Classification

STAT 471

September 28, 2021

Where we are

Unit 1: Intro to modern data mining

Unit 2: Tuning predictive models

Unit 3: Regression-based methods

Unit 4: Tree-based methods

Unit 5: Deep learning

Lecture 1: Model complexity

Lecture 2: Bias-variance trade-off

Lecture 3: Cross-validation

Lecture 4: Classification

Lecture 5: Unit review and quiz in class

Homework 1 due the following Monday.

Recall: Clinical decision support

A patient comes into the emergency room with stroke symptoms. Based on her CT scan, is the stroke ischemic or hemorrhagic?

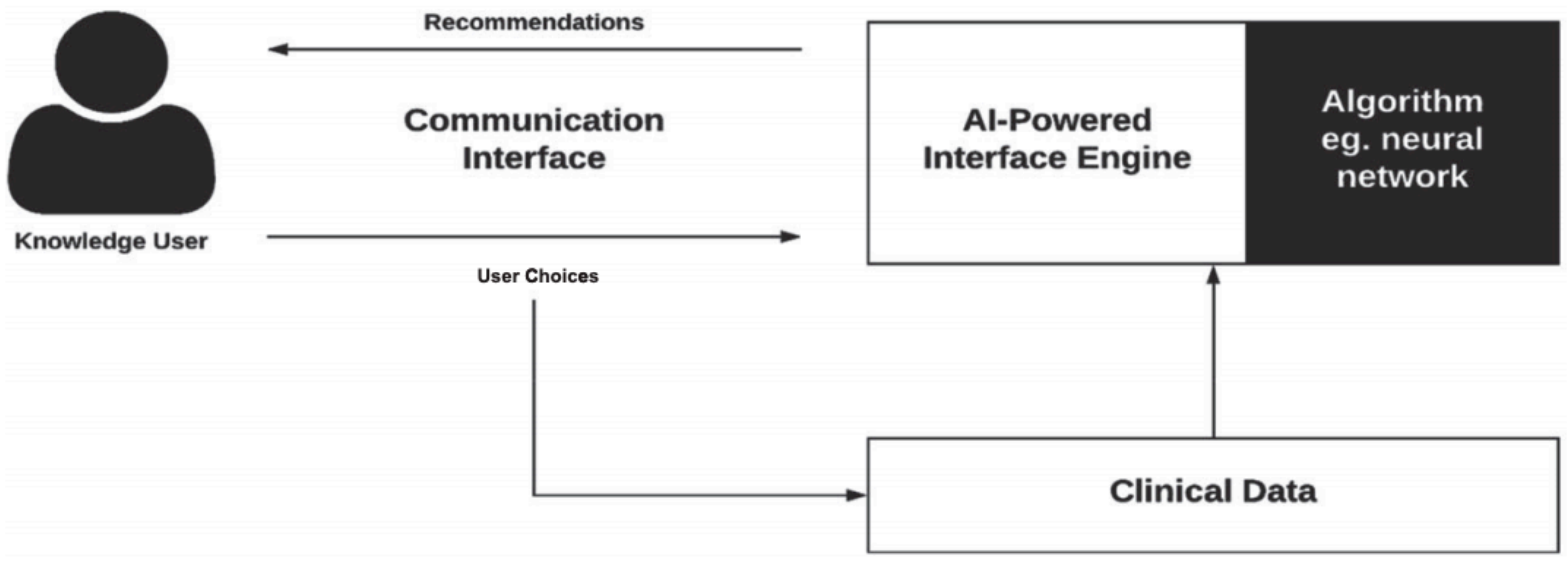


Image source: Sutton et al. 2020 (npj Digit. Med.)

This is a binary classification problem: $Y \in \{0,1\}$.

Given features $X = (X_1, \ldots, X_p)$, the goal is to guess a response $\widehat{Y} = \hat{f}(X)$ that is close to the true response, i.e. $\widehat{Y} \approx Y$. Measure of success is usually the

$$\text{test misclassification error} = \frac{1}{N}\sum_{i=1}^{N} I(Y_i^{\text{test}} \neq \hat{f}(X_i^{\text{test}})).$$

