	A2 due (Sunday 11.59pm)
12	7 Apr	No class (public holiday)		
13	14 Apr	Review & Outlook		Project due (Sunday 11.59pm)

REVIEW: ASSOCIATION RULE MINING

Given: a minimum support threshold *minsup*, and a minimum confidence threshold *minconf*

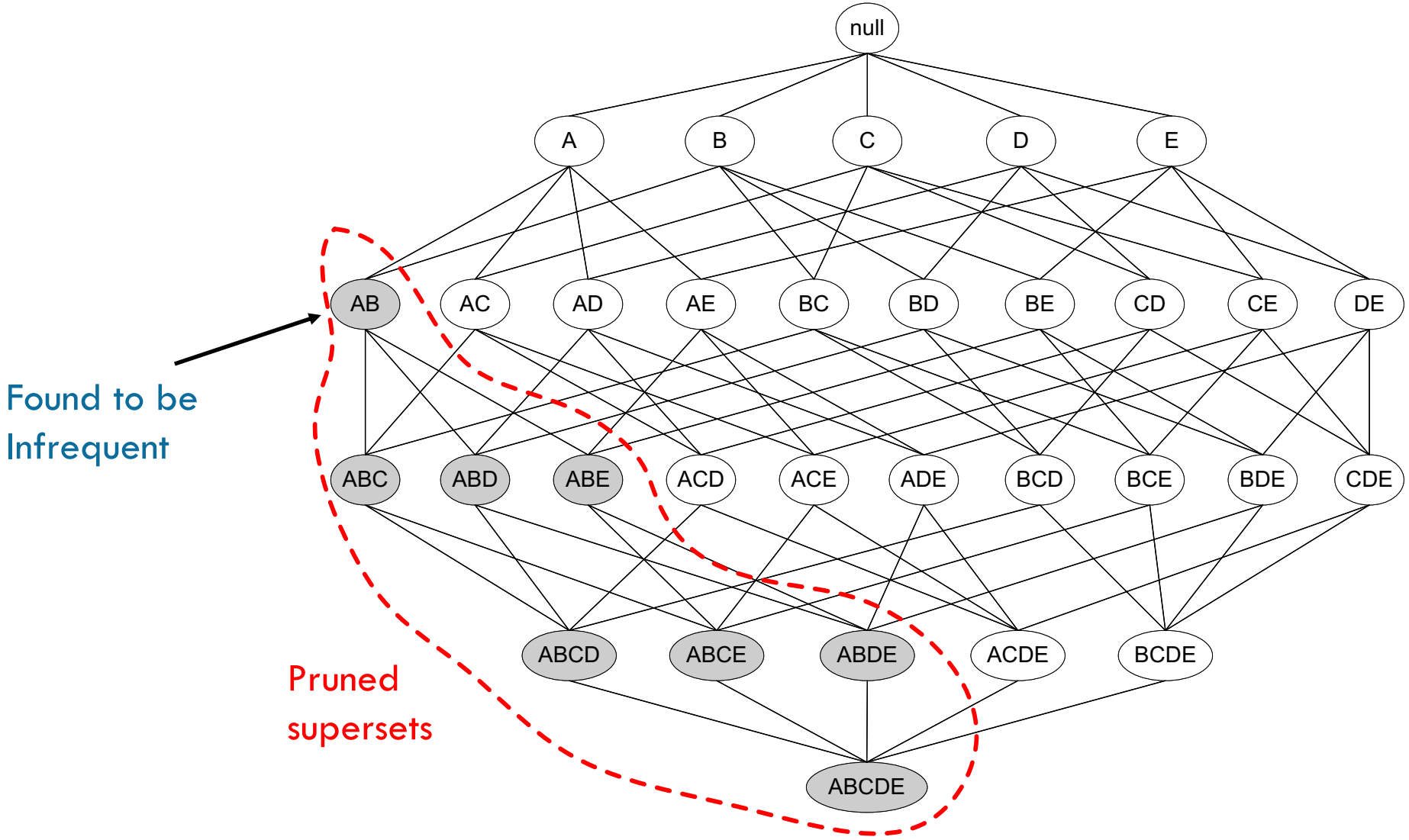
Find: all association rules with support ≥ *minsup*, and confidence ≥ *minconf*

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

minsup = 0.6 minconf = 1

Diaper → Beer, Beer →Diaper,
Milk → Diaper, Diaper → Milk,
Bread → Diaper, Diaper → Bread,
Milk → Bread, Bread → Milk

REVIEW: APRIORI PRINCIPLE

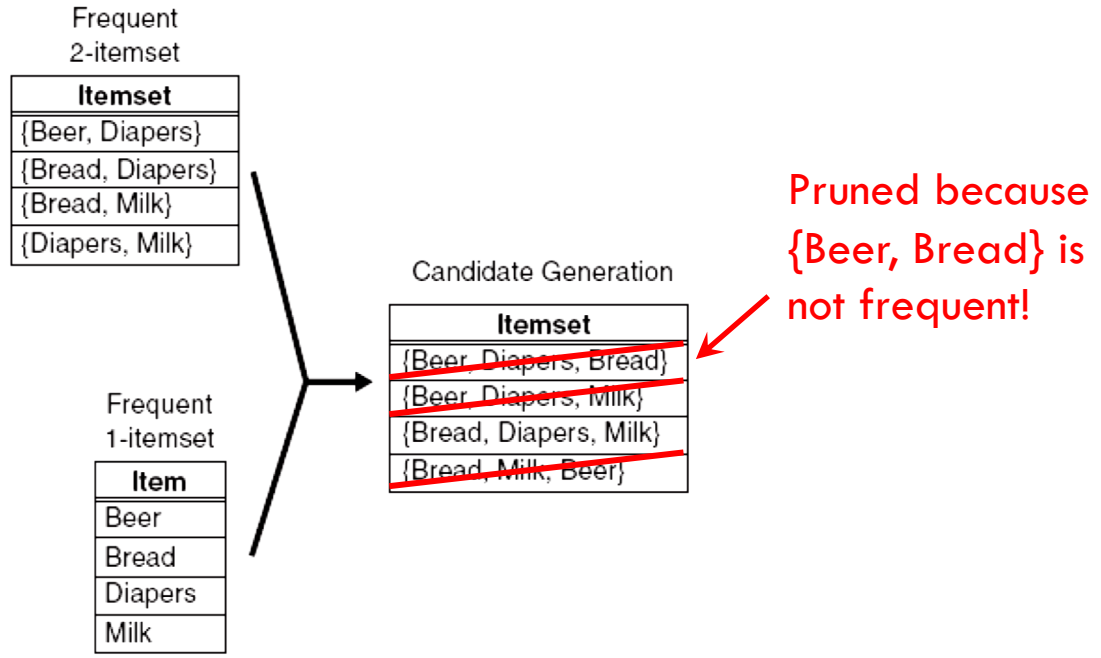


REVIEW: GENERATION STEP (F_{K-1} X F_1 METHOD)

Apriori Algorithm:

- For k = 1, 2, …
  - **Generate** candidate frequent k-itemsets
  - **Filter** candidates to get all frequent k-itemsets

- **Merge** frequent (k-1) and 1-itemsets, then
- **Prune** resulting k-itemsets if they have a (k-1) subset which is not frequent



REVIEW: FILTER STEP

Apriori Algorithm:

- For k = 1, 2, …
  - **Generate** candidate frequent k-itemsets
  - **Filter** candidates to get all frequent k-itemsets

- Query the database to compute support of each candidate
- Filter away candidates with support < *minsup*

Itemset	SC
ab	1
ac	2
ae	1
bc	2
be	3
ce	2

CLASSIFICATION & REGRESSION: OVERVIEW

1. Problem Setup
2. Evaluating Classifiers
3. Nearest Neighbor Methods
4. Tree-Based Methods
5. Ensemble Methods
6. Logistic Regression
7. Deep Learning

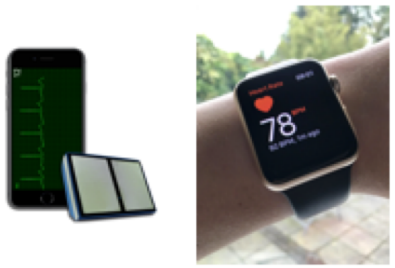
Today's Lecture

OVERVIEW

1. Problem Setup
2. Evaluating Classifiers
3. Nearest Neighbor Methods
4. Tree-Based Methods
5. Ensemble Methods
6. Logistic Regression
7. Deep Learning

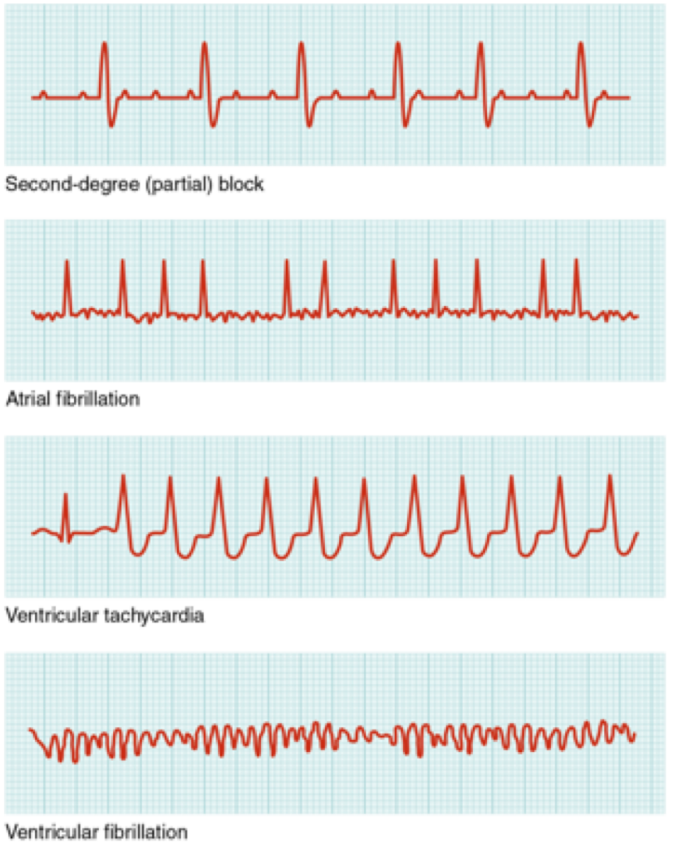
HEART RHYTHM CLASSIFICATION

Medical ECG devices



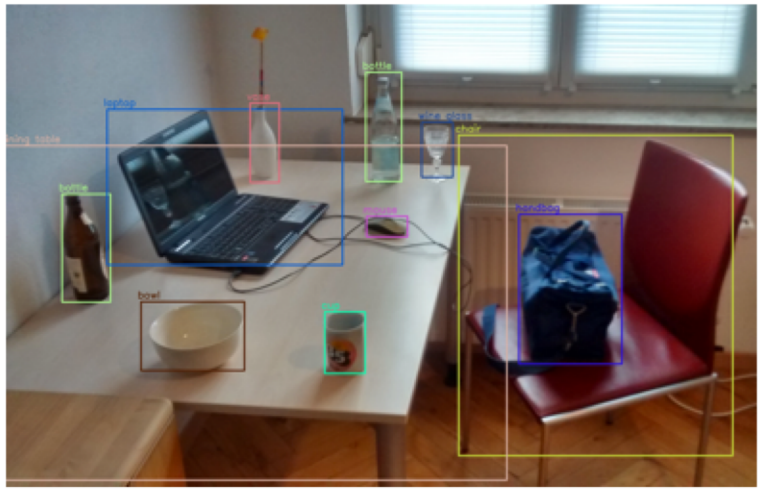
Wearable devices

Classification

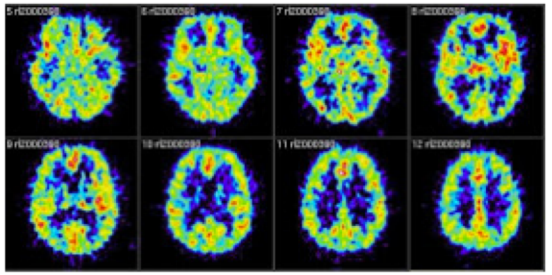


• • •

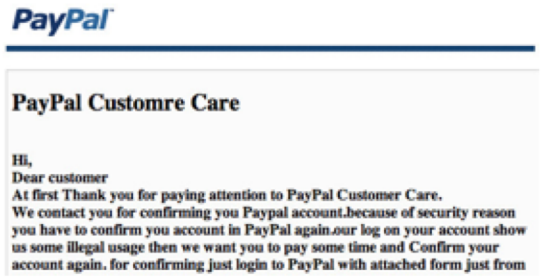
MORE APPLICATIONS



Object Recognition



Medical Imaging

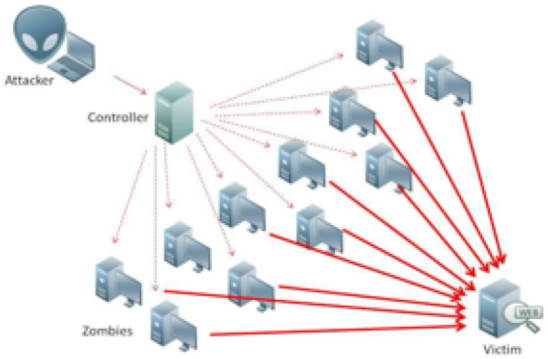


PayPal

PayPal Customre Care

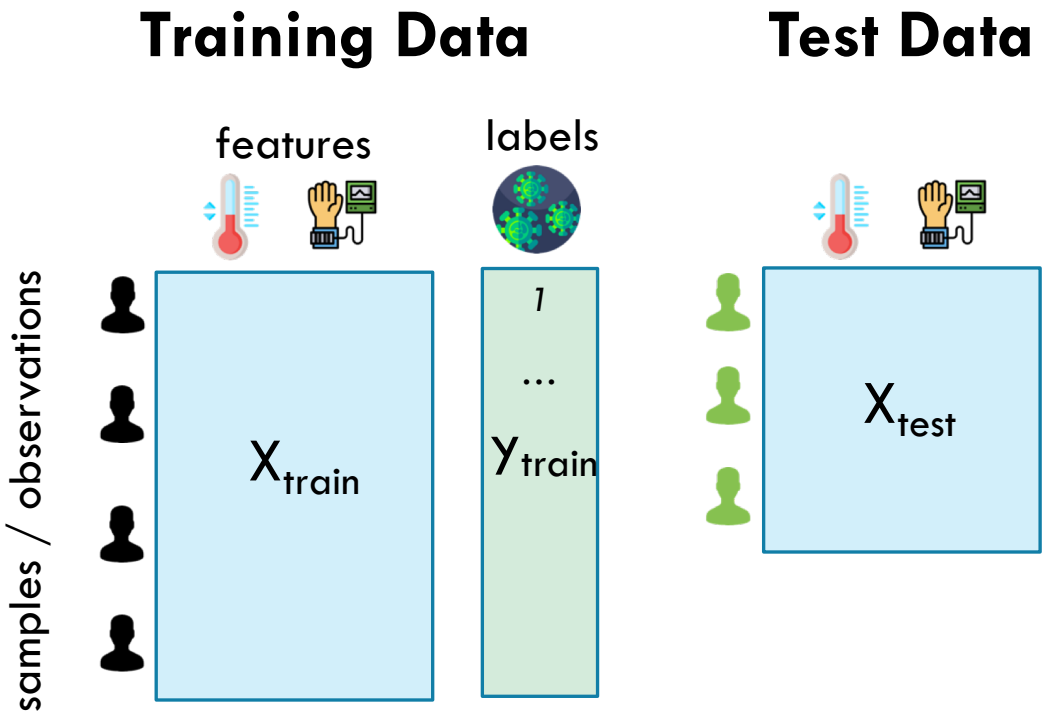
Hi,
Dear customer
At first Thank you for paying attention to PayPal Customer Care.
We contact you for confirming you Paypal account.because of security reason you have to confirm you account in PayPal again.our log on your account show us some illegal usage then we want you to pay some time and Confirm your account again. for confirming just login to PayPal with attached form just from

Spam Detection



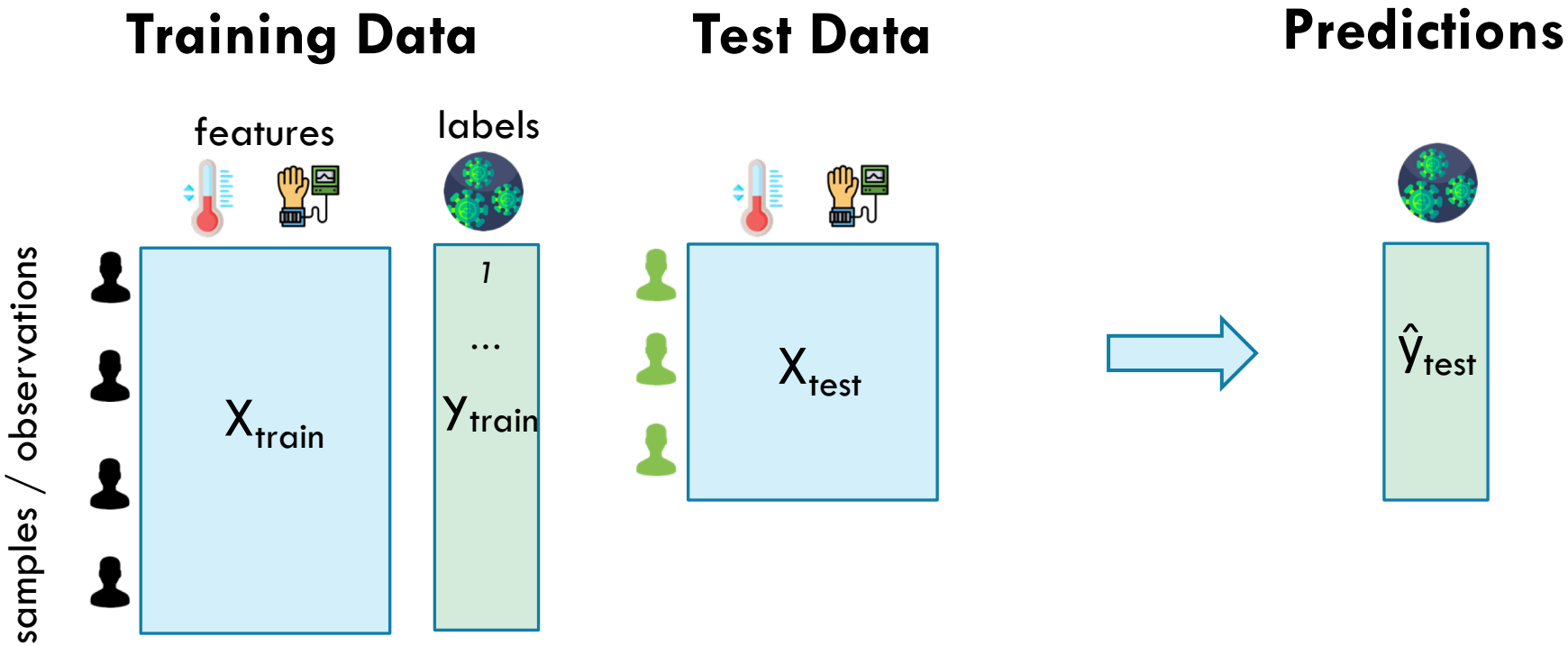
Network Intrusion Detection

CLASSIFICATION: SETUP



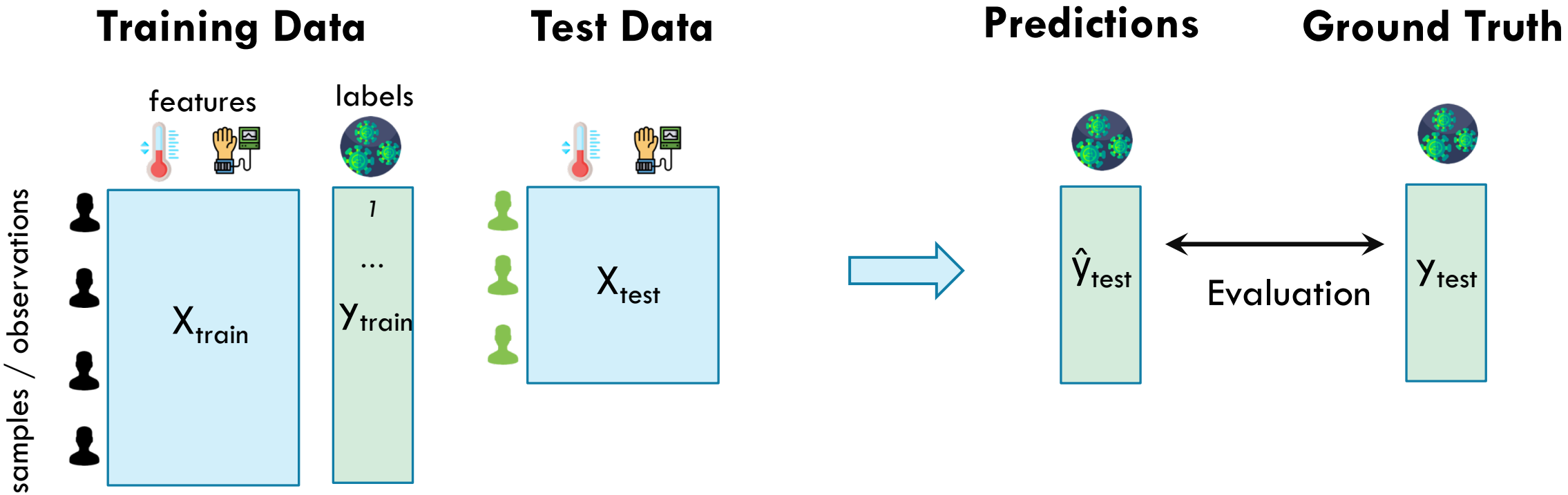
Categorize samples based on a **discrete** label, given training data

CLASSIFICATION: SETUP



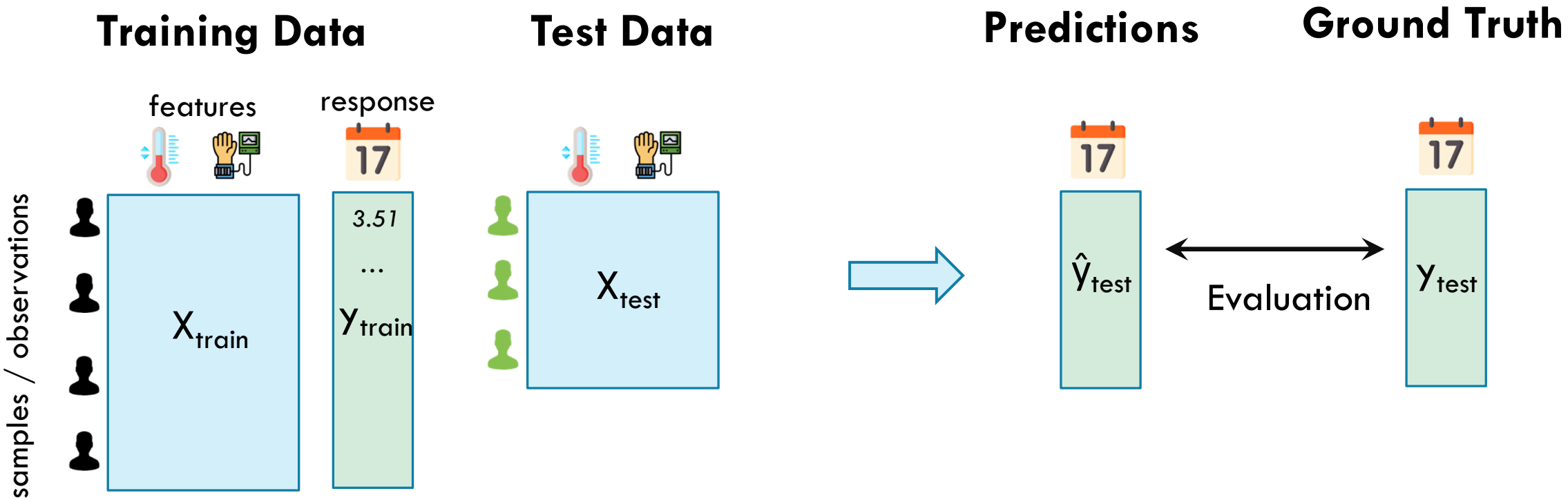
Categorize samples based on a **discrete** label, given training data

CLASSIFICATION: SETUP



Categorize samples based on a **discrete** label, given training data

REGRESSION: SETUP

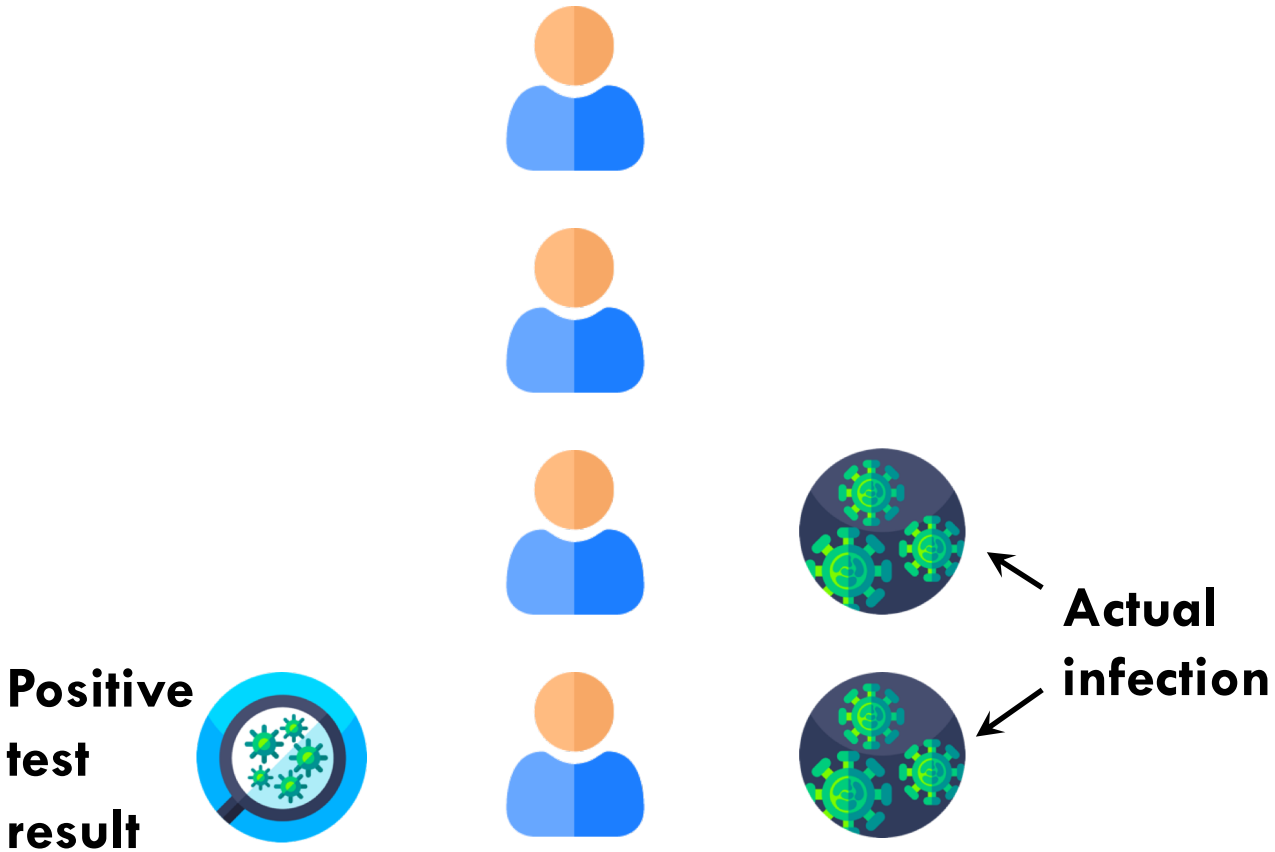


Predict a **numeric** response variable, given training data

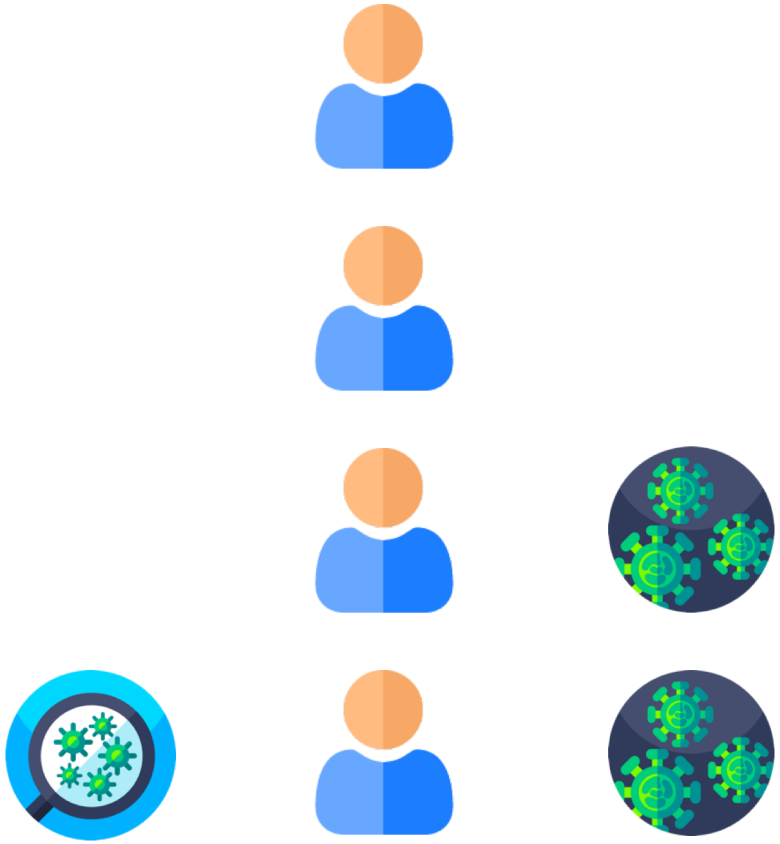
OVERVIEW

1. Problem Setup
2. Evaluating Classifiers
3. Nearest Neighbor Methods
4. Tree-Based Methods
5. Ensemble Methods
6. Logistic Regression
7. Deep Learning

BINARY CLASSIFICATION SETTING



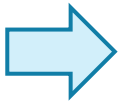
BINARY CLASSIFICATION SETTING



Predicted Label ($\hat{y}$)	Ground Truth Label (y)
0	0
0	0
0	1
1	1

CONFUSION MATRIX

Predicted Label ($\hat{y}$)	Ground Truth Label (y)
0	0
0	0
0	1
1	1



Ground Truth Label (y)

Predicted Label ($\hat{y}$)

	0	1
0	2	1
1	0	1

FALSE POSITIVES, FALSE NEGATIVES, ETC

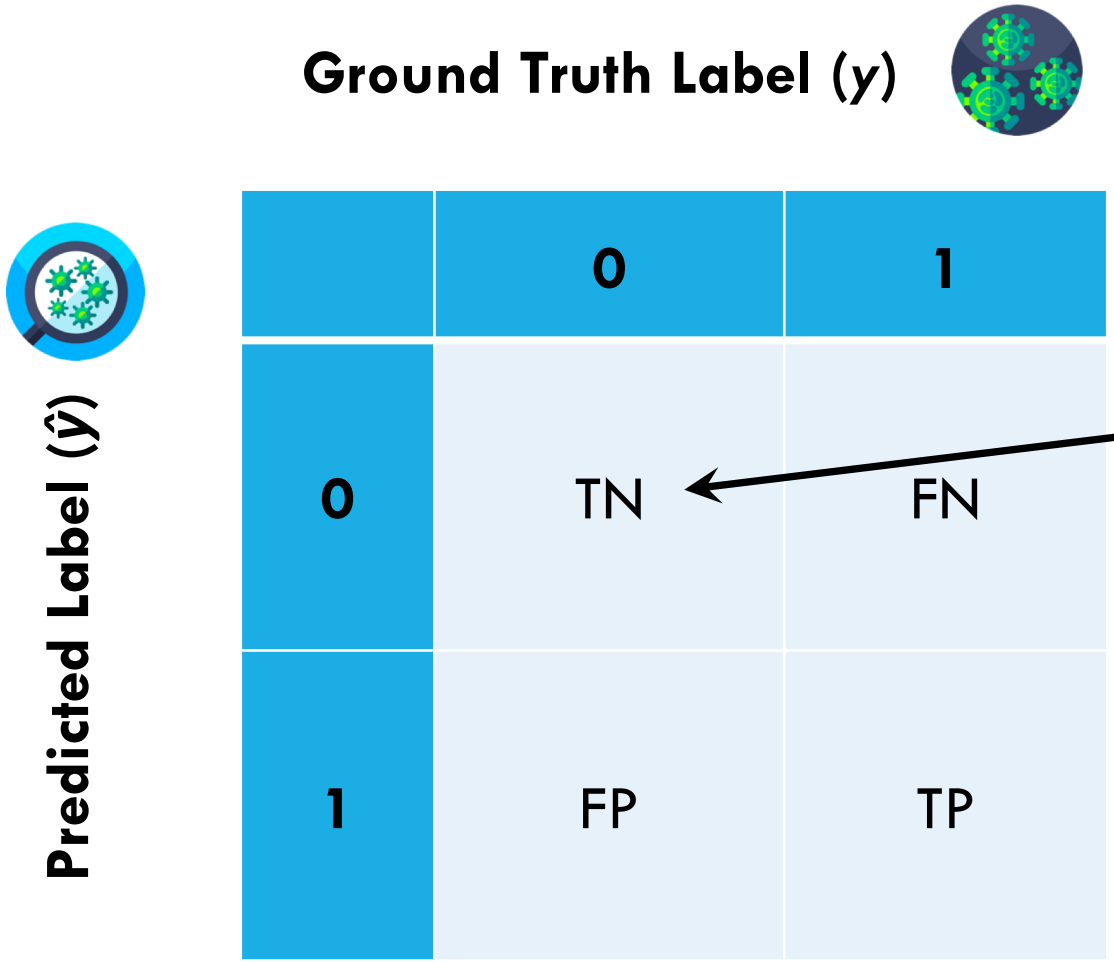
Ground Truth Label (*y*)

Predicted Label ($\hat{y}$)

	0	**1**
0	TN	FN
1	FP	TP

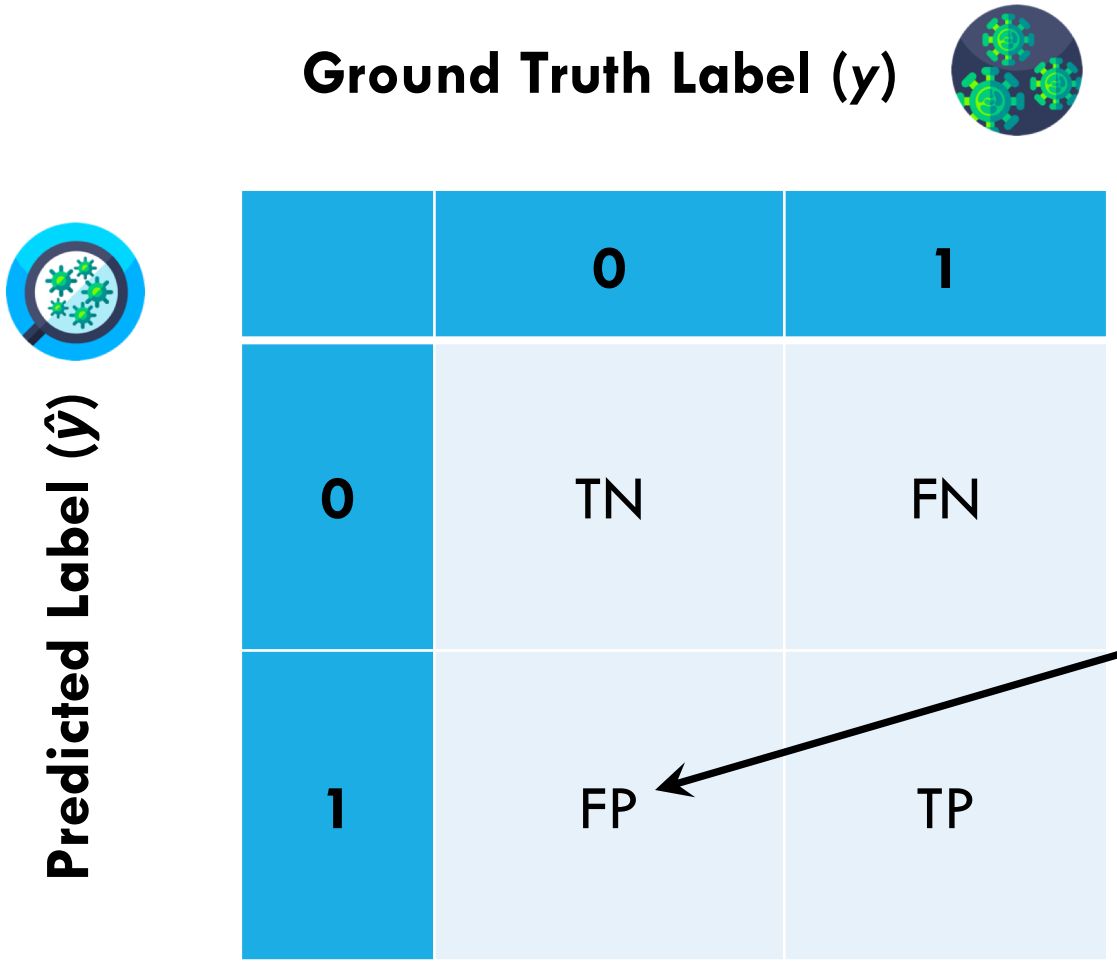
True Positive: test correctly output positive

FALSE POSITIVES, FALSE NEGATIVES, ETC



- **True Positive:** test correctly output positive
- **True Negative:** test correctly outputs negative

FALSE POSITIVES, FALSE NEGATIVES, ETC



- **True Positive:** test correctly output positive
- **True Negative:** test correctly outputs negative
- **False Positive:** test wrongly outputs positive

FALSE POSITIVES, FALSE NEGATIVES, ETC

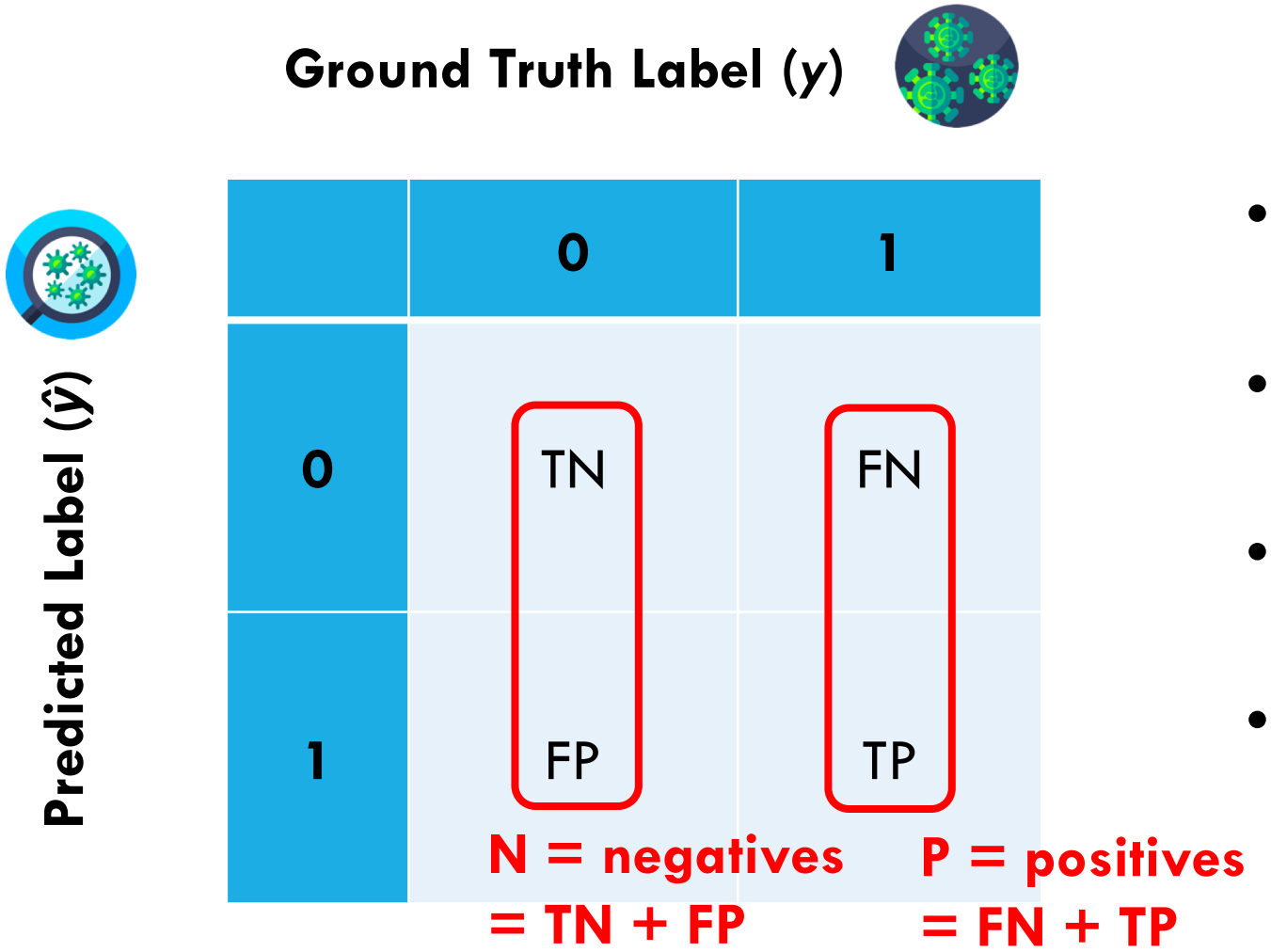
Ground Truth Label (*y*) 

Predicted Label ($\hat{y}$)

	0	**1**
0	TN	FN
1	FP	TP

- **True Positive:** test correctly output positive
- **True Negative:** test correctly outputs negative
- **False Positive:** test wrongly outputs positive
- **False Negative:** test wrongly outputs negative

FALSE POSITIVES, FALSE NEGATIVES, ETC



- **True Positive:** test correctly output positive
- **True Negative:** test correctly outputs negative
- **False Positive:** test wrongly outputs positive
- **False Negative:** test wrongly outputs negative

QUIZ: FALSE POSITIVES & NEGATIVES

Q: Suppose a positive test almost always correctly indicates the disease, but the test often misses the disease. How would you describe the (proportion of) false positives and negatives?

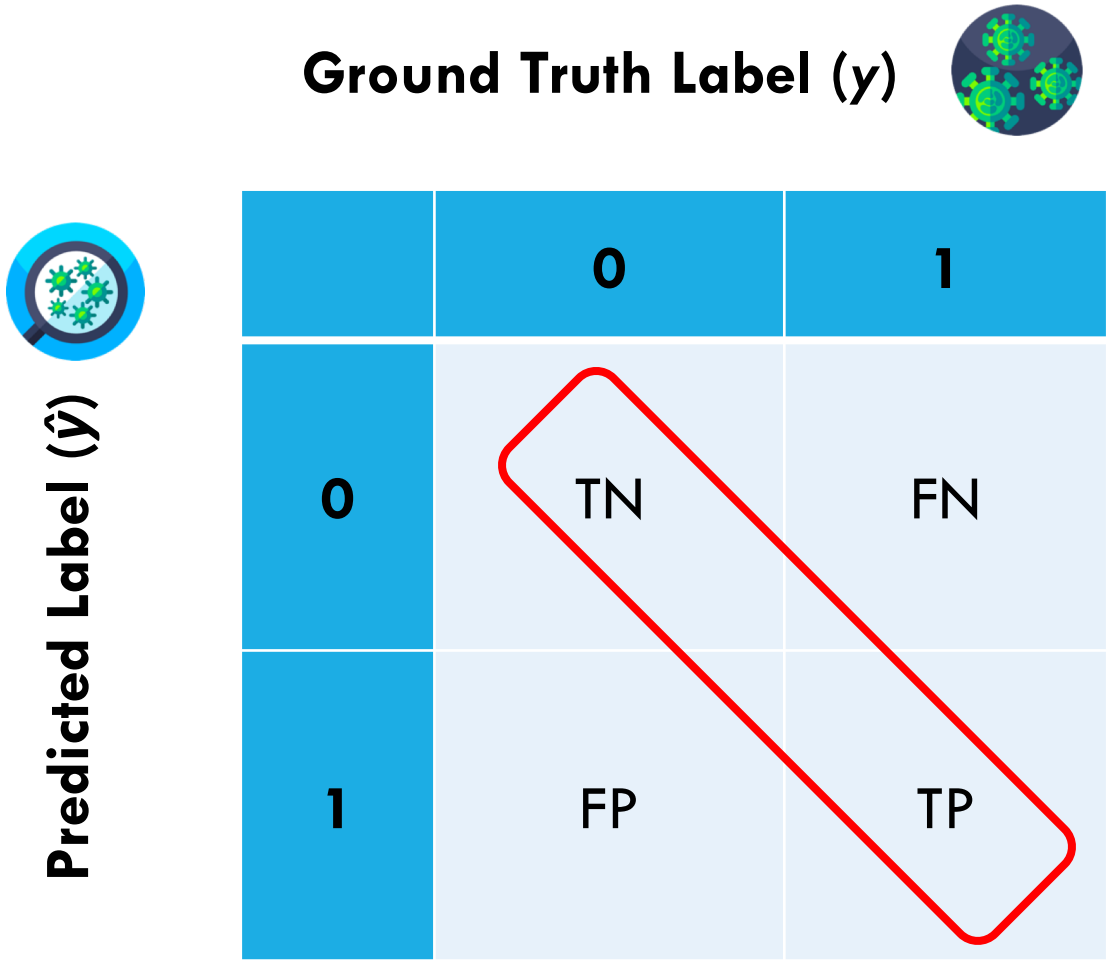
1. High false positives, high false negatives
2. High false positives, low false negatives
3. Low false positives, high false negatives
4. Low false positives, low false negatives

QUIZ: FALSE POSITIVES & NEGATIVES

Q: Suppose a positive test almost always correctly indicates the disease, but the test often misses the disease. How would you describe the (proportion of) false positives and negatives?