Classification via probability estimation

Suppose that the true relationship between Y and X is

$$\mathbb{P}[Y = 1 \,|\, X] = p(X), \quad \text{for some function } p.$$

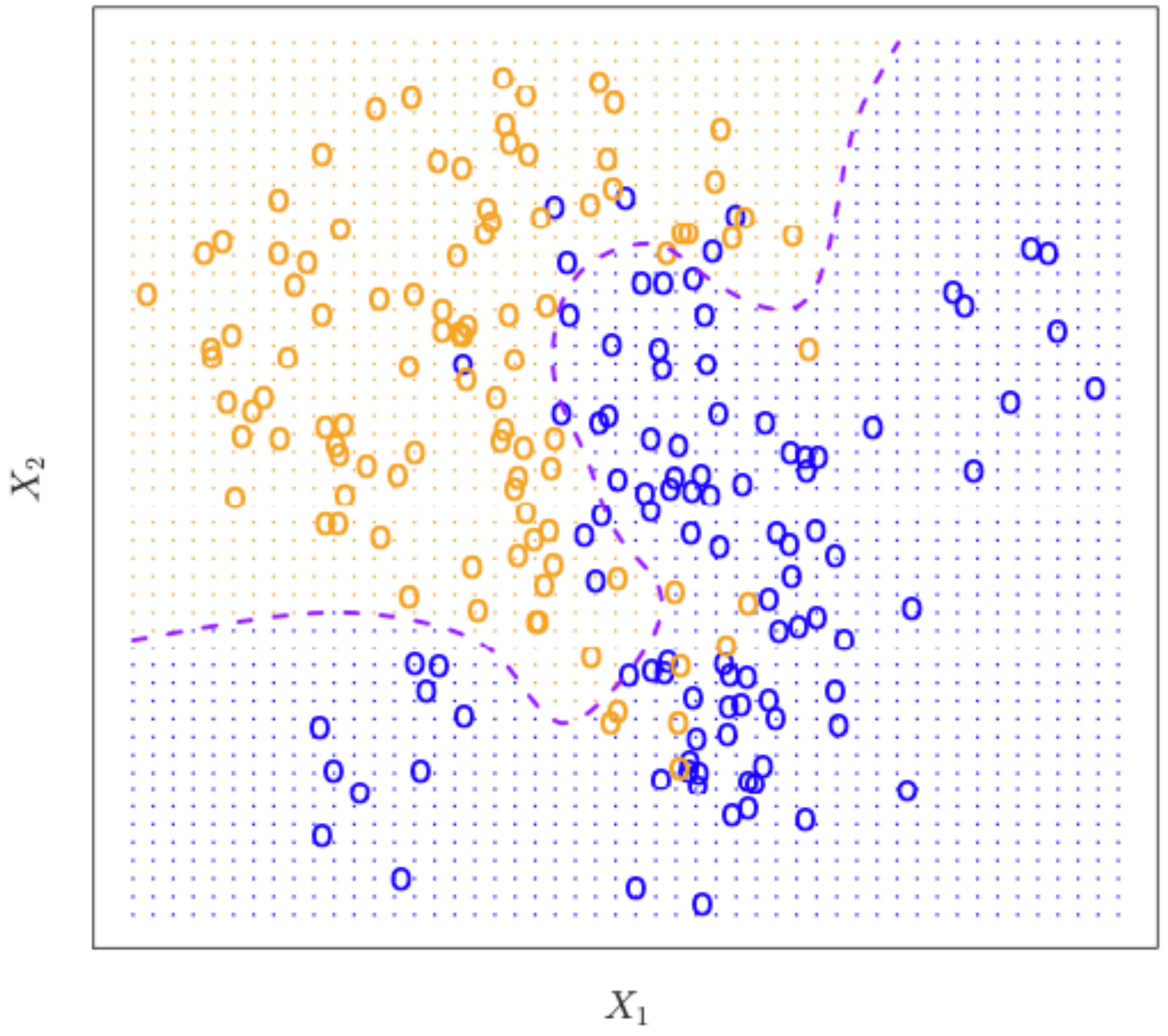
Then, the optimal classifier (called the Bayes classifier) is

$$\hat{f}^{\text{Bayes}}(X) = \begin{cases} 1, & \text{if } p(X) \geq 0.5; \\ 0 & \text{if } p(X) < 0.5. \end{cases}$$

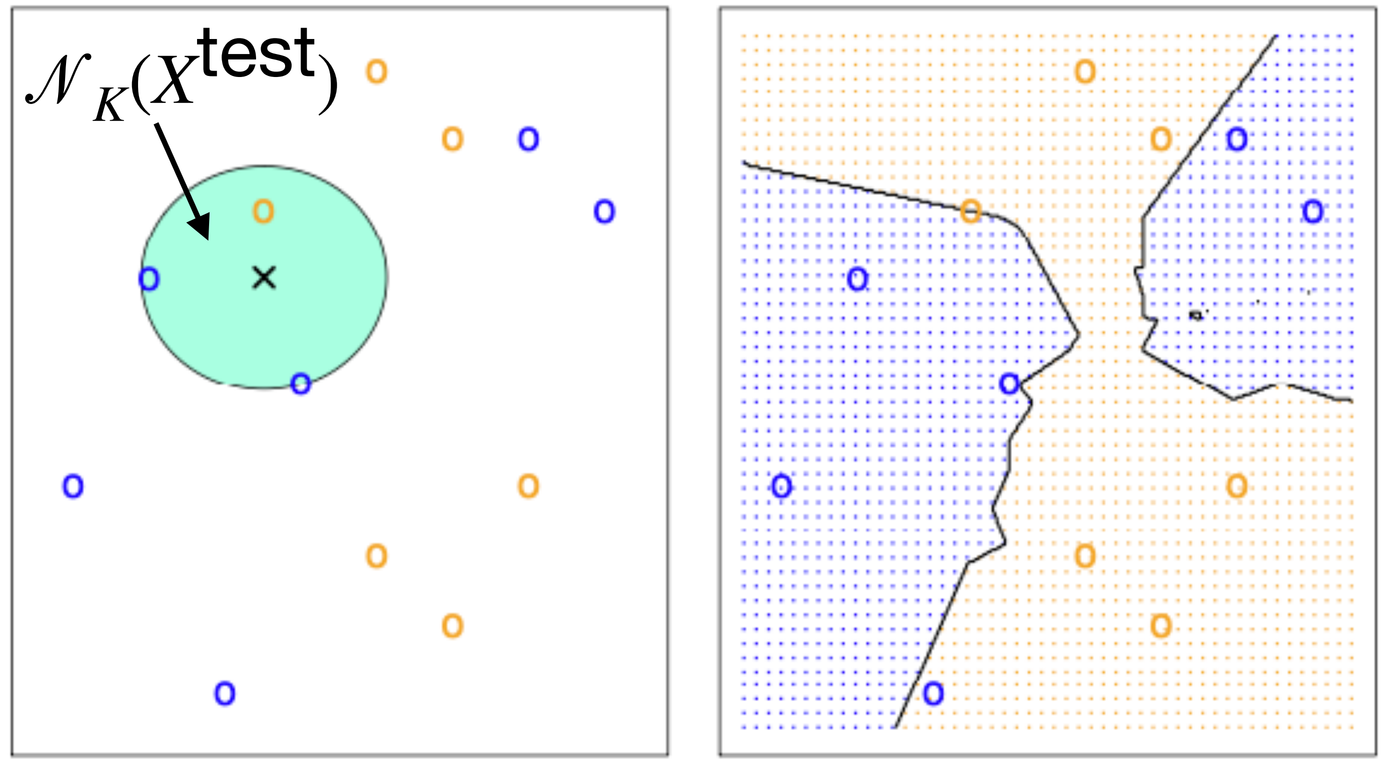
Classifiers usually build an approximation $\widehat{p}(X) \approx \mathbb{P}[Y = 1 \,|\, X]$, and define