1. High false positives, high false negatives
2. High false positives, low false negatives
3. Low false positives, high false negatives
4. Low false positives, low false negatives

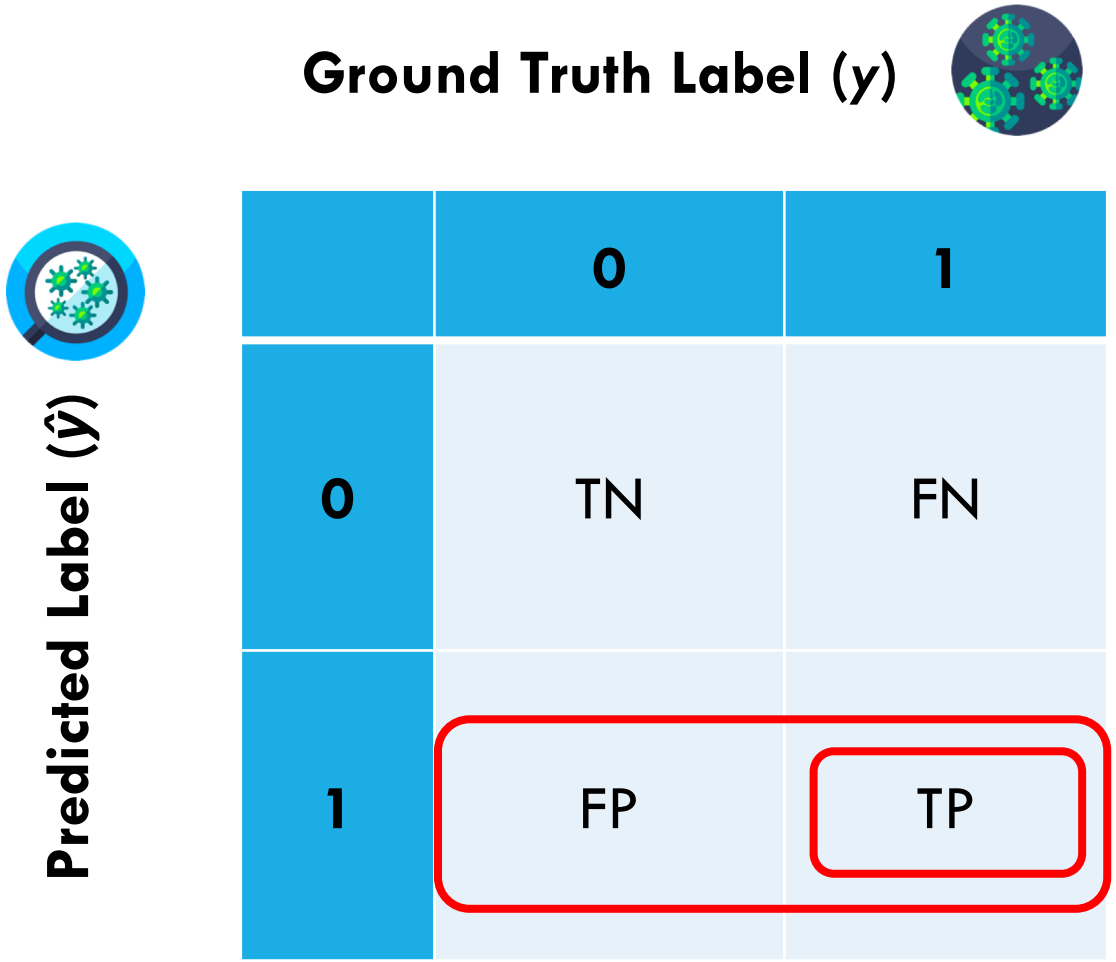
ACCURACY



Accuracy: fraction of predictions that are correct

= (TN + TP) / (TN + FN + FP + TP)

PRECISION

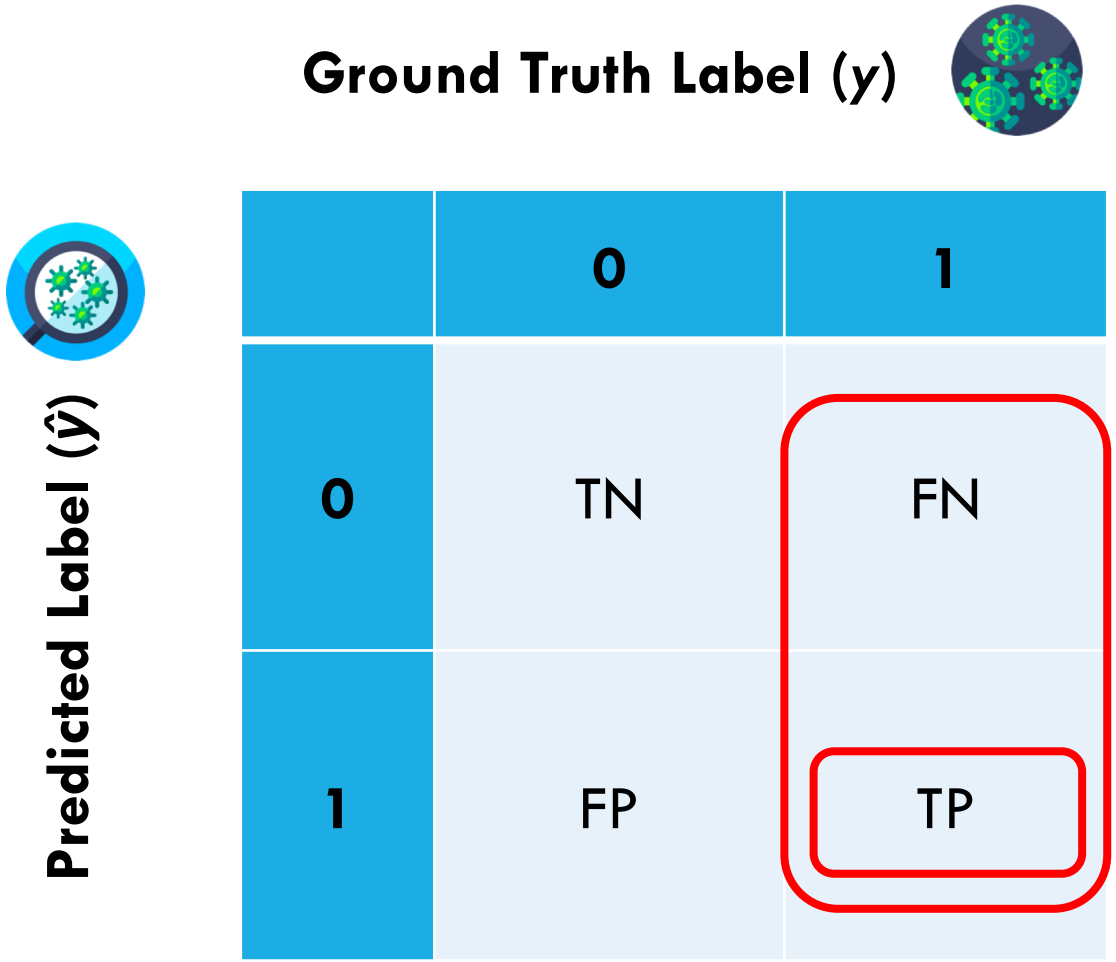


Precision: fraction of positive test results that are correct

= TP / (TP + FP)

Interpretation: if the test returns 1, how likely is the patient to be actually sick?

RECALL

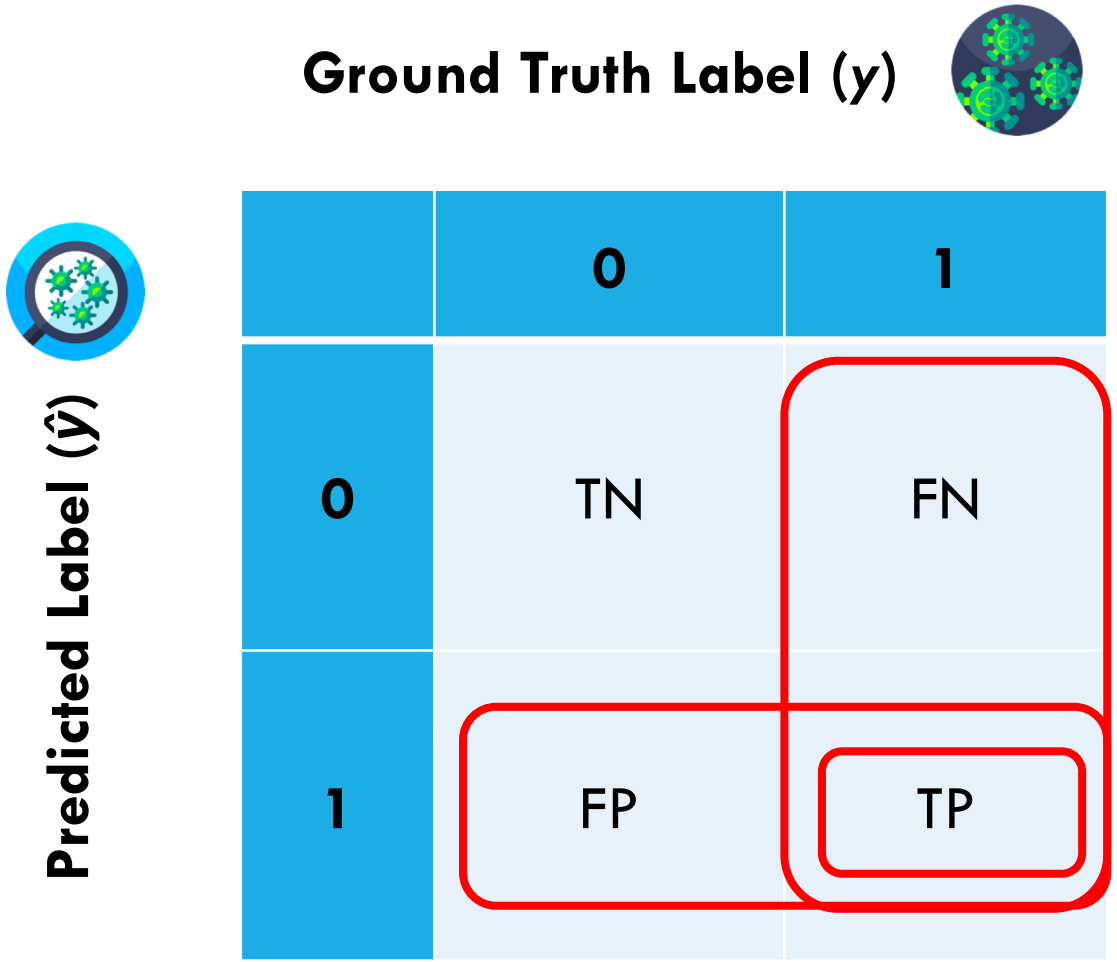


Recall: fraction of actual positives that are detected

= TP / (TP + FN)

Interpretation: if a patient is sick, how likely is the test to detect this?

F-MEASURE / F1 SCORE



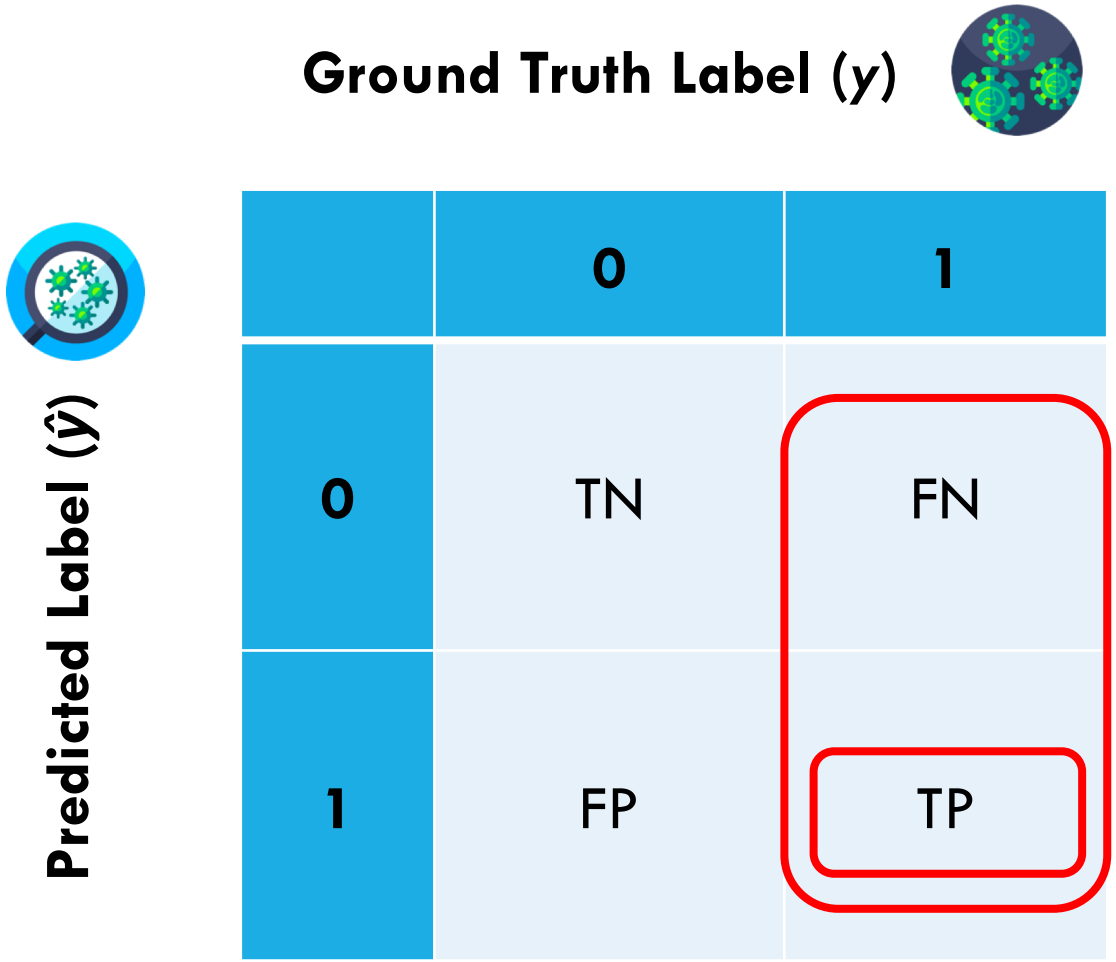
F-Measure:

$$= 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Interpretation: it is a 'pessimistic average' of precision and recall:

- I.e. F-measure is always between precision and recall, and closer to whichever of them is lower
- E.g. if precision = 1, recall = 0.1, then F1=0.18
- Formally, F-measure is the **harmonic mean** of precision and recall

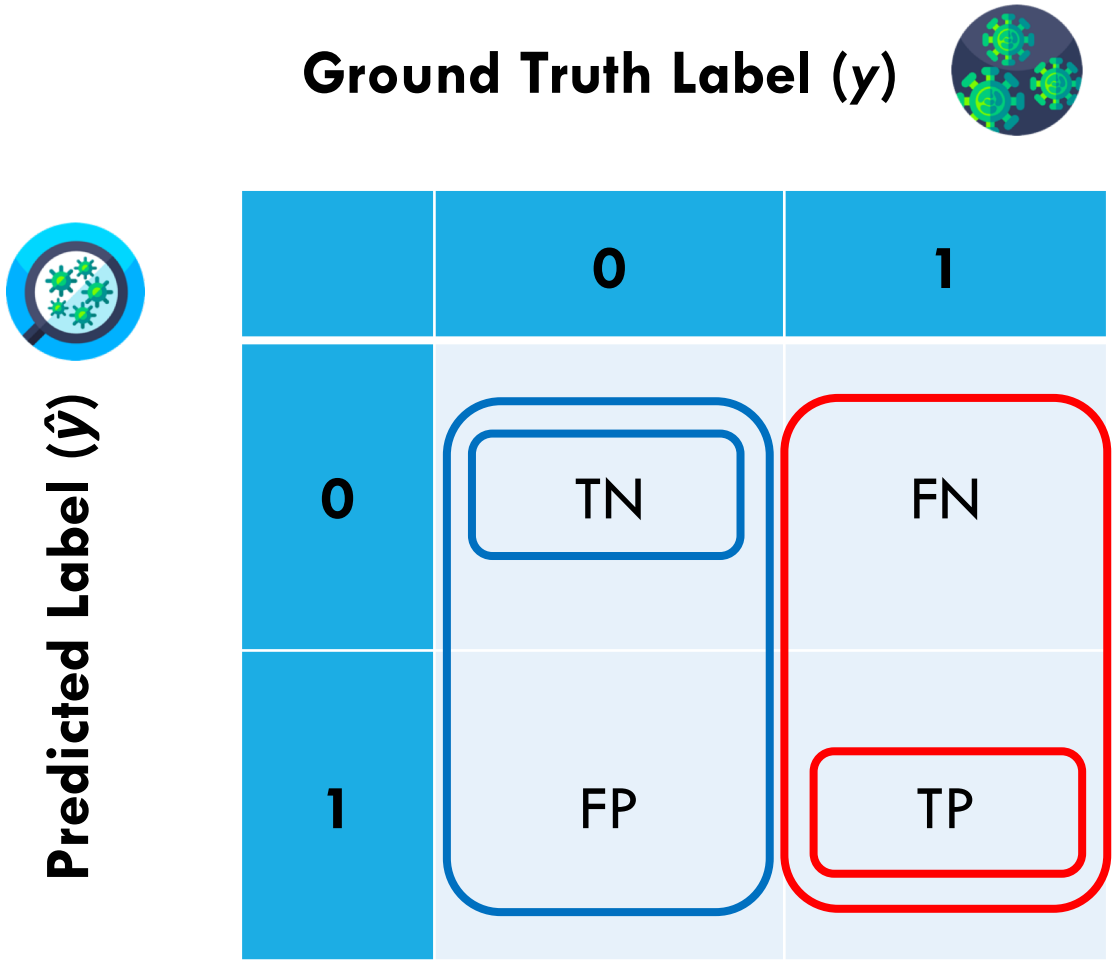
SENSITIVITY AND SPECIFICITY



Sensitivity: same as recall (fraction of actual positives that are correctly identified)

= TP / (TP + FN)

SENSITIVITY AND SPECIFICITY



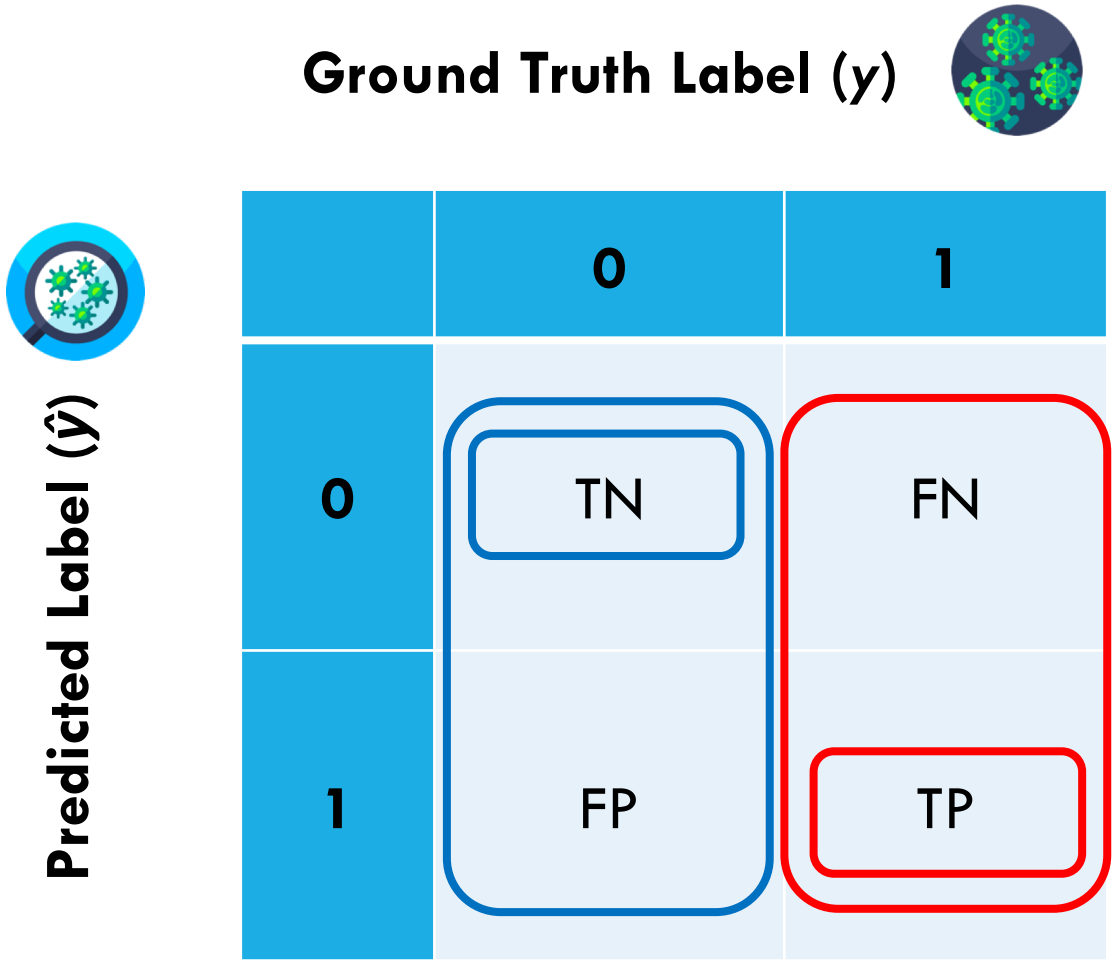
Sensitivity: same as recall (fraction of actual positives that are correctly identified)

= TP / (TP + FN)

Specificity: fraction of actual negatives that are correctly identified

= TN / (TN + FP)

SENSITIVITY AND SPECIFICITY



Sensitivity: same as recall (fraction of actual positives that are correctly identified)

= TP / (TP + FN)

Specificity: fraction of actual negatives that are correctly identified

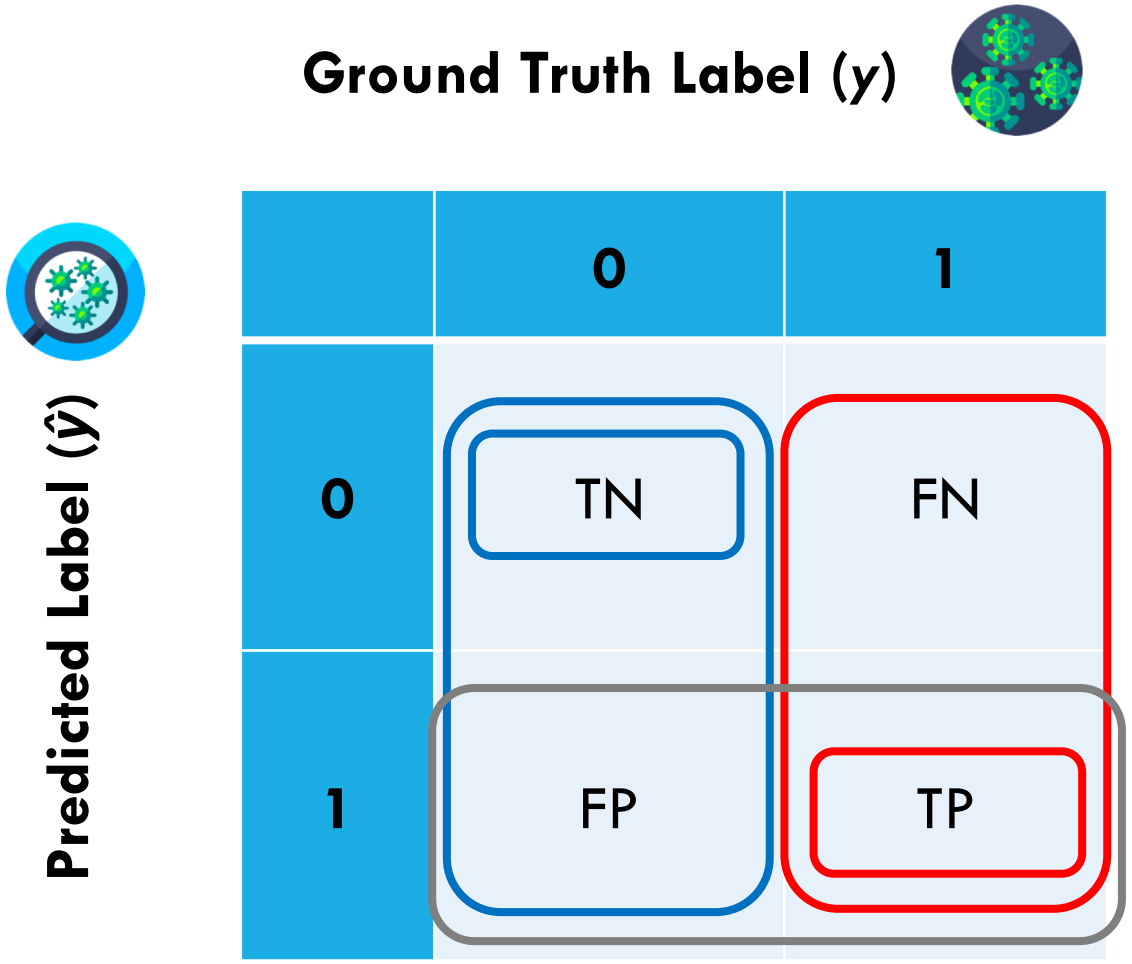
= TN / (TN + FP)

Interpretation:

Sensitivity = accuracy among positives (y=1)

Specificity = accuracy among negatives (y=0)

SENSITIVITY AND SPECIFICITY



Sensitivity: same as recall (fraction of actual positives that are correctly identified)

= TP / (TP + FN)

Specificity: fraction of actual negatives that are correctly identified

= TN / (TN + FP)

Interpretation:

Sensitivity = accuracy among positives (y=1)

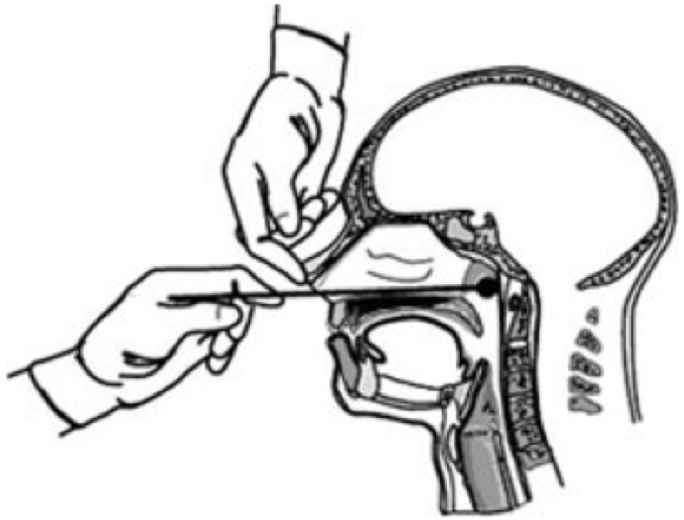
Specificity = accuracy among negatives (y=0)

Precision = accuracy among detected (ŷ=1)

QUIZ: SENSITIVITY & SPECIFICITY

Q: RT-PCR is a common test for COVID-19. A study [1] found that 78.2% of actual COVID-19 cases are correctly detected, while among patients without COVID-19, positive tests occur only 1.2% of the time. What is the recall (or sensitivity)?

1. 0.782
2. 0.218
3. 0.012
4. 0.988

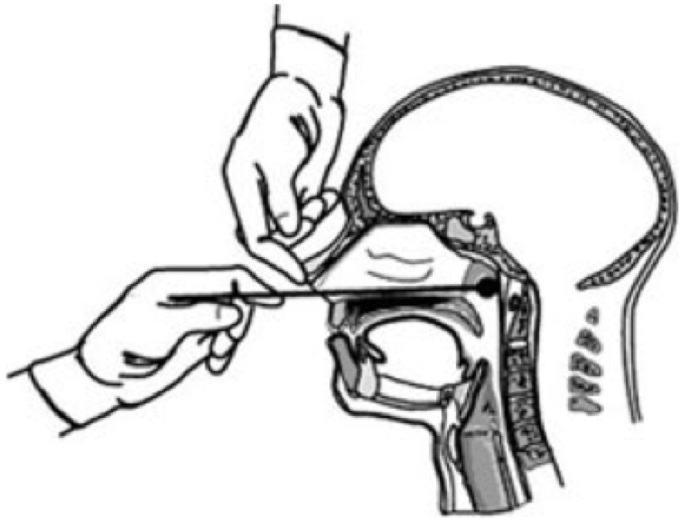


[1] Ren, et al. "Application and optimization of RT-PCR in diagnosis of SARS-CoV-2 infection.".

QUIZ: SENSITIVITY & SPECIFICITY

Q: RT-PCR is a common test for COVID-19. A study [1] found that 78.2% of actual COVID-19 cases are correctly detected, while among patients without COVID-19, positive tests occur only 1.2% of the time. What is the recall (or sensitivity)?

1. 0.782
2. 0.218
3. 0.012
4. 0.988

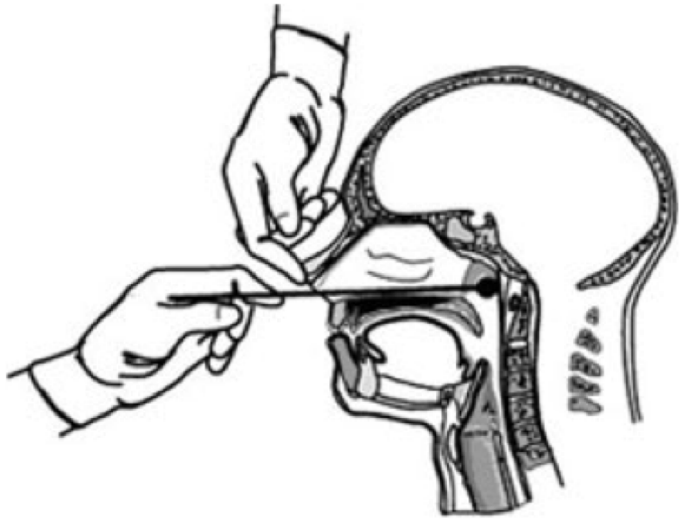


[1] Ren, et al. "Application and optimization of RT-PCR in diagnosis of SARS-CoV-2 infection.".

QUIZ: SENSITIVITY & SPECIFICITY

Q: RT-PCR is a common test for COVID-19. A study [1] found that 78.2% of actual COVID-19 cases are correctly detected, while among patients without COVID-19, positive tests occur only 1.2% of the time. What is the specificity?

1. 0.782
2. 0.218
3. 0.012
4. 0.988

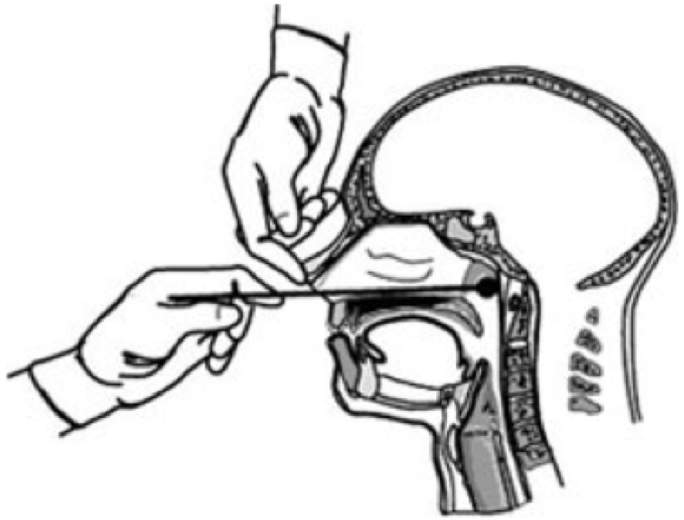


[1] Ren, et al. "Application and optimization of RT-PCR in diagnosis of SARS-CoV-2 infection.".

QUIZ: SENSITIVITY & SPECIFICITY

Q: RT-PCR is a common test for COVID-19. A study [1] found that 78.2% of actual COVID-19 cases are correctly detected, while among patients without COVID-19, positive tests occur only 1.2% of the time. What is the specificity?

1. 0.782
2. 0.218
3. 0.012
4. 0.988



[1] Ren, et al. "Application and optimization of RT-PCR in diagnosis of SARS-CoV-2 infection.".

PROBLEM: HIGHLY UNBALANCED DATA?

Predicted Label ($\hat{y}$)

Ground Truth Label (y) 

	0	**1**
0	1000	20
1	0	0

In real data, **highly unbalanced data** is common: e.g. most people don't have any given disease.

PROBLEM: HIGHLY UNBALANCED DATA?

Ground Truth Label (*y*)

Predicted Label ($\hat{y}$)

	0	**1**
0	1000	20
1	0	0

In real data, **highly unbalanced data** is common: e.g. most people don't have any given disease.

In the example on the left, how would you describe the accuracy and the specificity?

1. High accuracy, high specificity
2. High accuracy, low specificity
3. Low accuracy, high specificity
4. Low accuracy, low specificity

PROBLEM: HIGHLY UNBALANCED DATA?

Ground Truth Label (*y*)

Predicted Label ($\hat{y}$)

	0	**1**
0	1000	20
1	0	0

In real data, **highly unbalanced data** is common: e.g. most people don't have any given disease.

In the example on the left, how would you describe the accuracy and the specificity?

1. High accuracy, high specificity
2. High accuracy, low specificity
3. Low accuracy, high specificity
4. Low accuracy, low specificity

PROBLEM: HIGHLY UNBALANCED DATA?

Ground Truth Label (*y*)

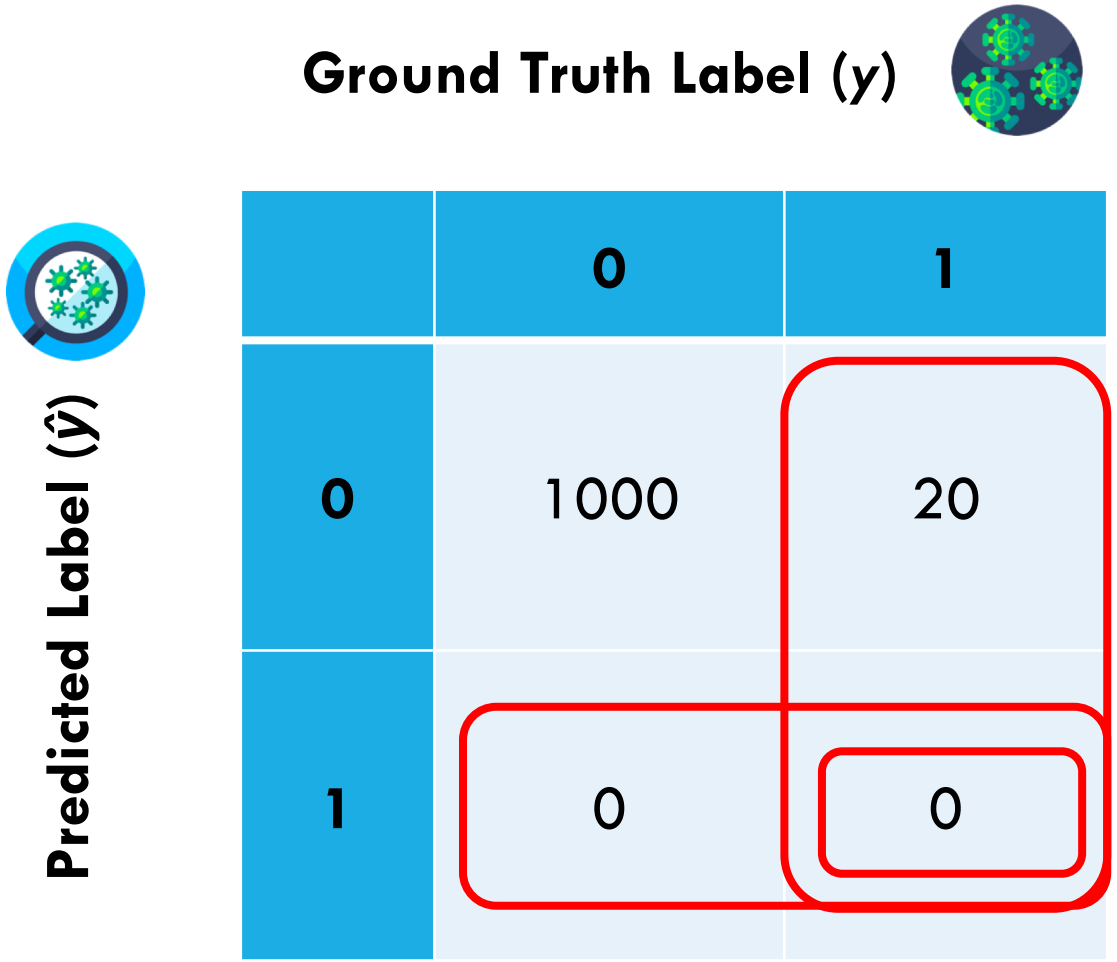
Predicted Label ($\hat{y}$)

	0	**1**
0	1000	20
1	0	0

In real data, **highly unbalanced data** is common: e.g. most people don't have any given disease.

The test on the left shows very **high accuracy and perfect specificity,** despite a test that always returns negative!

PROBLEM: HIGHLY UNBALANCED DATA?



Main problem is that the top-left entry can be extremely large (i.e. many true negatives)

Solution: for unbalanced data, use precision and recall, which ignore the true negatives

Use F-measure instead of accuracy

EVALUATING NUMERIC SCORES

In some cases, our "predictions" are in the form of numeric scores.

Examples:

- Tests may output a "risk score" instead of a 0 or 1 prediction
- Classification algorithms often output probabilities as their prediction; i.e. P(y=1)

Numeric prediction ($\hat{y}$)	Ground Truth Label (y)
0.05	0
0.1	0
0.6	1
0.9	1

THRESHOLDING

Given any threshold (e.g. 0.5), we can convert $\hat{y}$ to a binary variable by setting everything above the threshold to 1

Problem: hard to know what threshold to use

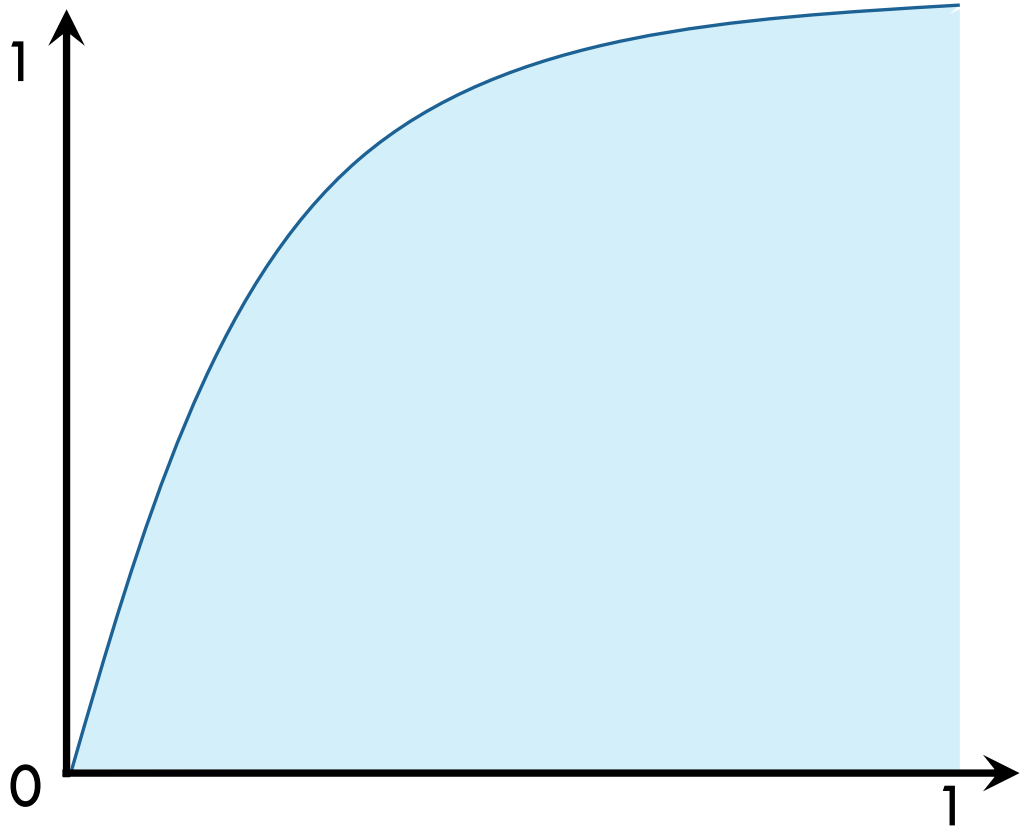
Solution: try all thresholds and plot the resulting curve!

Numeric prediction ($\hat{y}$)	Ground Truth Label (y)
0.05 → 0	0
0.1 → 0	0
0.6 → 1	1
0.9 → 1	1

threshold =0.5

RECEIVER-OPERATING CHARACTERISTIC (ROC) CURVE

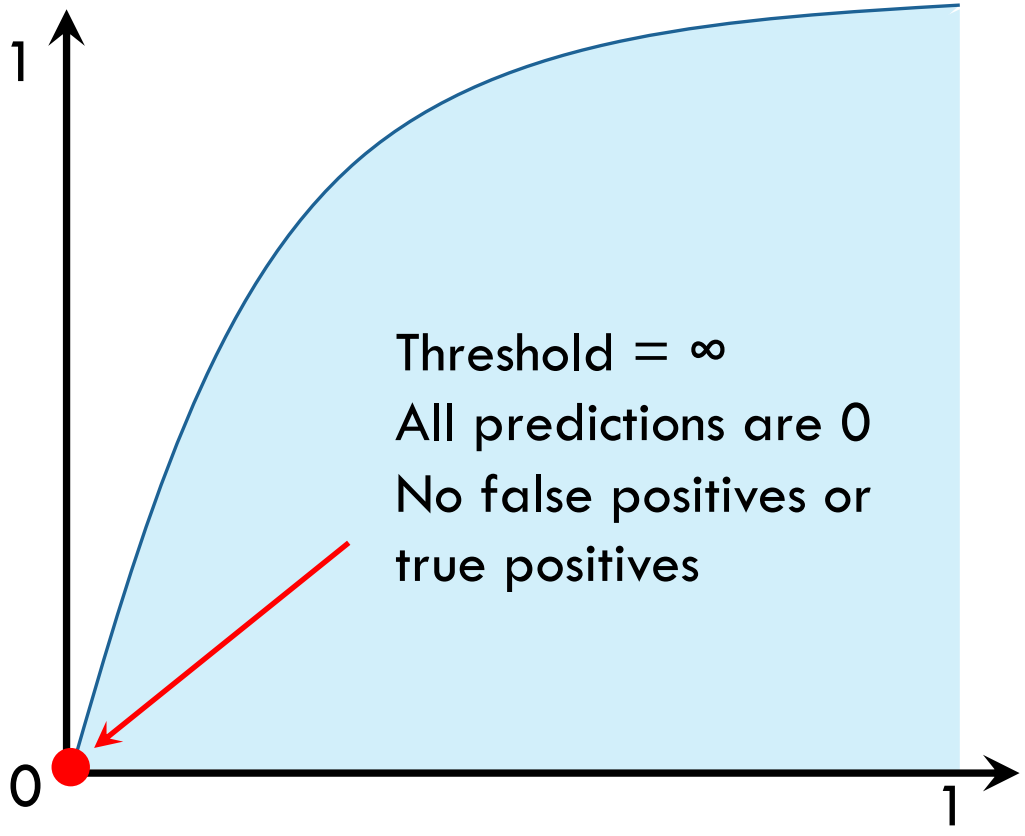
Sensitivity (i.e. True Positive Rate = TP / P)



1 – Specificity (i.e. False Positive Rate = FP / N)

RECEIVER-OPERATING CHARACTERISTIC (ROC) CURVE

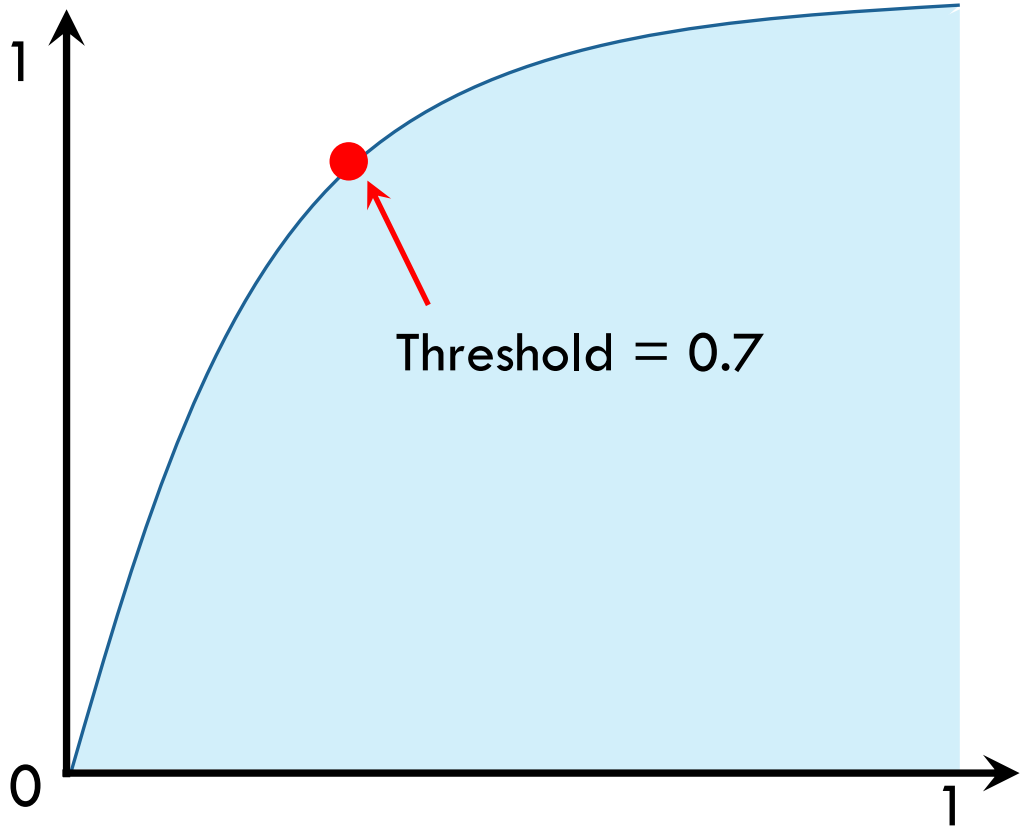
Sensitivity (i.e. True Positive Rate = TP / P)