$$\hat{f}(X) = \begin{cases} 1, & \text{if } \widehat{p}(X) \geq 0.5; \\ 0 & \text{if } \widehat{p}(X) < 0.5. \end{cases}$$

Example: K-nearest neighbors

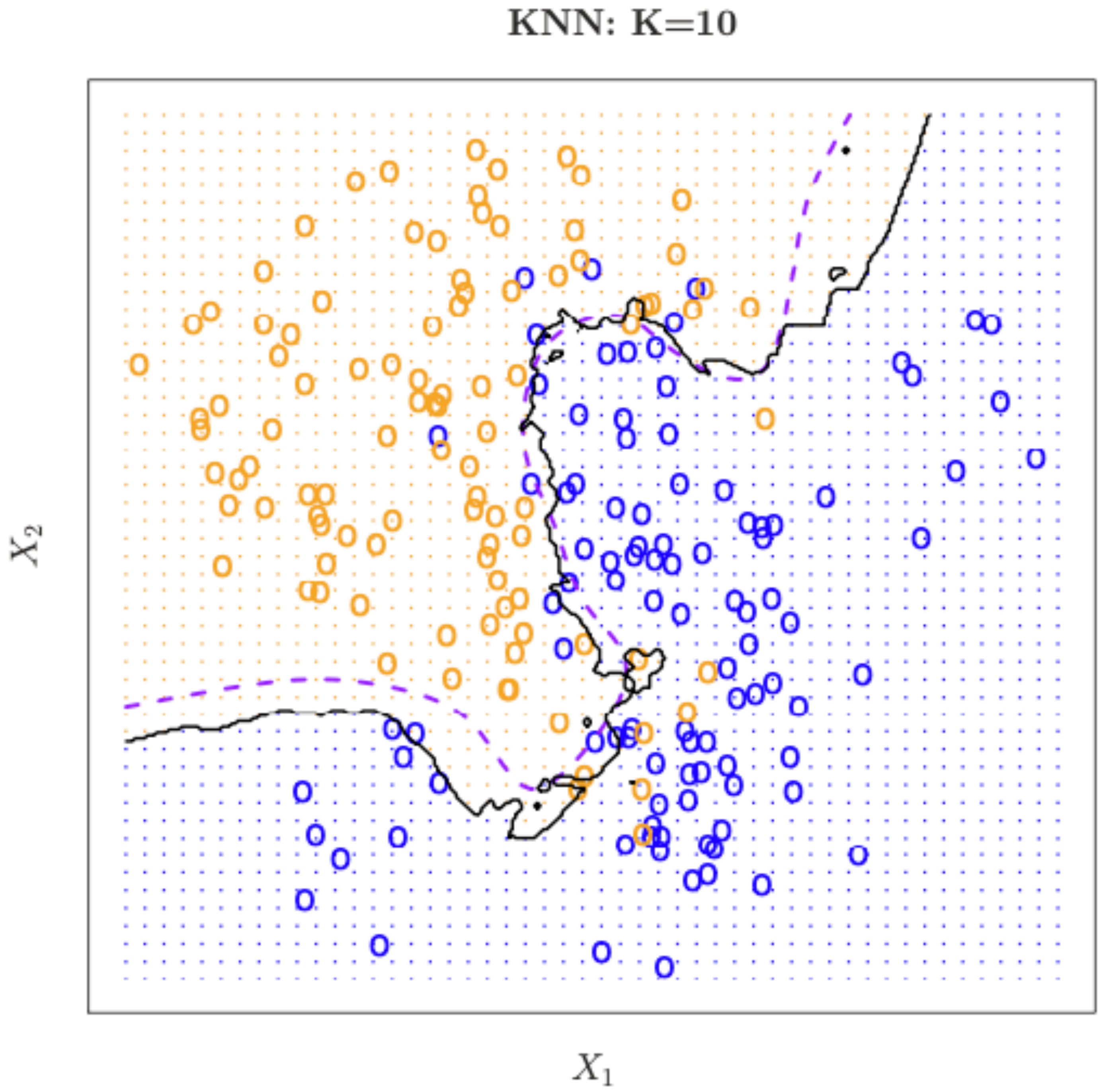


Simulated binary classification data.
Bayes classifier in purple.
E.g., color = stroke type, (X_1, X_2) = CT image.