1 – Specificity (i.e. False Positive Rate = FP / N)

RECEIVER-OPERATING CHARACTERISTIC (ROC) CURVE

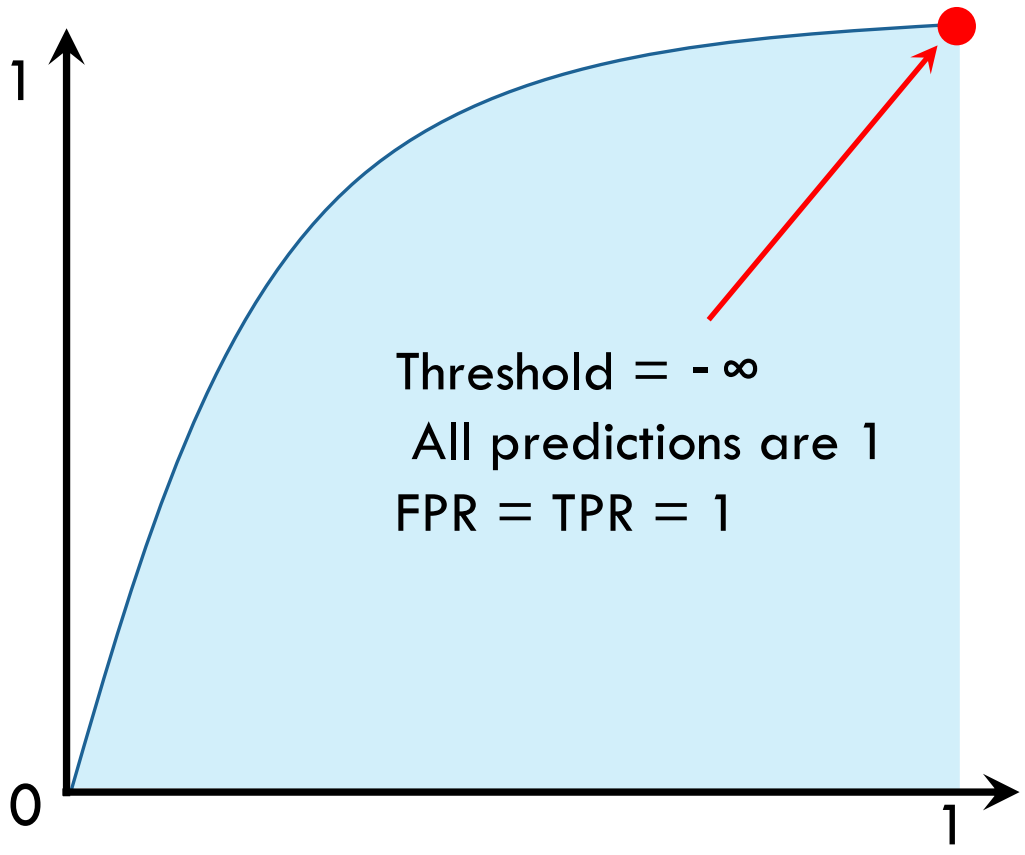
Sensitivity (i.e. True Positive Rate = TP / P)



1 – Specificity (i.e. False Positive Rate = FP / N)

RECEIVER-OPERATING CHARACTERISTIC (ROC) CURVE

Sensitivity (i.e. True Positive Rate = TP / P)

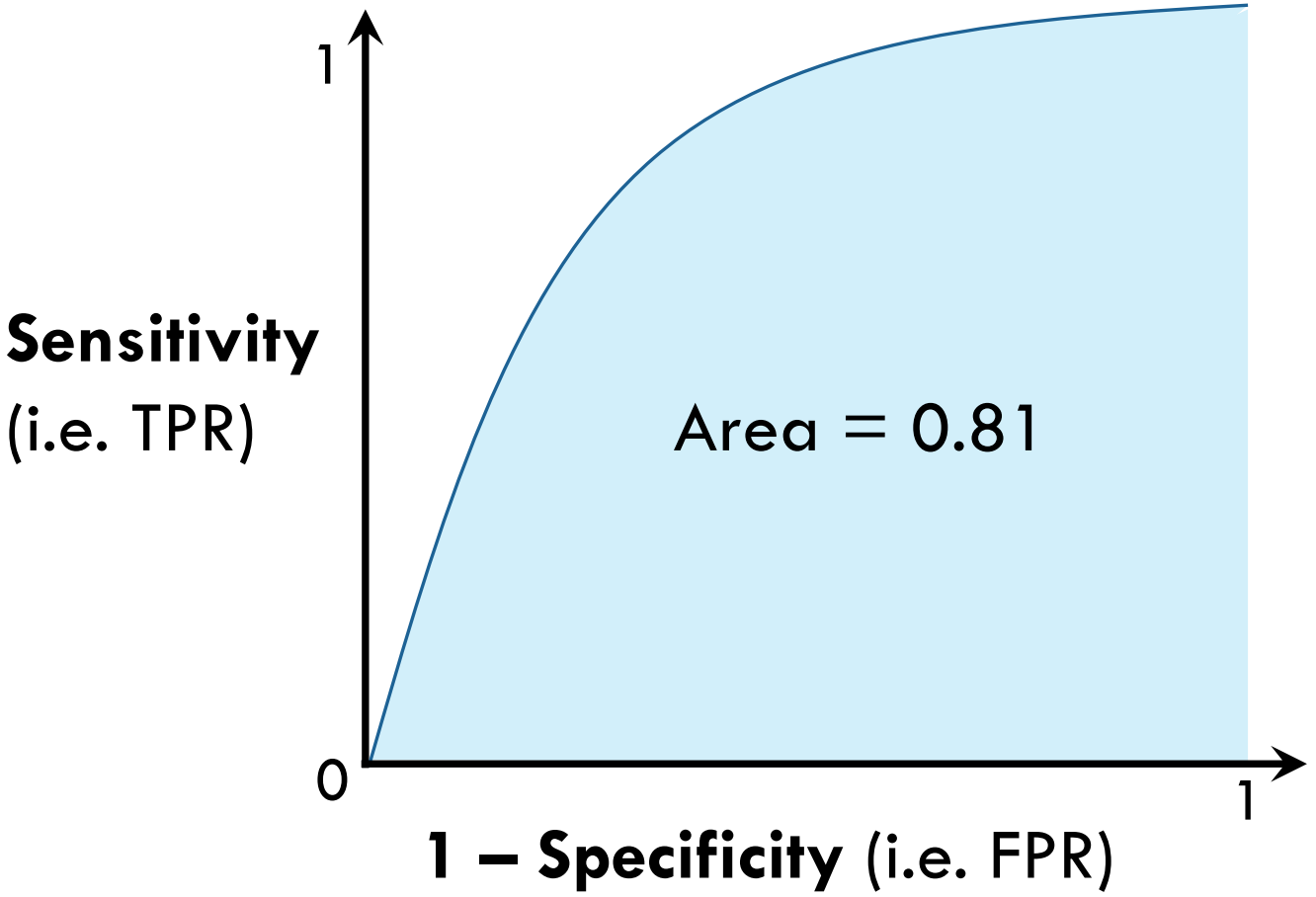


1 – Specificity (i.e. False Positive Rate = FP / N)

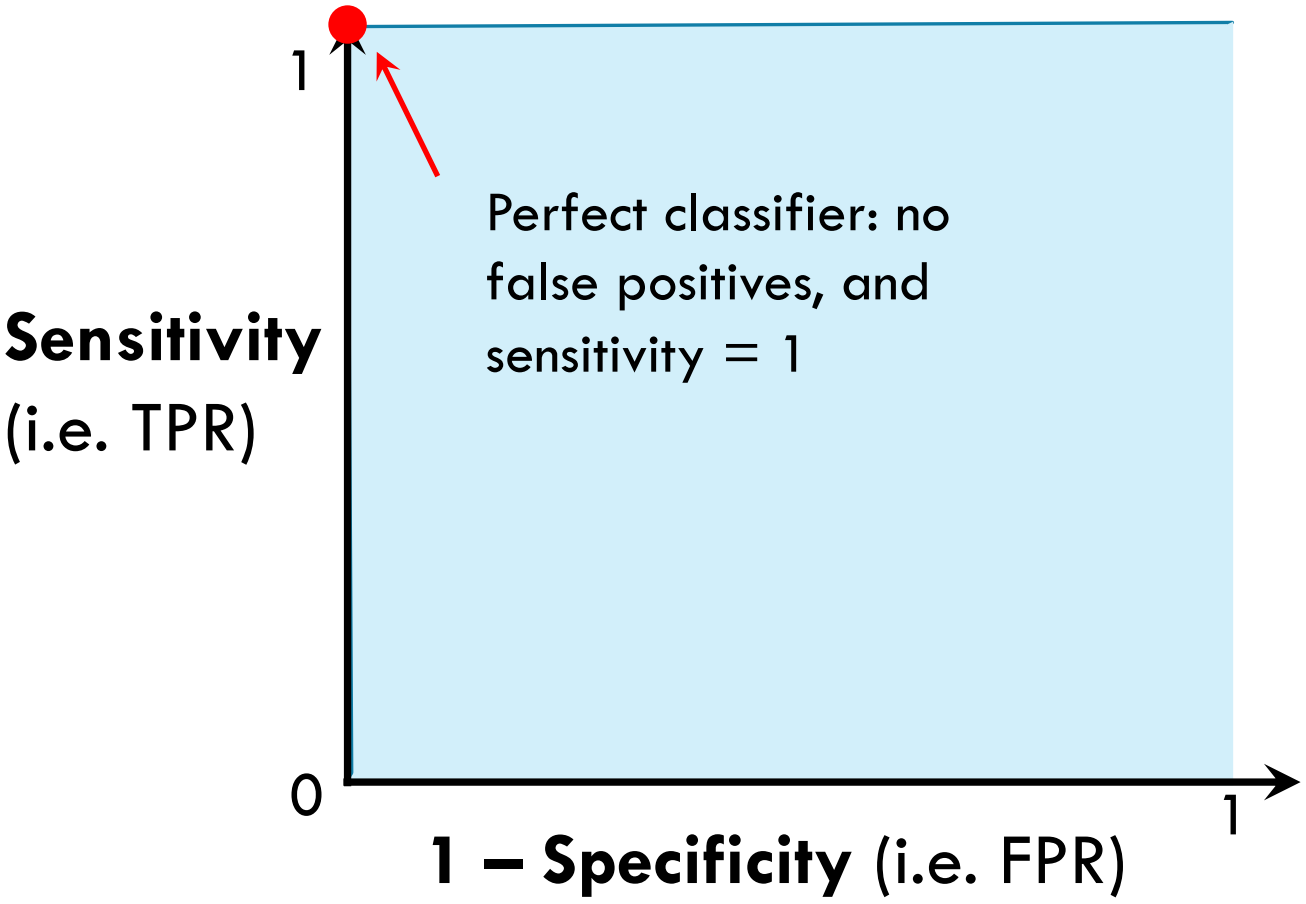
AREA UNDER RECEIVER-OPERATING CHARACTERISTIC (AUROC, OR AUC)

AUROC is defined as the **area under the ROC curve**

AUROC is between 0 and 1, with higher values meaning better detection



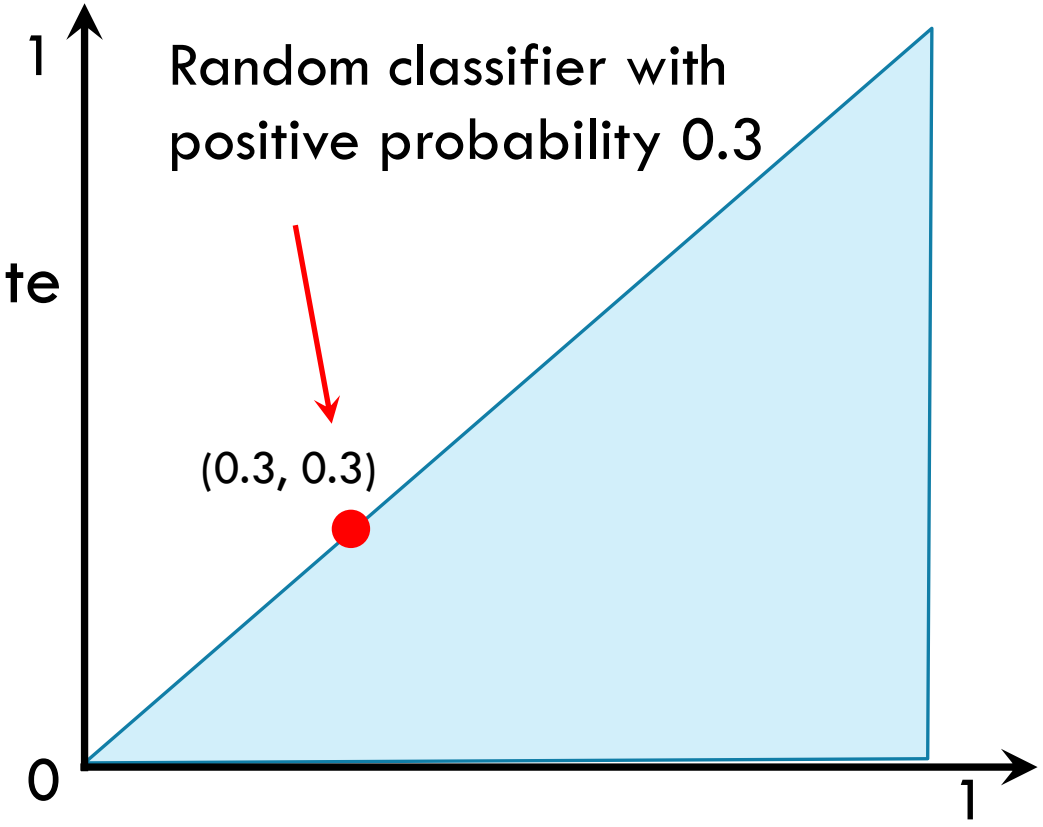
AUROC OF PERFECT CLASSIFIER = 1



Maximum AUROC is 1, indicating perfect classifier

AUROC OF RANDOM CLASSIFIERS

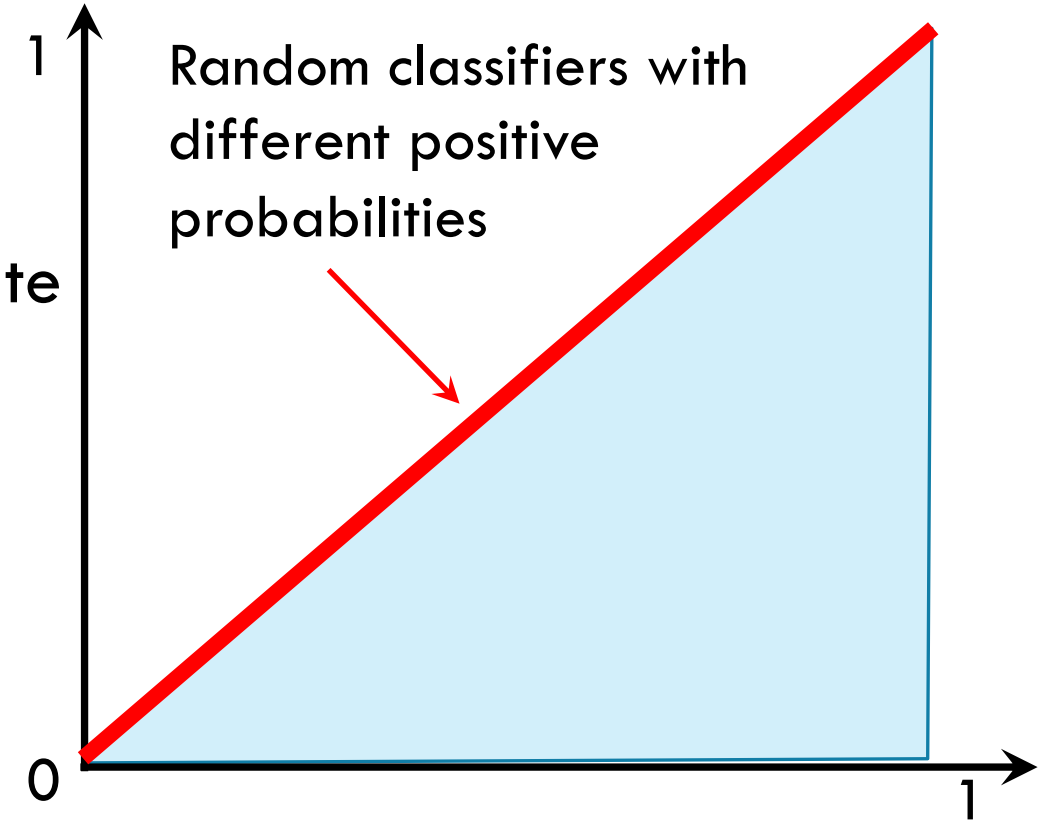
Sensitivity (i.e. True Positive Rate = TP / P)



1 – Specificity (i.e. False Positive Rate = FP / N)

AUROC OF RANDOM CLASSIFIERS

Sensitivity (i.e. True Positive Rate = TP / P)



⇒ AUROC of random classifiers is **0.5**

⇒ If a classifier has AUC below 0.5, it is doing worse than random

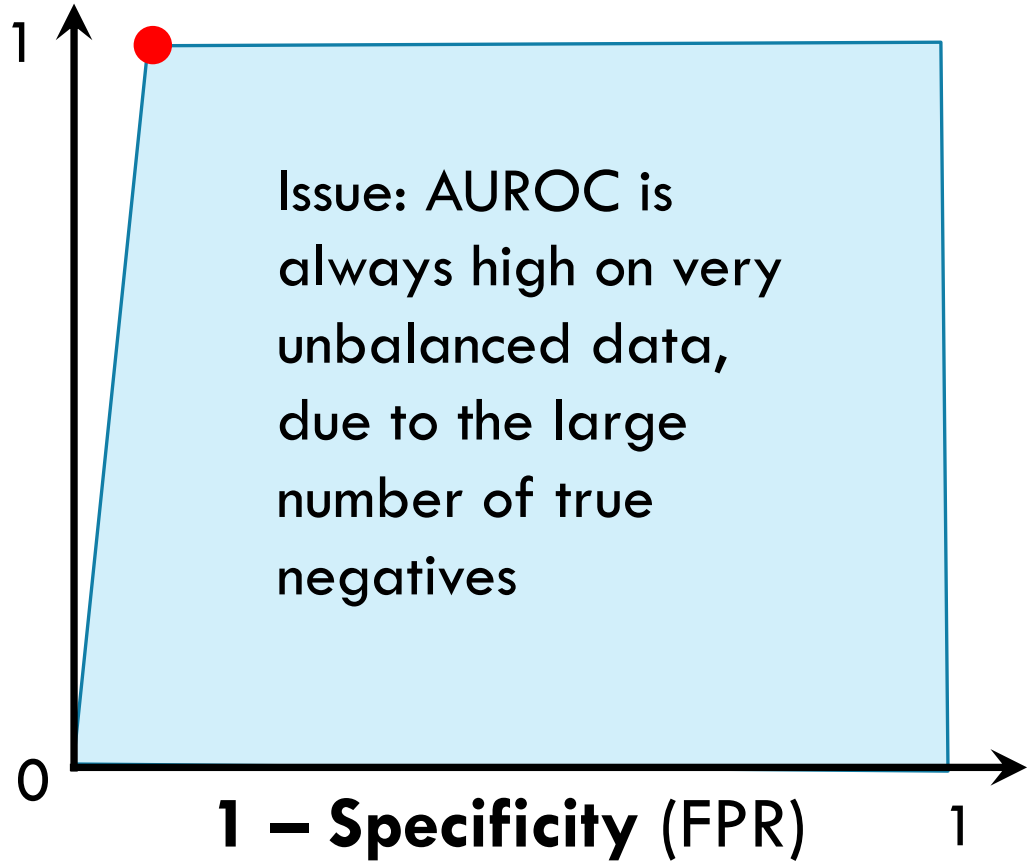
PROBLEM: AUROC ON HIGHLY UNBALANCED DATA

Predicted Label ($\hat{y}$)

Ground Truth Label (y)

	0	**1**
0	900	0
1	90	10

Sensitivity (TPR)

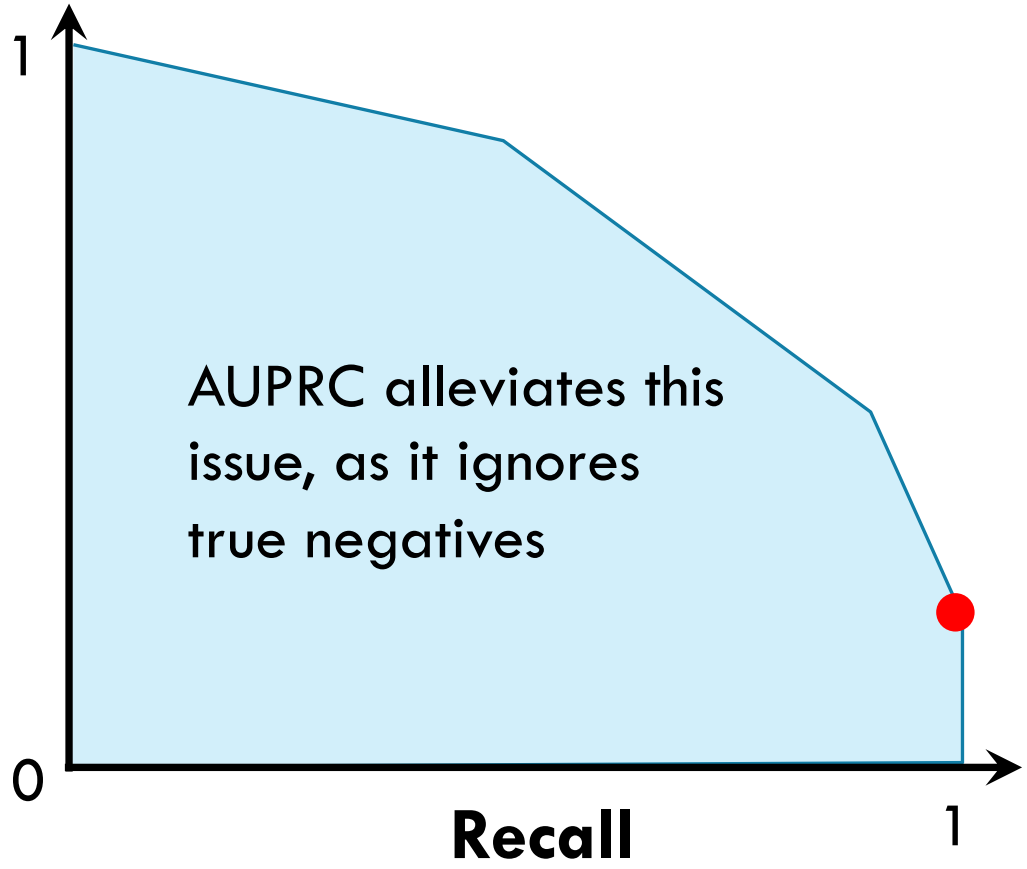


AREA UNDER PRECISION-RECALL CURVE (AUPRC)