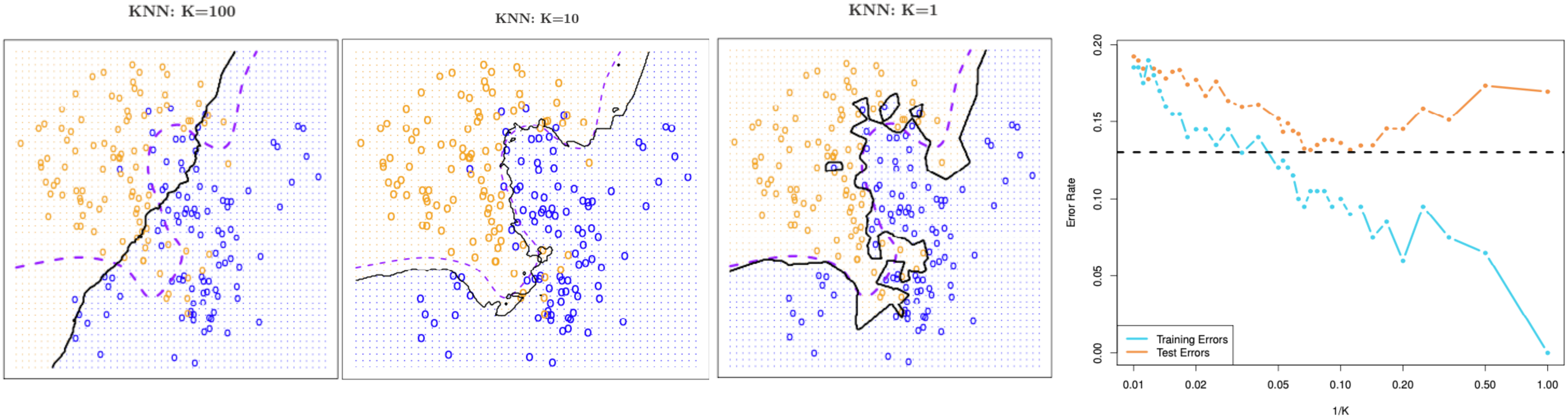
KNN illustration: Classify a test point based on majority vote among 3 nearest neighbors.

$$\hat{p}(X^{\text{test}}) = \frac{1}{K} \sum_{i \in \mathcal{N}_K} I(X_i^{\text{train}} = 1)\,.$$



Applying KNN with K = 10 to simulated data.

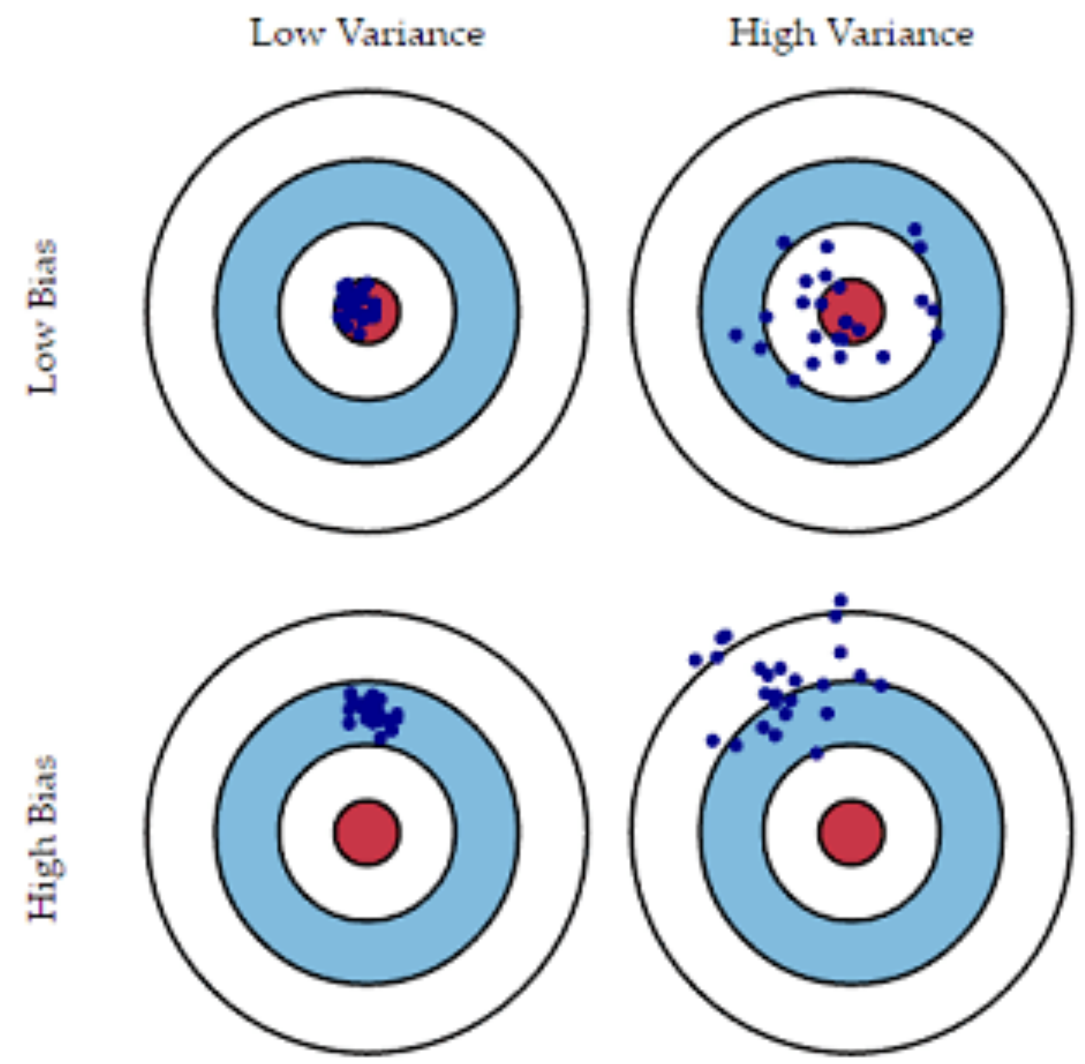
Model complexity and misclassification error