Ground Truth Label (y)

Predicted Label ($\hat{y}$)

	0	**1**
0	900	0
1	90	10

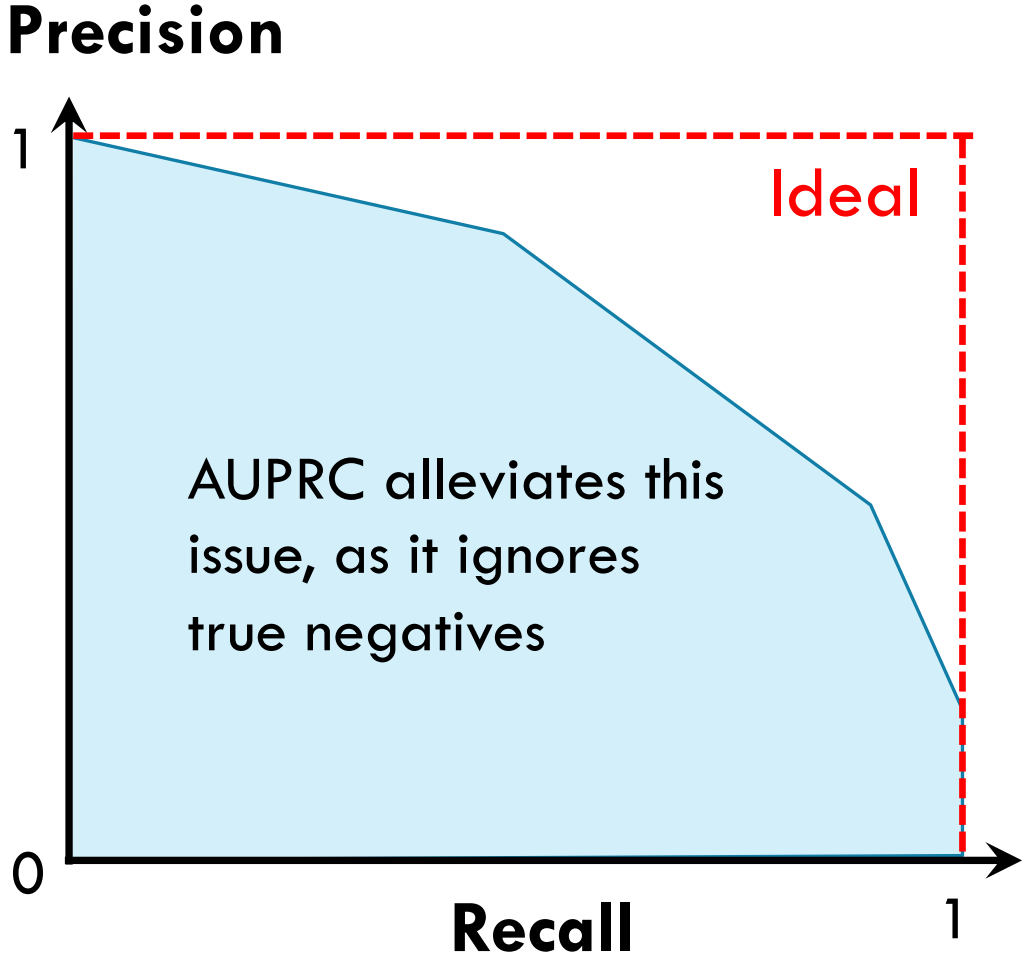


AREA UNDER PRECISION-RECALL CURVE (AUPRC)

Predicted Label ($\hat{y}$)

Ground Truth Label (y)

	0	**1**
0	900	0
1	90	10



MULTI-CLASS CLASSIFICATION EVALUATION

Ground Truth Label (*y*)

Predicted Label ($\hat{y}$)

	0	**1**	**2**
0	5	1	0
1	0	6	2
2	1	1	4

Total no. of samples

Accuracy

$= (5 + 6 + 4) / 20$

ONE VS REST CONFUSION MATRICES

Ground Truth Label (y)

Predicted Label ($\hat{y}$)

	0	1	2
0	5	1	0
1	0	6	2
2	1	1	4

0 vs. rest

0 vs. Rest

Ground Truth Label (y)

Predicted Label ($\hat{y}$)

	0	rest
0	5	1
rest	1	13

ONE VS REST CONFUSION MATRICES

Predicted Label ($\hat{y}$)

Ground Truth Label (y)

	0	1	2
0	5	1	0
1	0	6	2
2	1	1	4

0 vs. rest

	0	rest
0	5	1
rest	1	13

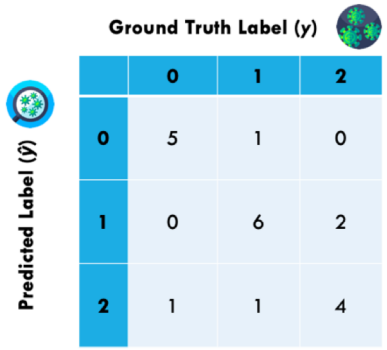
1 vs. rest

	1	rest
1	6	2
rest	2	10

2 vs. rest

	2	rest
2	4	2
rest	2	12

MICRO PRECISION / RECALL / F1



	0	1	2
0	5	1	0
1	0	6	2
2	1	1	4

0 vs. rest

	0	rest
0	TP_0	FP_0
rest	FN_0	TN_0

1 vs. rest

	1	rest
1	TP_1	FP_1
rest	FN_1	TN_1

2 vs. rest

	2	rest
2	TP_2	FP_2
rest	FN_2	TN_2

Micro-averaging:
Average over the *confusion matrix* values

MICRO PRECISION / RECALL / F1

Predicted Label (ŷ) \ Ground Truth Label (y)	0	1	2
0	5	1	0
1	0	6	2
2	1	1	4

0 vs. rest

	0	rest
0	TP_0	FP_0
rest	FN_0	TN_0

1 vs. rest

	1	rest
1	TP_1	FP_1
rest	FN_1	TN_1

2 vs. rest

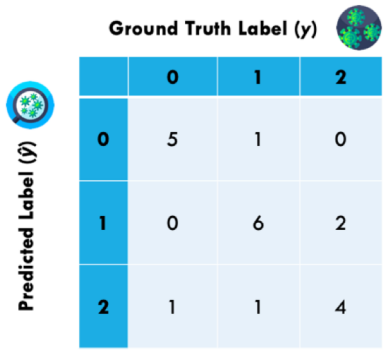
	2	rest
2	TP_2	FP_2
rest	FN_2	TN_2

Averaging

	-	-
-	TP_{ave}	FP_{ave}
-	FN_{ave}	TN_{ave}

Micro-averaging:
Average over the *confusion matrix* values

MICRO PRECISION / RECALL / F1



0 vs. rest

	0	rest
0	TP_0	FP_0
rest	FN_0	TN_0

1 vs. rest

	1	rest
1	TP_1	FP_1
rest	FN_1	TN_1

2 vs. rest

	2	rest
2	TP_2	FP_2
rest	FN_2	TN_2

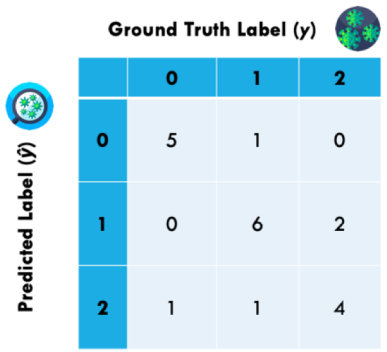
Averaging

	-	-
-	TP_{ave}	FP_{ave}
-	FN_{ave}	TN_{ave}

Micro-averaging:
Average over the *confusion matrix* values

$Precision_{micro}$
$Recall_{micro}$
$F1\text{-}Score_{micro}$

MACRO PRECISION / RECALL / F1



0 vs. rest

	0	rest
0	TP_0	FP_0
rest	FN_0	TN_0

$Precision_0$
$Recall_0$
$F1\text{-}Score_0$

1 vs. rest

	1	rest
1	TP_1	FP_1
rest	FN_1	TN_1

$Precision_1$
$Recall_1$
$F1\text{-}Score_1$

2 vs. rest

	2	rest
2	TP_2	FP_2
rest	FN_2	TN_2

$Precision_2$
$Recall_2$
$F1\text{-}Score_2$

Averaging

$Precision_{macro}$
$Recall_{macro}$
$F1\text{-}Score_{macro}$

Macro-averaging:
Average over the *performance metric* values

MICRO VS MACRO METRICS: SUMMARY

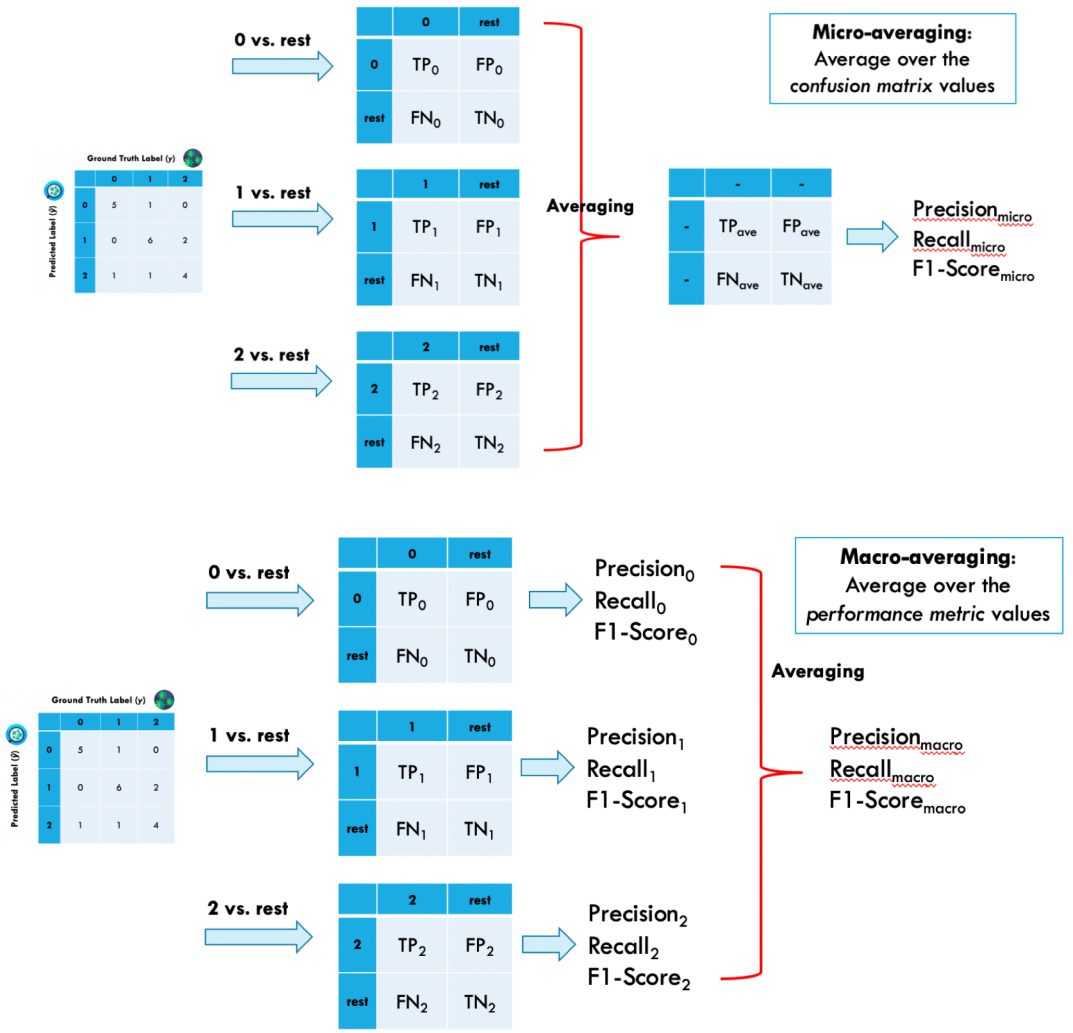
Both based on **one vs rest** confusion matrices

Micro: averaging takes place over *confusion matrix* values (i.e. TP, TN, FP, FN)

Macro: averaging takes place over *performance metric* values (i.e. Precision / Recall / F1)

Differences in behavior: macro-averaging treats all classes equally, while micro-averaging favors bigger classes

So, e.g. if we want to to prioritize all classes equally, we should use macro-averaging



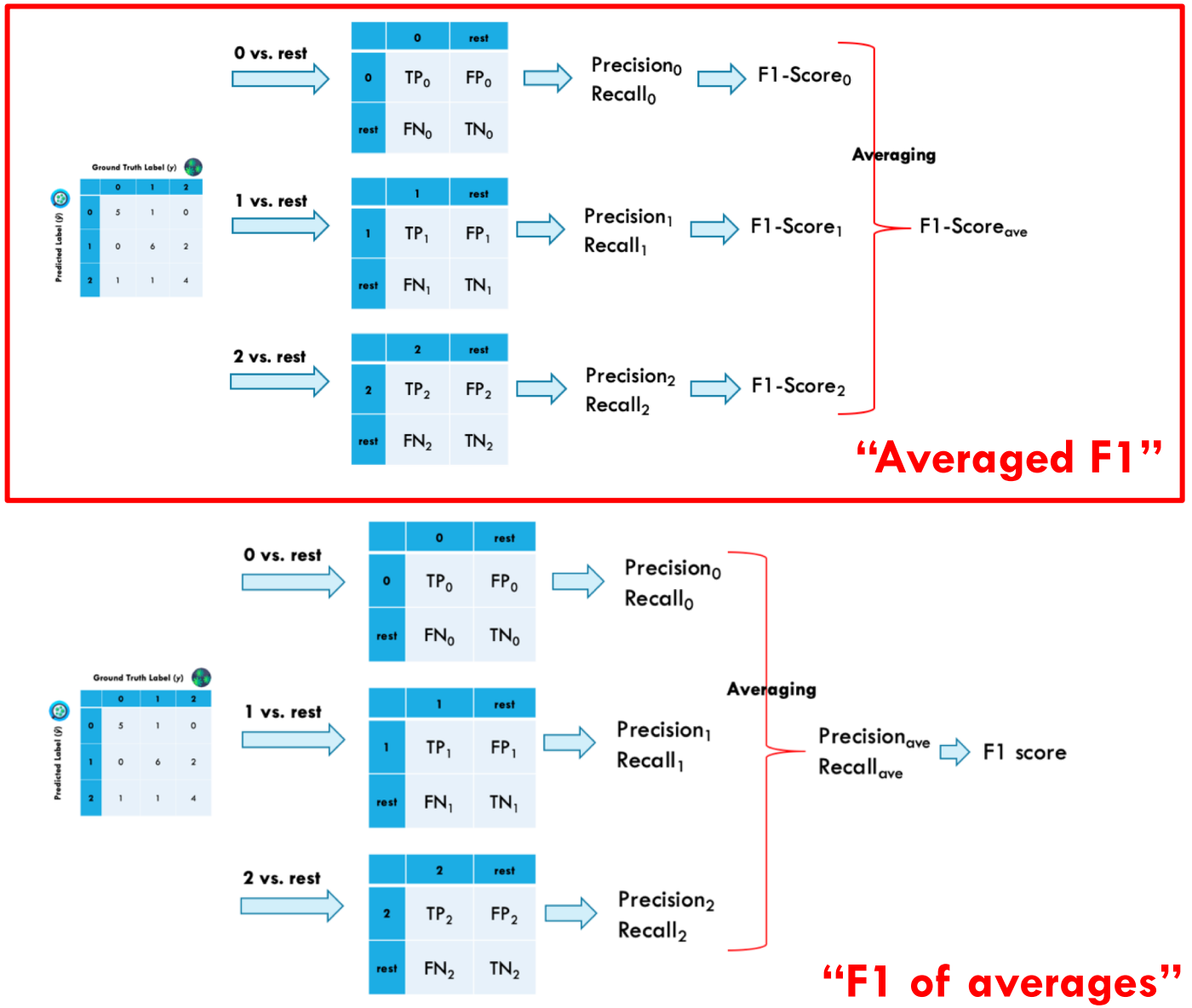
AMBIGUITY: TWO VERSIONS OF MACRO-F1

Optional

Averaged F1: Compute class specific F1 score; then average them to get macro-F1 score.

F1 of averages: Compute class-specific Precision / Recall, then average them to get Macro-Precision / Recall. Then macro-F1 is harmonic mean of Macro-Precision / Recall.

"History is written by the victors": `sklearn.metrics.f1_score` uses **Averaged F1**



Opitz, Juri, and Sebastian Burst. "Macro F1 and Macro F1." *arXiv* (2019).

DATA SPLITTING: TRAIN, VALIDATION, TEST

Typical splitting procedure: split the data into training, validation, and test sets

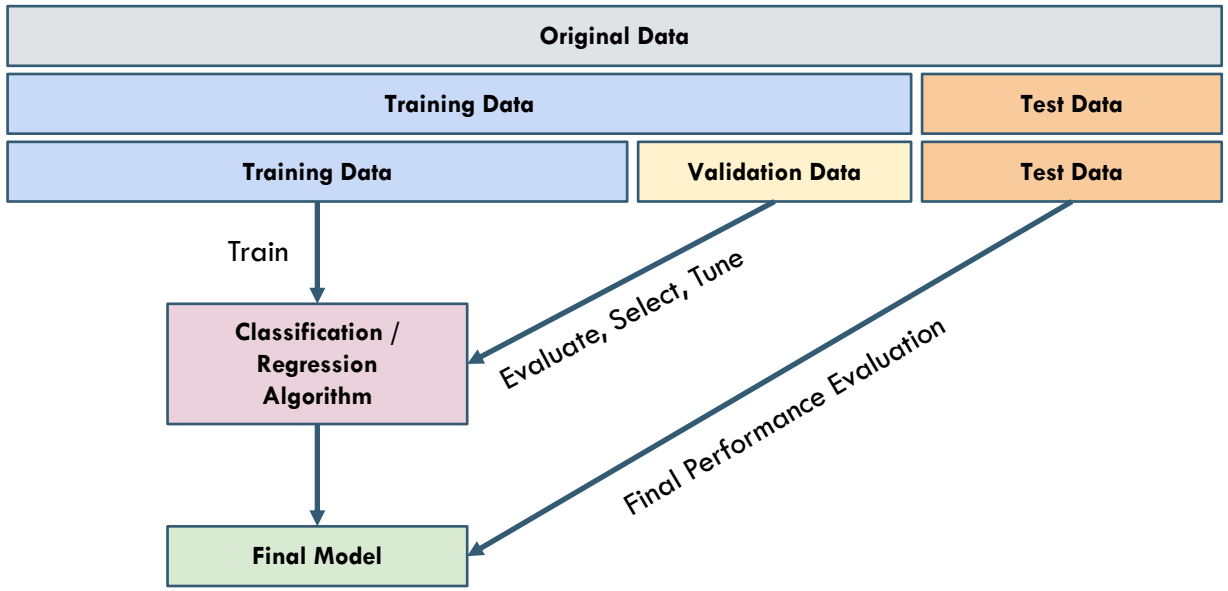
Use training data for training our classification / regression model

Use validation data for choosing which model to use or choosing between hyperparameters

- E.g.: to choose between Random Forests or k nearest neighbors, or to choose a hyperparameter k=5, 10, or 20: we try each such setting and then select the setting which leads to the highest validation accuracy

Use test data at the end to measure performance of the final model

(Proportions can vary, but a common choice is around 60% : 20% : 20%)

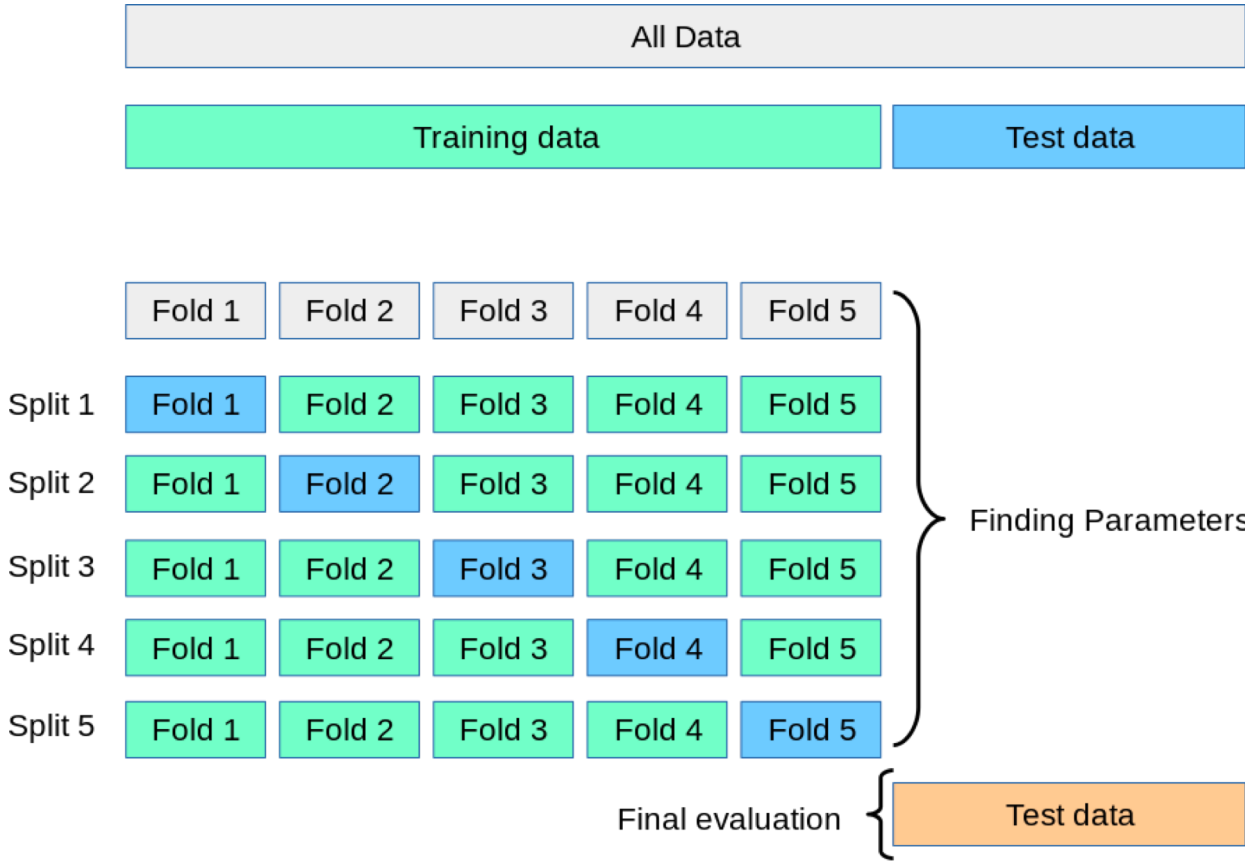


K-FOLD CROSS VALIDATION

Approach for hyperparameter / model selection: partition data into *folds*, and repeatedly train the data on all folds except 1, while evaluating the model on the held-out (validation) fold.

(Very similar to using a validation set, except that we average the results over the 5 choices of validation fold)

Finally use the test data at the end to measure performance



Credit: https://scikit-learn.org/stable/_images/grid_search_cross_validation.png

AVOIDING INFORMATION LEAKAGE

Important guideline:

- Do not normalize before splitting into training and test data
- Normalize training and test data but *only based on the statistics computed from training data*

```
# Split input data into training and test set
X_train, X_test = split(X, 0.2)

# Fit StandardScaler (i.e., calculate mean and variance)
scaler = preprocessing.StandardScaler().fit(X_train)  # CORRECT!
#scaler = preprocessing.StandardScaler().fit(X)       # WRONG!!!

# Fit both training and test data
X_train_transformed = scaler.transform(X_train)
X_test_transformed = scaler.transform(X_test)
```

Intuition: test performance is only a valid estimate of an algorithm's performance if the *algorithm's behavior is not influenced by the test data*

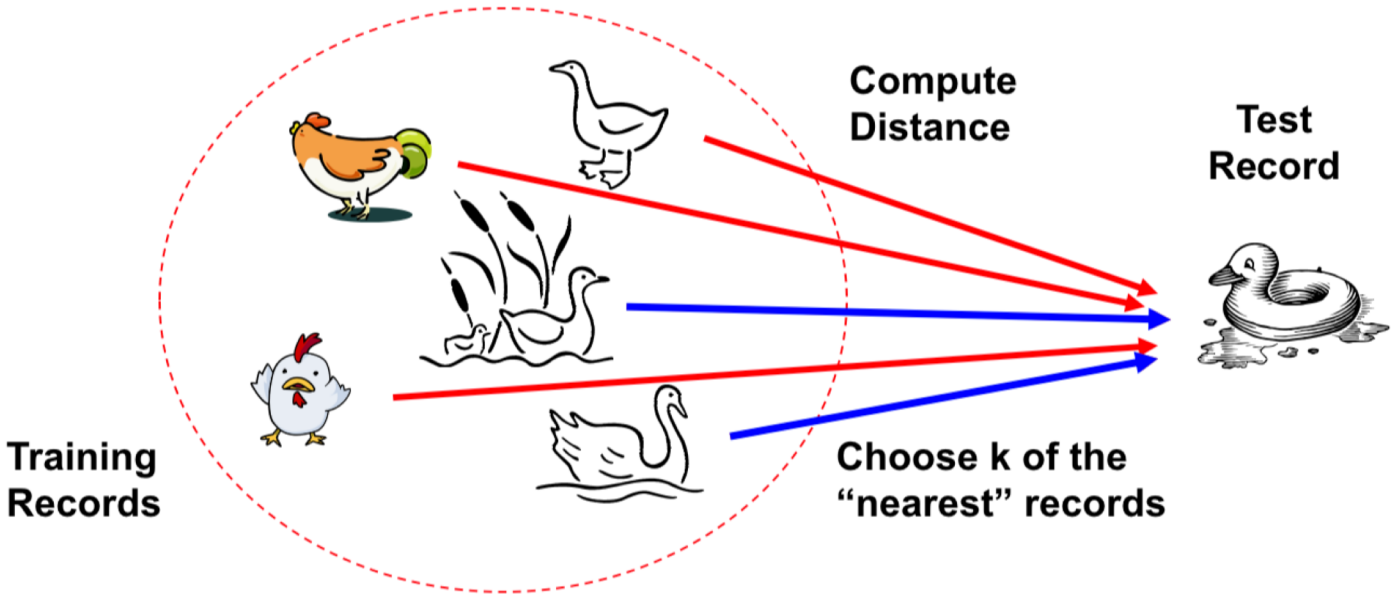
The preprocessor (the StandardScaler in the code above) is part of our algorithm, and is learning some parameters from the data (i.e. mean/variance of each feature). If we fit the preprocessor using X, this makes it influenced by the test data.

OVERVIEW

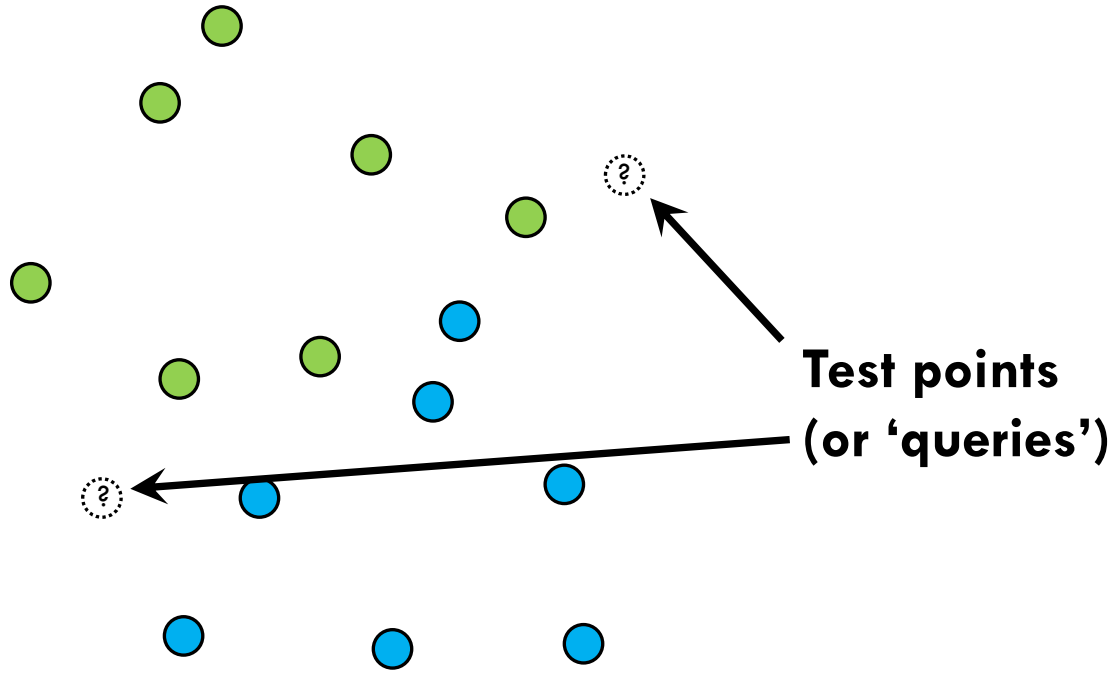
1. Problem Setup
2. Evaluating Classifiers
3. Nearest Neighbor Methods
4. Tree-Based Methods
5. Ensemble Methods
6. Logistic Regression
7. Deep Learning

K NEAREST NEIGHBORS ALGORITHM

Principle: classify each test object (or 'query') based on its ***nearest neighbors***, i.e. the training objects most similar to it

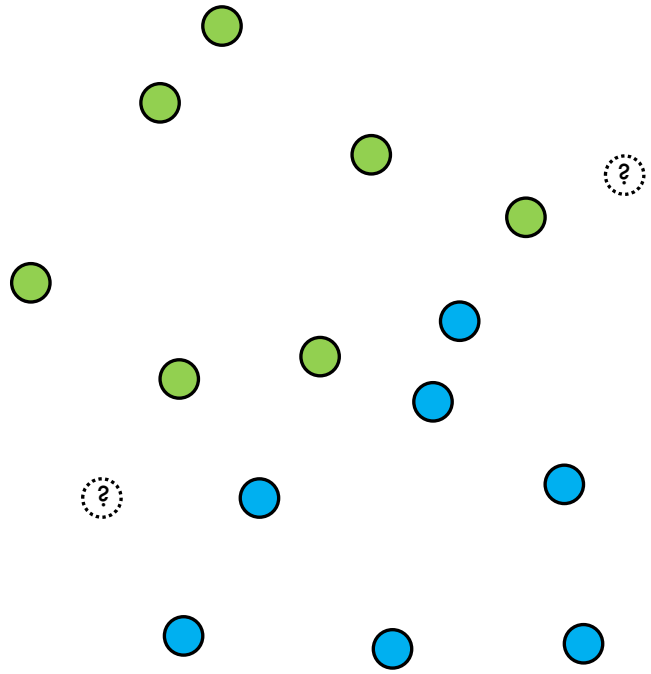


K NEAREST NEIGHBORS ALGORITHM



Training Data
(Blue = 0, Green = 1)

K NEAREST NEIGHBORS ALGORITHM

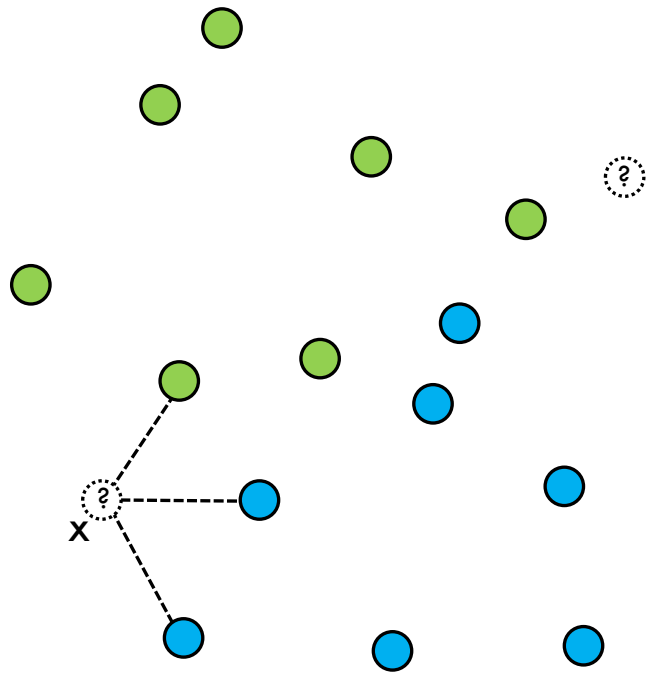


Training Data
(Blue = 0, Green = 1)

K NEAREST NEIGHBORS ALGORITHM

For each test point x:

- Compute its distances to all training instances: $d_1, \ldots, d_n$
- **Majority Vote:** Output most frequent class among the k nearest neighbors of x

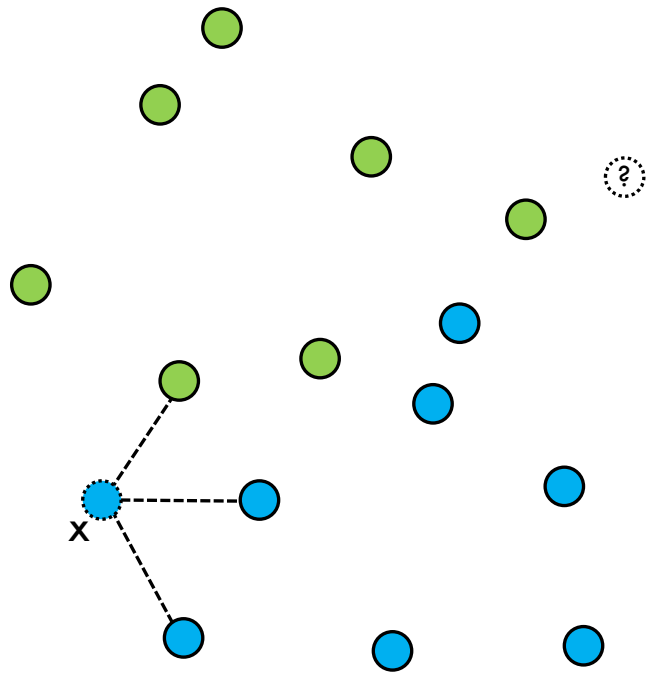


$k = 3$

K NEAREST NEIGHBORS ALGORITHM

For each test point x:

- Compute its distances to all training instances: $d_1, \ldots, d_n$
- **Majority Vote:** Output most frequent class among the k nearest neighbors of x

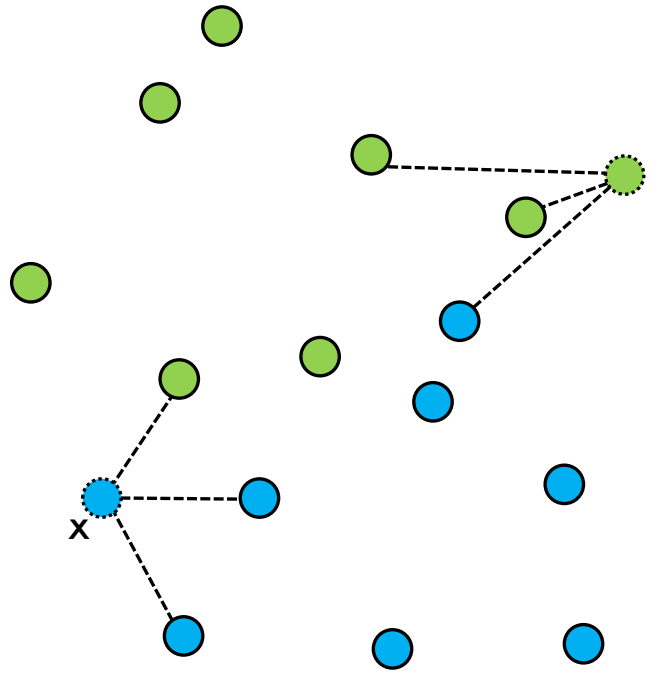


$k = 3$

K NEAREST NEIGHBORS ALGORITHM

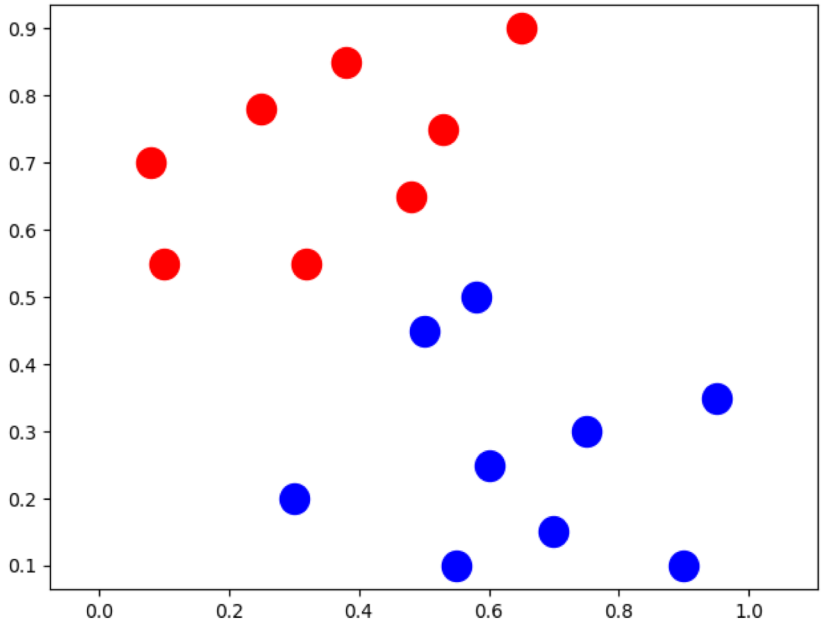
For each test point x:

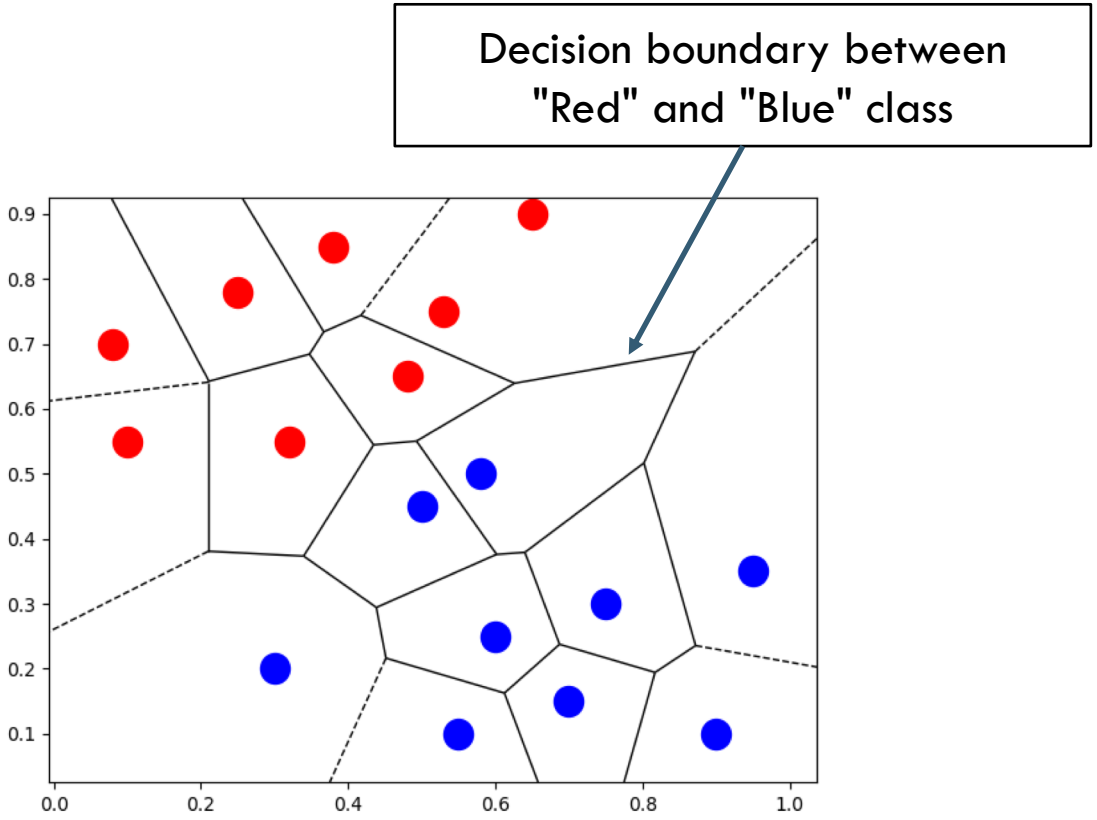
- Compute its distances to all training instances: $d_1, \ldots, d_n$
- **Majority Vote:** Output most frequent class among the k nearest neighbors of x



$k = 3$

1-NEAREST NEIGHBOR: VORONOI DIAGRAM



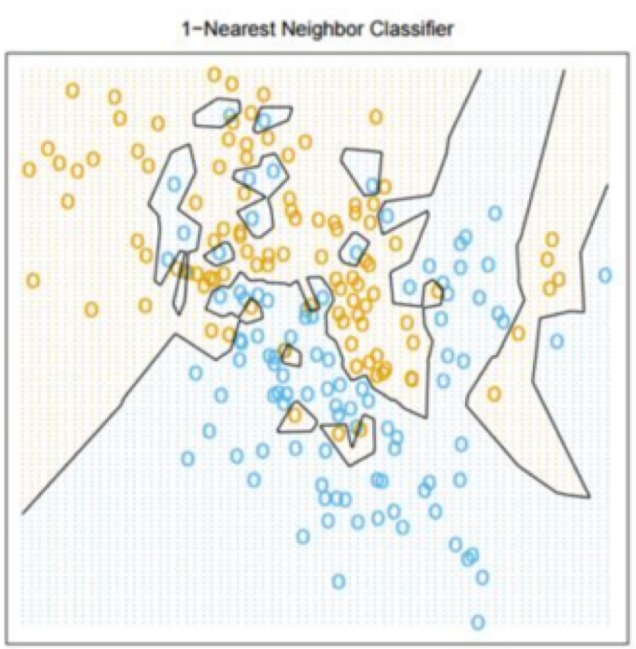


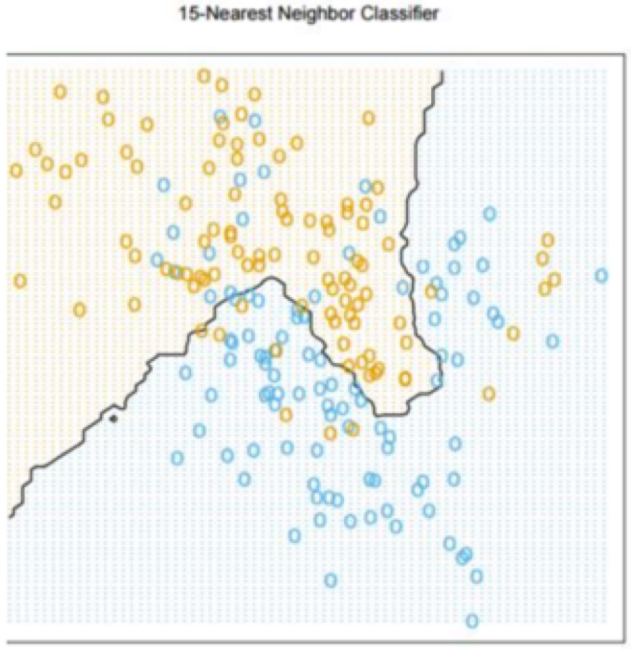
SELECTING K (UNDERFITTING VS OVERFITTING)

Selecting k is a trade-off:

- Smaller k: more flexible model, tends to overfit
- Larger k: less flexible model, tends to underfit

We can select k based on accuracy on a **validation set** or **cross-validation**