Same Goldilocks principle as in regression case:

- Too little complexity: Can’t capture the true trend in the data.
- Too much complexity: Too sensitive to noise in the training data (overfitting).

Bias-variance tradeoff



Mathematically: Applies only to continuous response variables and MSE.

Intuitively: Applies to any prediction problem, including classification.

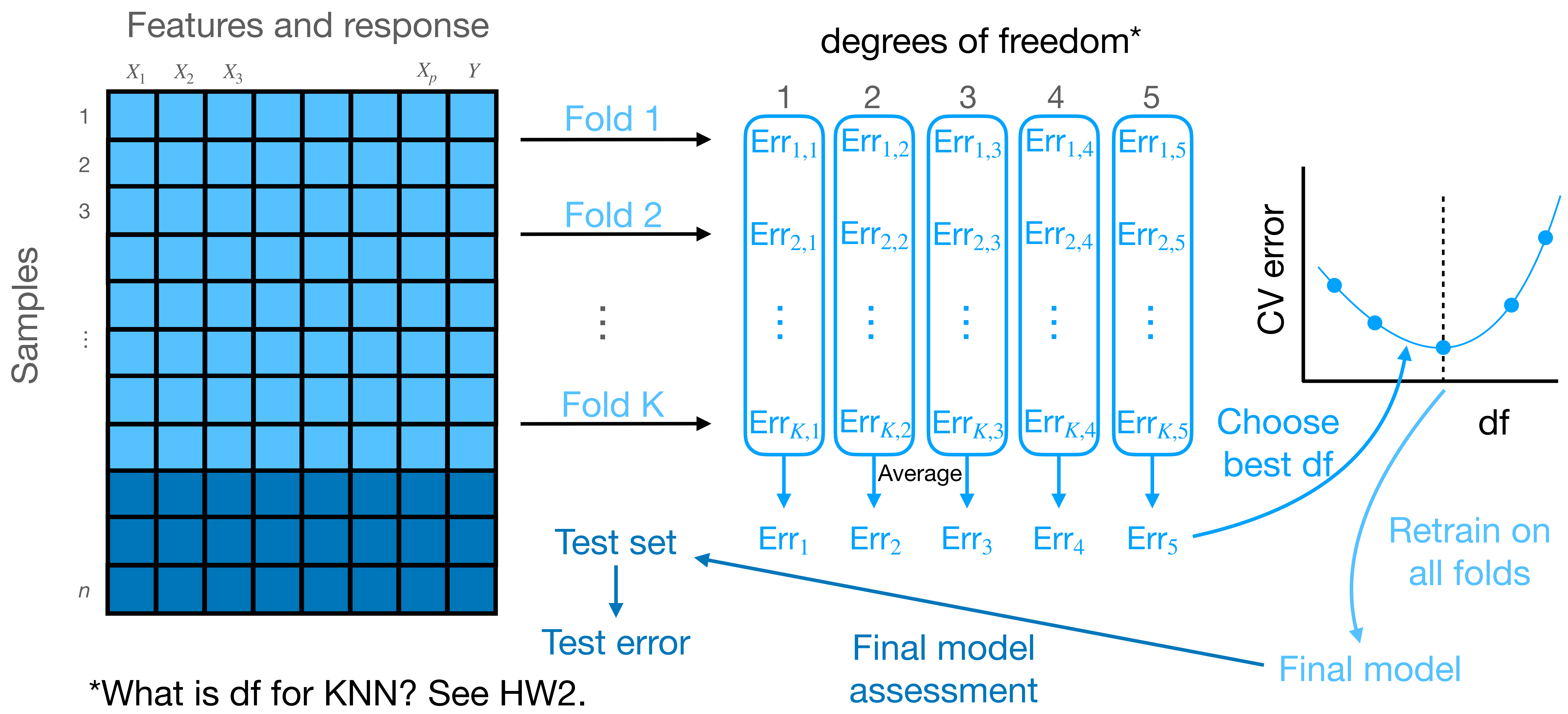
<u>For the estimate $\widehat{p}(X)$</u>

- Bias: $\mathbb{E}[\widehat{p}(X)] - p(X)$ $\longrightarrow$
- Variance: $\mathrm{Var}[\widehat{p}(X)]$ $\longrightarrow$