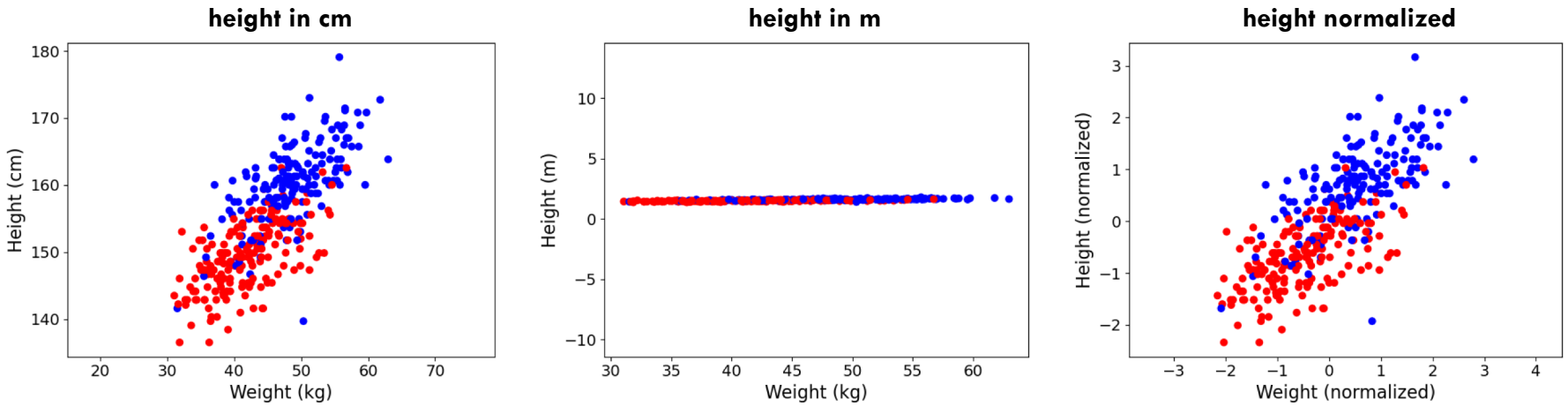


Figures from Elements of Statistical Learning (Hastie, Tibshirani, Friedman)

CAVEAT 1: SENSITIVE TO CHOICE OF SCALE

Scaling the variables (e.g. cm vs m) will change the distances, and thus the k-nearest neighbor results

Example: measuring height in meters makes the variation in height basically negligible; normalization can fix this problem

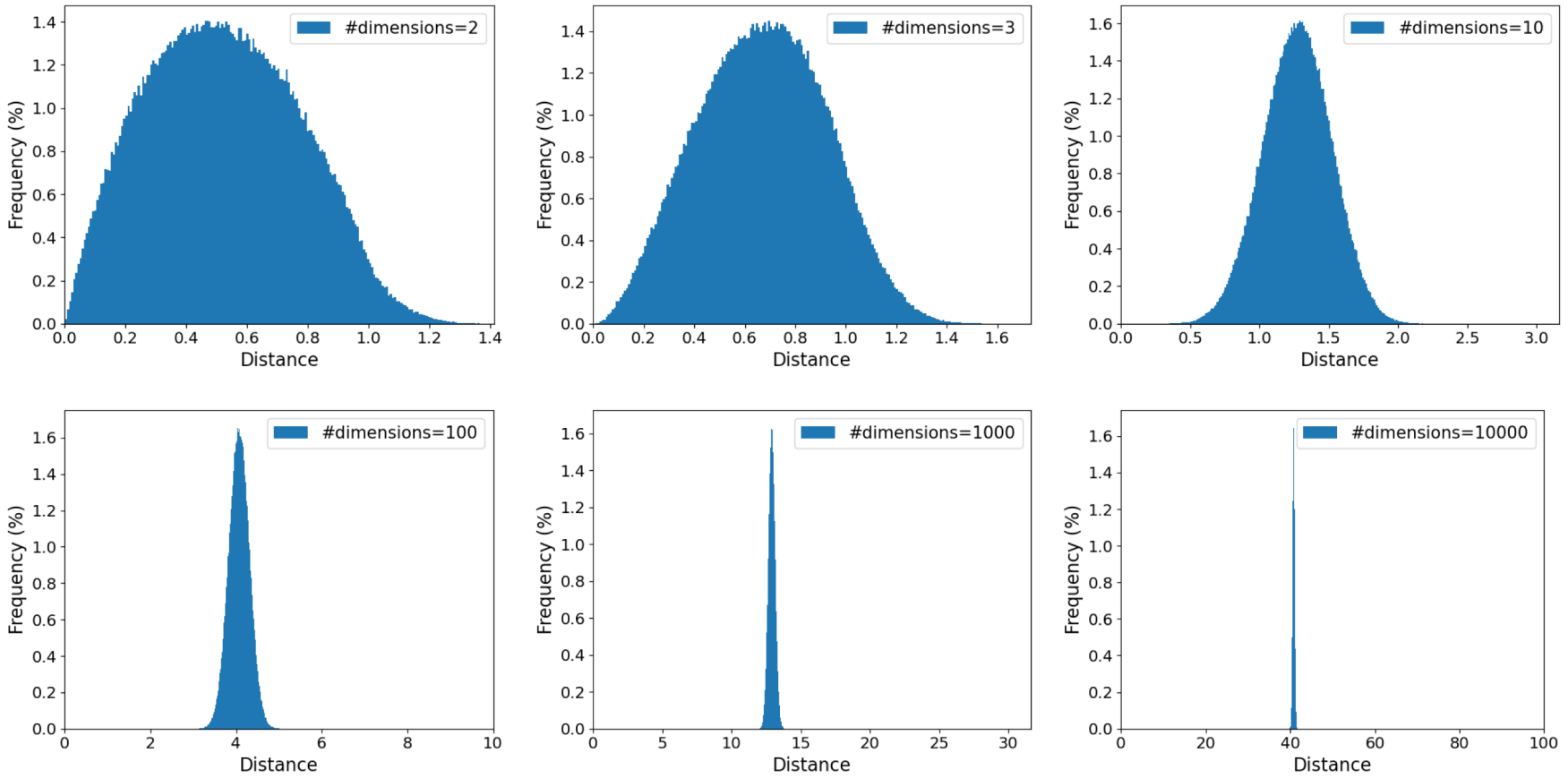


CAVEAT 2: CURSE OF DIMENSIONALITY

Intuition: As dimension size increases, the distances between most pairs of points tend to become *large* and *increasingly similar*

Hence, both distances and nearest neighbors tend to become uninformative.

Frequency plots of distances between all pairs of points, with varying dimensionality

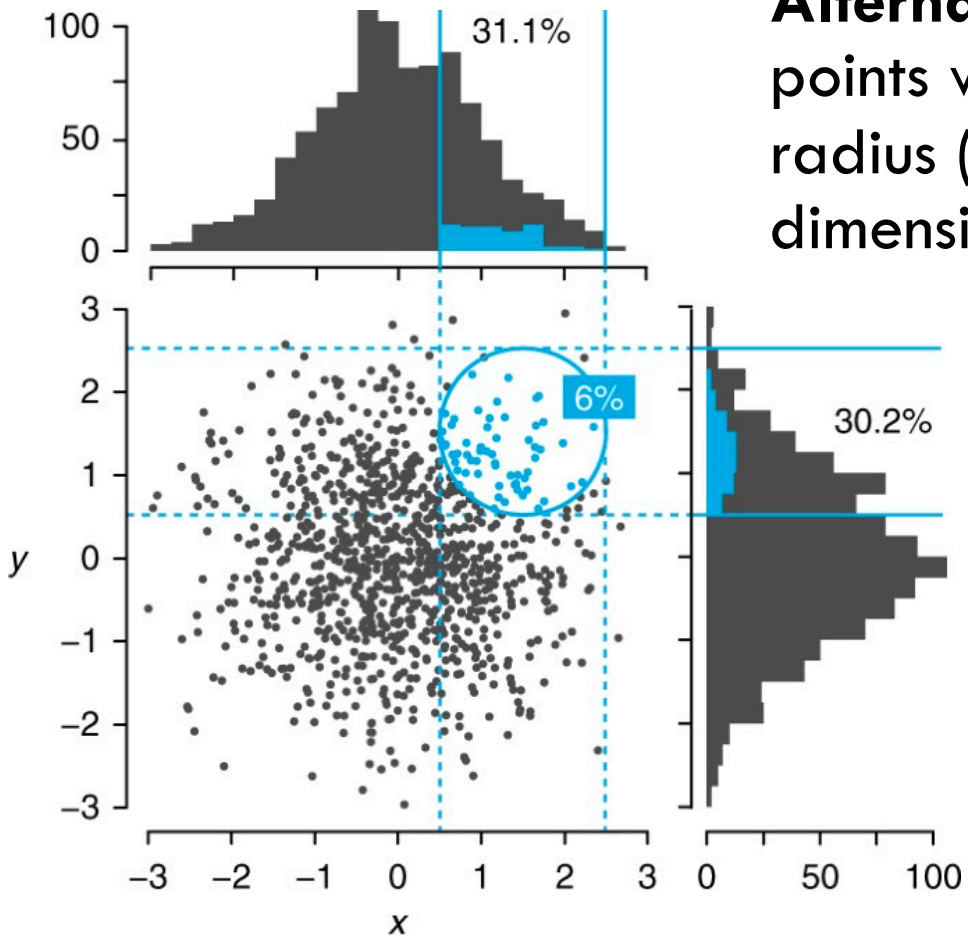


Credit: https://www.cs.cornell.edu/courses/cs4780/2018fa/

CAVEAT 2: CURSE OF DIMENSIONALITY

Intuition: As dimension size increases, the distances between most pairs of points tend to become *large* and *increasingly similar*

Hence, both distances and nearest neighbors tend to become uninformative.



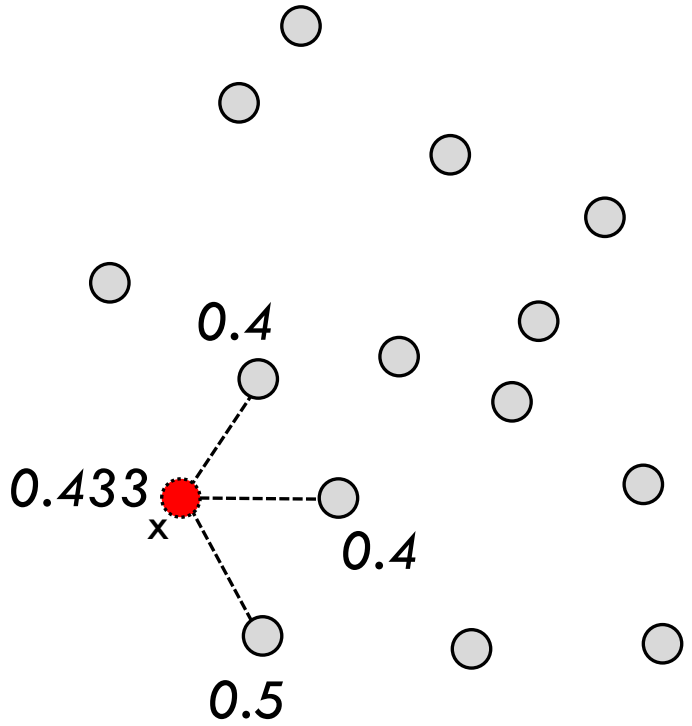
Alternate view: number of points within a ball of fixed radius (e.g. 1) decays as the dimensionality increases

Altman N, Krzywinski M. The curse (s) of dimensionality. Nat Methods. 2018 Jun 1;15(6):399-400.

K NEAREST NEIGHBORS REGRESSION

For each test point x:

- Compute its distances to all training instances: d_1, …, d_n
- **Aggregate:** Output the *average* (e.g. mean, median) of the *response variable* among the k-nearest neighbors of x

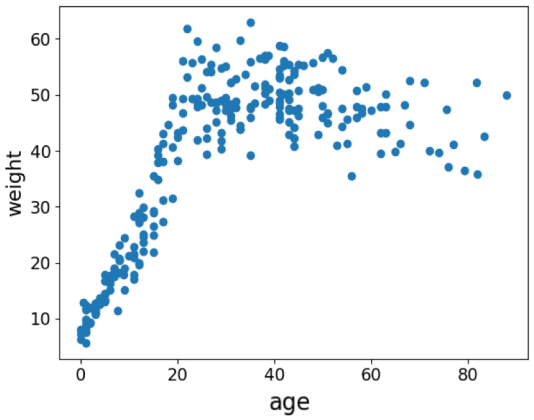


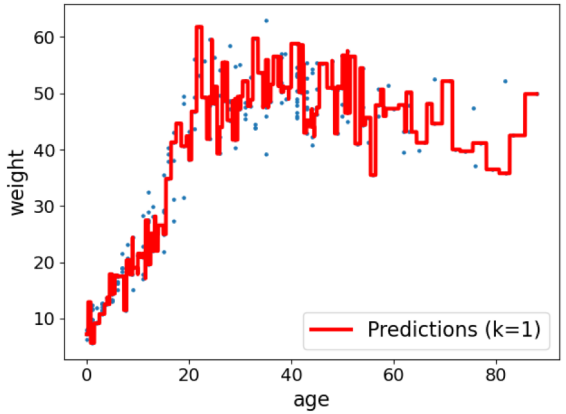
$k = 3$

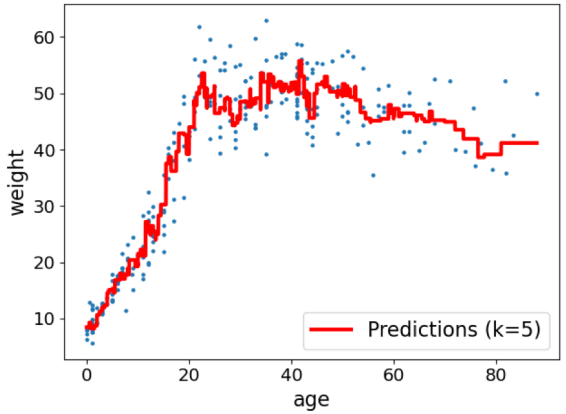
KNN FOR REGRESSION: EXAMPLE

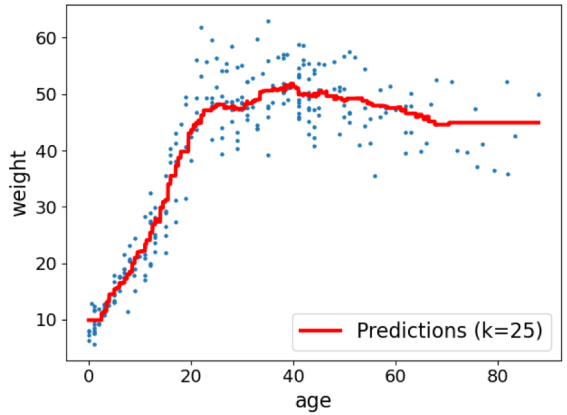
Example use-case: predict weight from age

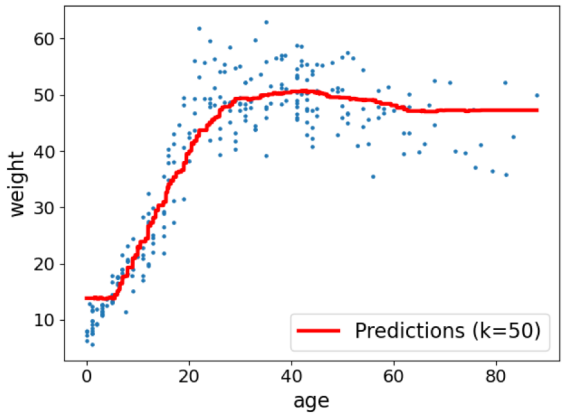
Method: k nearest neighbours for k = 1, 5, 25, 50, 100, 200

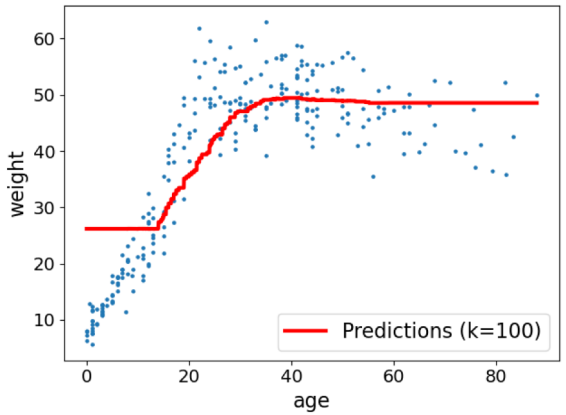


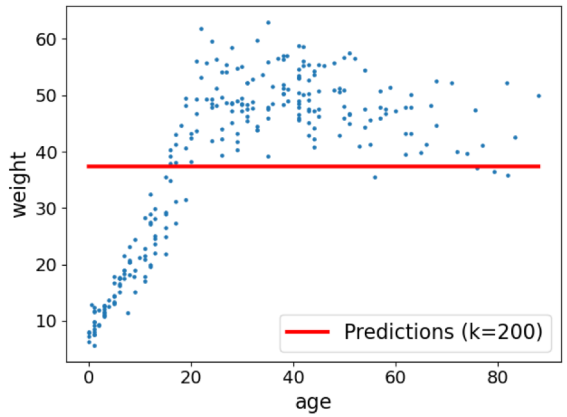








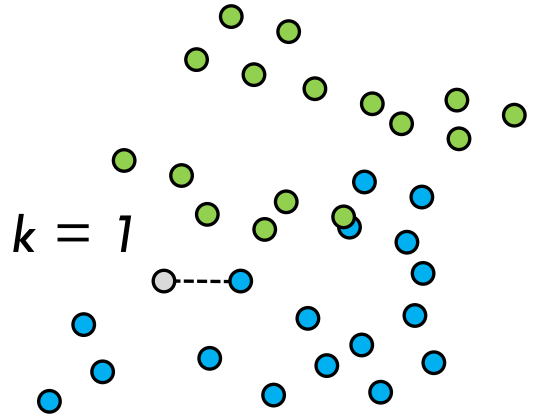




Optional

THEORETICAL GUARANTEES

(Cover, Hart) As $n \to \infty$, the error of 1-nearest neighbours is at most 2 times the error of the optimal[1] classifier.



1: The optimal ("Bayes optimal") classifier is the classifier that (unrealistically) predicts based on perfect knowledge of the **true distribution** $P(y|x)$. Specifically, the Bayes optimal classifier is defined as $y^* = \text{argmax}_y P(y|x)$. In practice, the true distribution cannot be known, so such optimal classifiers cannot be learned in practice; they are just seen as an upper bound for any classifier's performance.

Optional

SURPRISINGLY COMPETITIVE PERFORMANCE

Recommendation Systems

Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches

Maurizio Ferrari Dacrema
Politecnico di Milano, Italy
maurizio.ferrari@polimi.it

Paolo Cremonesi
Politecnico di Milano, Italy
paolo.cremonesi@polimi.it

Dietmar Jannach
University of Klagenfurt, Austria
dietmar.jannach@aau.at

"Only 7 of them could be reproduced … 6 of them can often be outperformed with comparably simply heuristic methods, e.g. based on nearest neighbor or graph-based techniques."

(Dacrema et al., 2019)

Time Series Classification

"…multiple rigorous independent studies show that for the core problem of time series classification, Nearest Neighbor DTW is very hard to beat."

(Mueen & Keogh, 2016, "Extracting Optimal Performance from Dynamic Time Warping")

PROS AND CONS

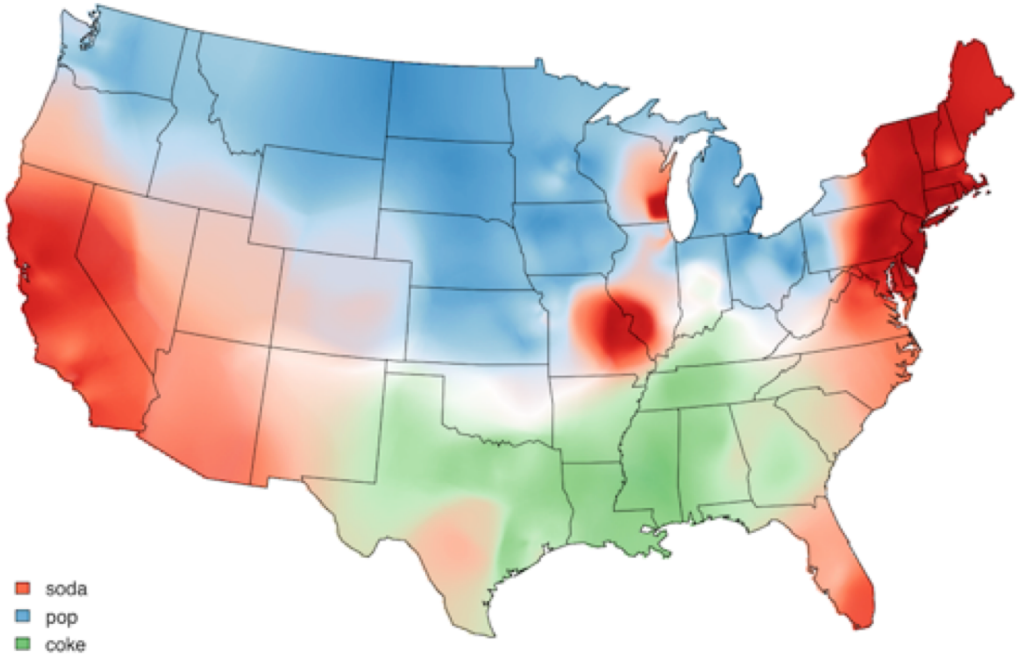
Pros:

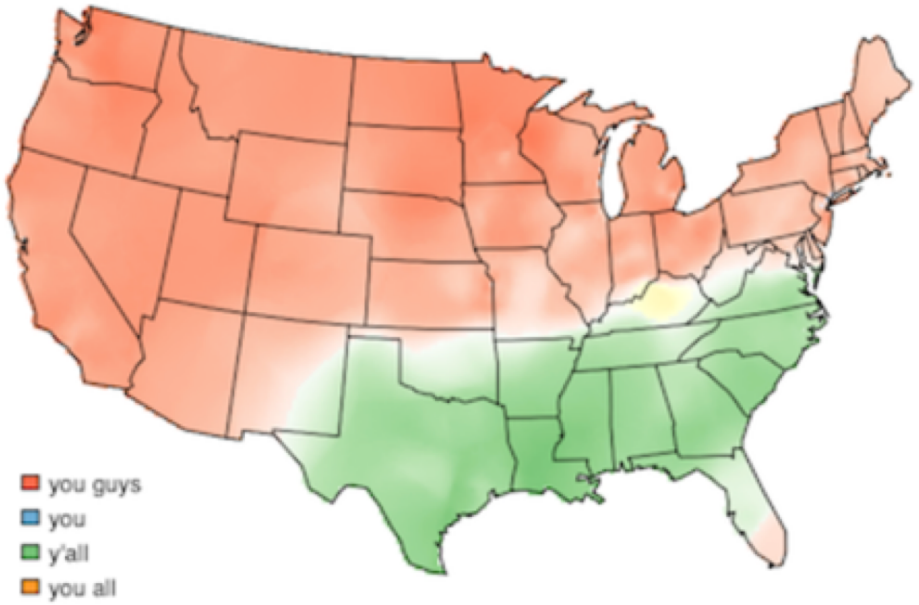
- Very simple & interpretable algorithm
- Generic: can run as long as a distance exists
- No training time
- Can produce arbitrarily shaped decision boundaries

Cons:

- Finding neighbors at test time can be slow: $O(n \times m \times d)$ time
  - (Efficiency can often be improved using fast nearest neighbor search, e.g. k-d trees for lower dimensionality data (e.g. up to 20 or so), or approximate similarity search for higher dimensionality)
- Curse of dimensionality (in high dimensions)
- Need to store & query all the training data (may lead to high memory usage)

EXAMPLE: DIALECTS OVER THE U.S.





Joshua Katz: http://www4.ncsu.edu/~jakatz2/files/dialectposter.png