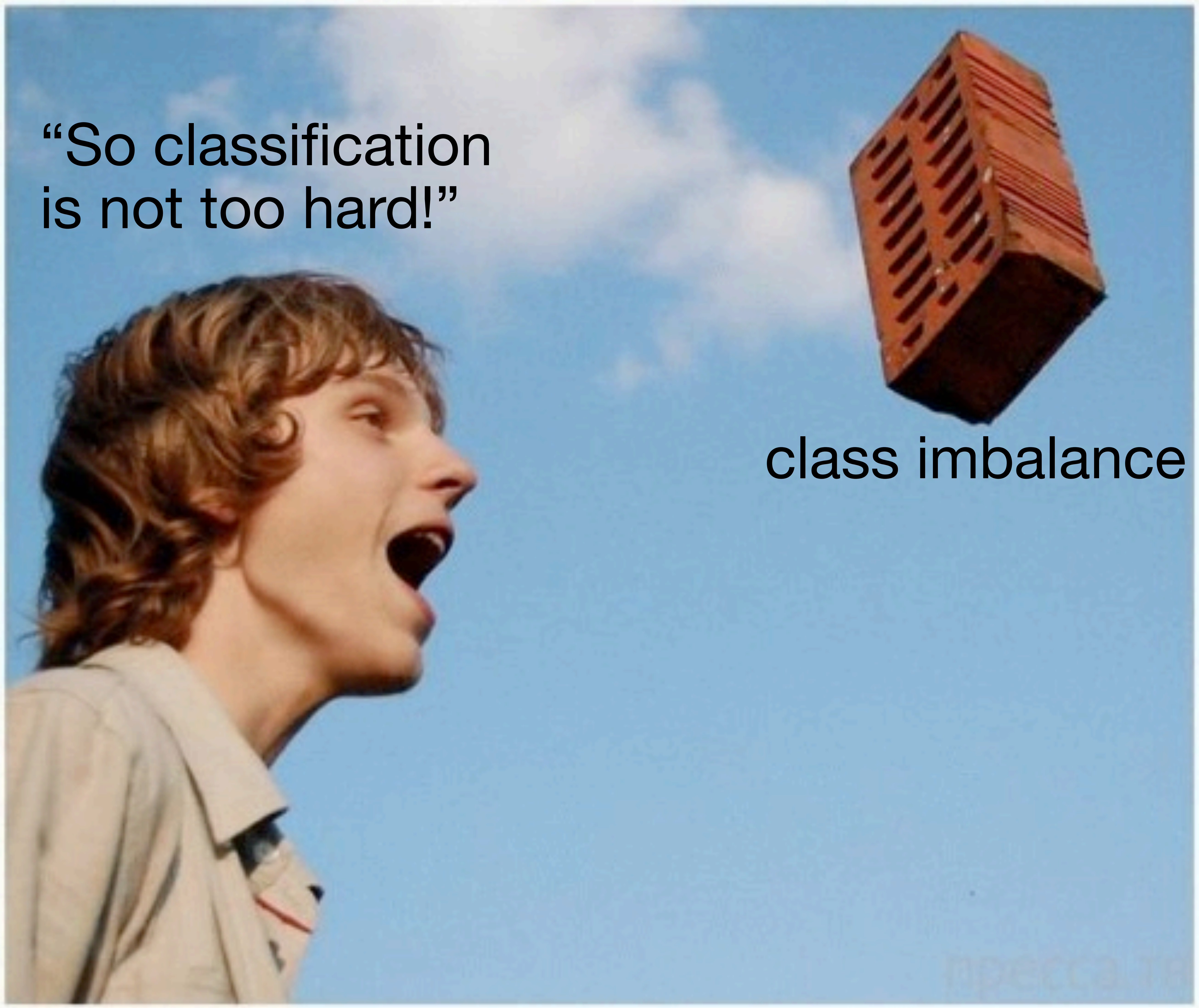
<u>For classifying $\widehat{Y} = I(p(X) \geq 0.5)$</u>

- Bias: Predict wrong class on average, to the extent $\widehat{p}$ on wrong side of 0.5
- Variance: Prediction varies with training set, to the extent $\widehat{p}$ fluctuates above or below 0.5
- Irreducible error (AKA Bayes error): Error incurred by Bayes classifier because $0 < \mathbb{P}[Y = 1 \mid X] < 1$.

Cross-validation based on misclassification error
(otherwise same as before)



*What is df for KNN? See HW2.


"So classification
is not too hard!"
class imbalance

Class imbalance

In many real-world classification problems, one class (say $Y = 1$) is significantly less frequent than the other. For example:

- Credit card transaction classification: normal versus fraudulent
- COVID testing: negative versus positive

Often in these cases, the costs of misclassification are also asymmetric, i.e. the misclassification error is not the right metric.

Let's say 1% of credit card transactions are fraudulent. Then, the classifier that always predicts "not fraudulent" will have a misclassification error of only 1%.

Cross-validation based on misclassification error leads to overly simple models that ignore the minority class.

A more wholistic picture of a classifier

Confusion matrix

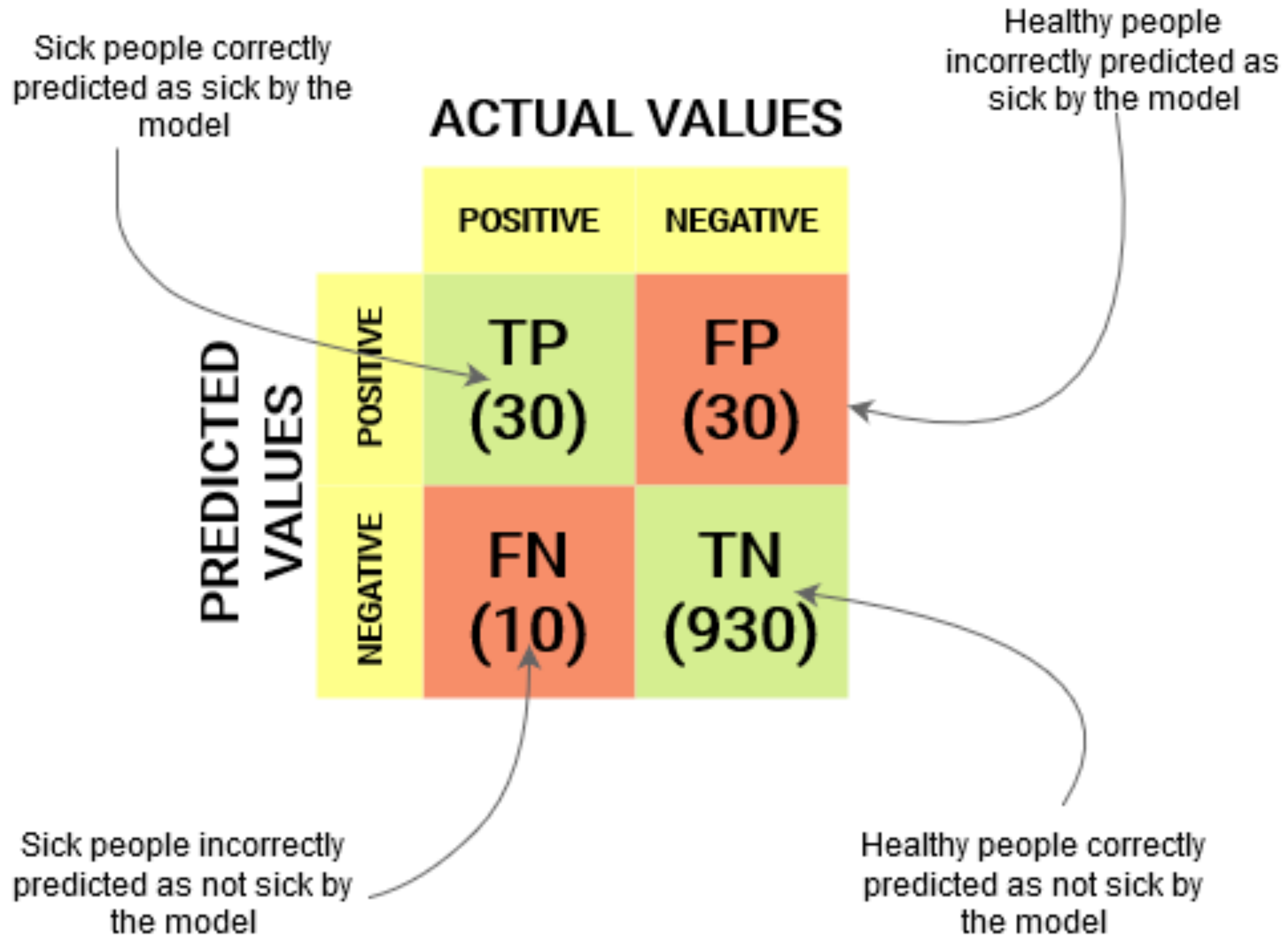


Image source: https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/

Summaries of confusion matrix

$$\text{False positive rate} = \frac{\text{number false positives}}{\text{total actual negatives}}$$

$$\text{False negative rate} = \frac{\text{number false negatives}}{\text{total actual positives}}$$

Thinking about misclassification costs

The cost of a false negative might be much greater than a false positive:

- Undetected fraudulent credit card transaction (false negative)
  → drained bank account. Cost: $C_{\text{FN}} = \$10{,}000$.
- False alarm of fraud (false positive)
  → annoying text message and/or replaced credit card. Cost: $C_{\text{FP}} = \$10$.

Weighted misclassification error:

$$\frac{1}{N}\sum_{i=1}^{N} C_{\text{FP}} \cdot I(\widehat{Y}_i^{\text{test}} = 1, Y_i^{\text{test}} = 0) + C_{\text{FN}} \cdot I(\widehat{Y}_i^{\text{test}} = 0, Y_i^{\text{test}} = 1).$$

Building misclassification costs into training

There may be two issues with

$$\hat{f}(X) = \begin{cases} 1, & \text{if } \widehat{p}(X) \geq 0.5; \\ 0 & \text{if } \widehat{p}(X) < 0.5. \end{cases}$$

1. The minority class is poorly captured by the probability model $\widehat{p}(X)$.
2. The probability threshold of 0.5 is suboptimal.

To fix these, a variety of strategies can be employed:

- Downsample the majority class by a factor $C_{\text{FP}}/C_{\text{FN}}$.
- Choose the probability threshold $C_{\text{FP}}/(C_{\text{FN}} + C_{\text{FP}})$ instead of 0.5.
- Build cost directly into the objective function when training.

Example: KNN with $K = \infty$

Suppose we apply KNN with $K = \infty$ (each data point has the same prediction); class 0 costs \$10 to misclassify and class 1 costs \$1000 to misclassify.

Let $\hat{c}$ be the class predicted for each data point. Then, we have

$$10 \cdot \mathbb{P}[\hat{Y} = 1, Y = 0] + 1000 \cdot \mathbb{P}[\hat{Y} = 0, Y = 1] = \begin{cases} 10 \cdot \mathbb{P}[Y = 0], & \text{if } \hat{c} = 0; \\ 1000 \cdot \mathbb{P}[Y = 1], & \text{if } \hat{c} = 1. \end{cases}$$

Therefore, we should set

$$\hat{c} = \begin{cases} 1 & \text{if } \mathbb{P}[Y = 1] \geq \dfrac{10}{10 + 1000}; \\ 0 & \text{if } \mathbb{P}[Y = 1] < \dfrac{10}{10 + 1000}. \end{cases}$$

We can recover this prediction rule from KNN via downsampling, threshold adjustment, or cost-sensitive training (in general these three strategies can give different answers).

Evaluating classification errors on a test set

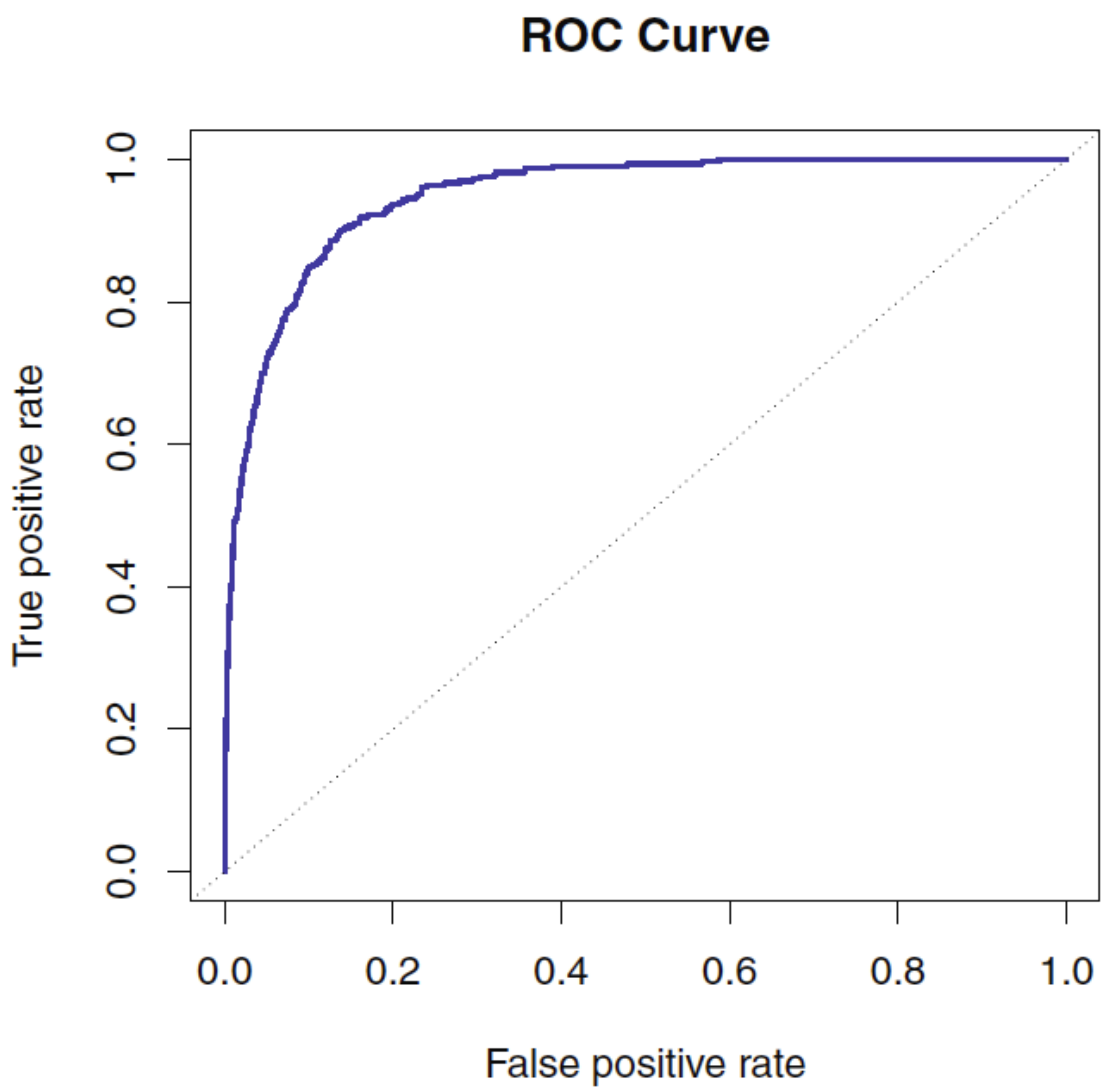
Given C_{FN} and C_{FP}, best single number to summarize classification performance is the weighted misclassification error on the test set.

There are other ways of assessing classification performance without quantifying these costs:

- Confusion matrix
- False positive rate and false negative rate
- Receiver operating characteristic (ROC) curve; area under the curve (AUC)

ROC curve

- The ROC curve plots the true positive rate (one minus the false negative rate) versus the false positive rate, as the threshold is varied from 0 to 1.
- We want the curve to get as close to the upper left-hand corner as possible.
- Area under the curve (AUC) is another measure of the quality of a classifier.



Summary

- Classification problem is similar in some ways to regression; different in others.
- Classification typically done by estimating $\mathbb{P}[Y = 1 \mid X]$, thresholding at 0.5 (e.g. KNN).
- The bias-variance tradeoff carries over intuitively, but not mathematically, to classification.
- The misclassification error is not a good metric for problems when different misclassifications have different costs; often the case when classes are imbalanced.
- Other metrics for classifiers include the weighted misclassification error, false positive and false negative rates (based on the confusion matrix), and ROC curve.
- Class imbalance can be remedied through downsampling, threshold adjustment, or cost-sensitive training.