Classification

STAT 4710

September 26, 2023

Where we are

Unit 1: R for data mining

Unit 2: Prediction fundamentals

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

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?

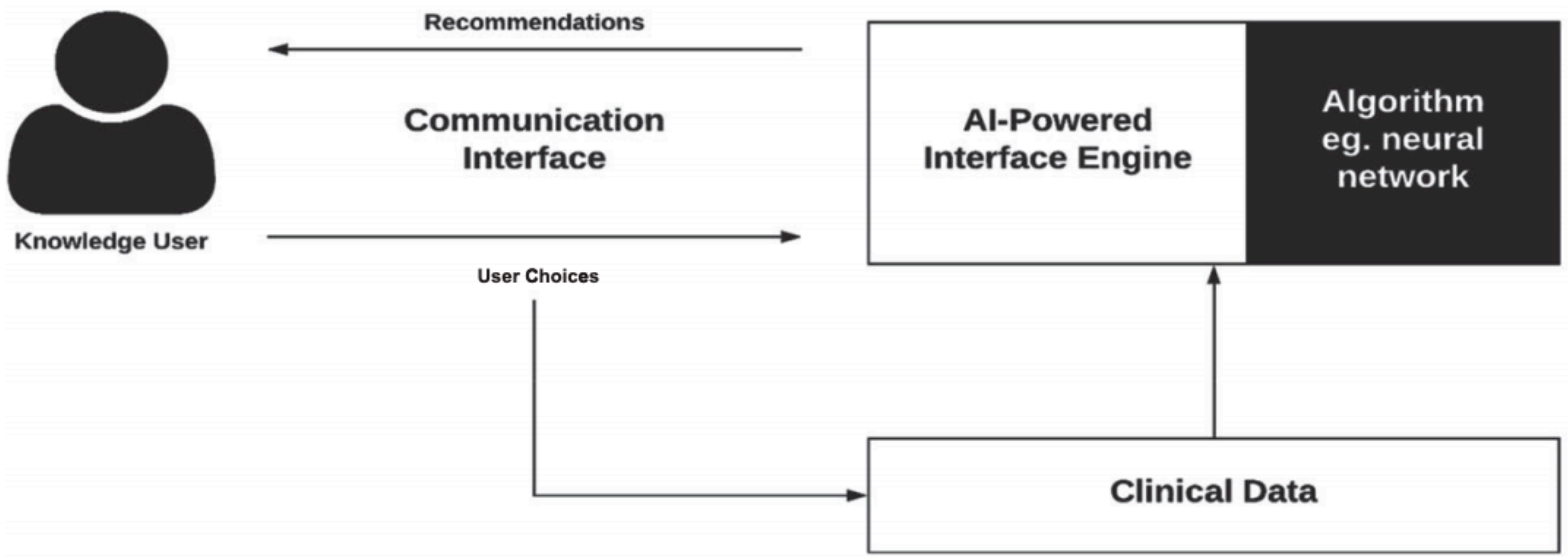


Image source: Sutton et al. 2020 (npj Digit. Med.)

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?

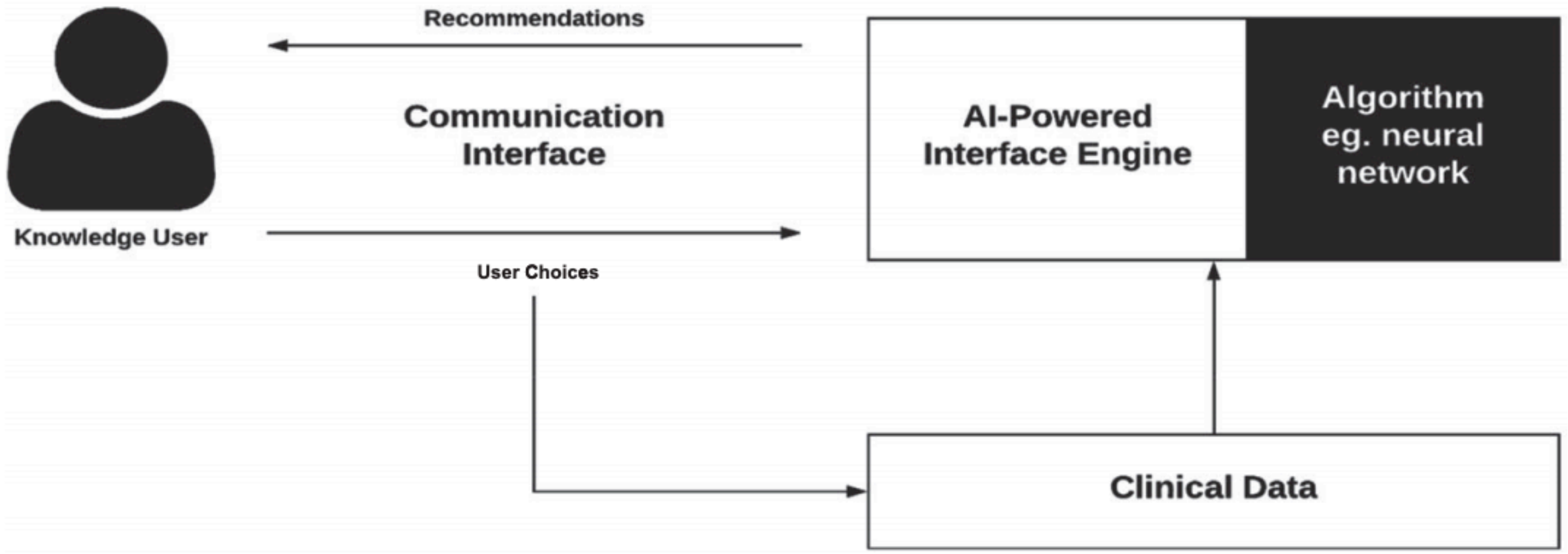


Image source: Sutton et al. 2020 (npj Digit. Med.)

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 $\hat{Y} = \hat{f}(X)$ that is close to the true response, i.e. $\hat{Y} \approx Y$. 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

Classification via probability estimation

Suppose that the true relationship between Y and X is

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

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

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

Classification via probability estimation

Suppose that the true relationship between Y and X is

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

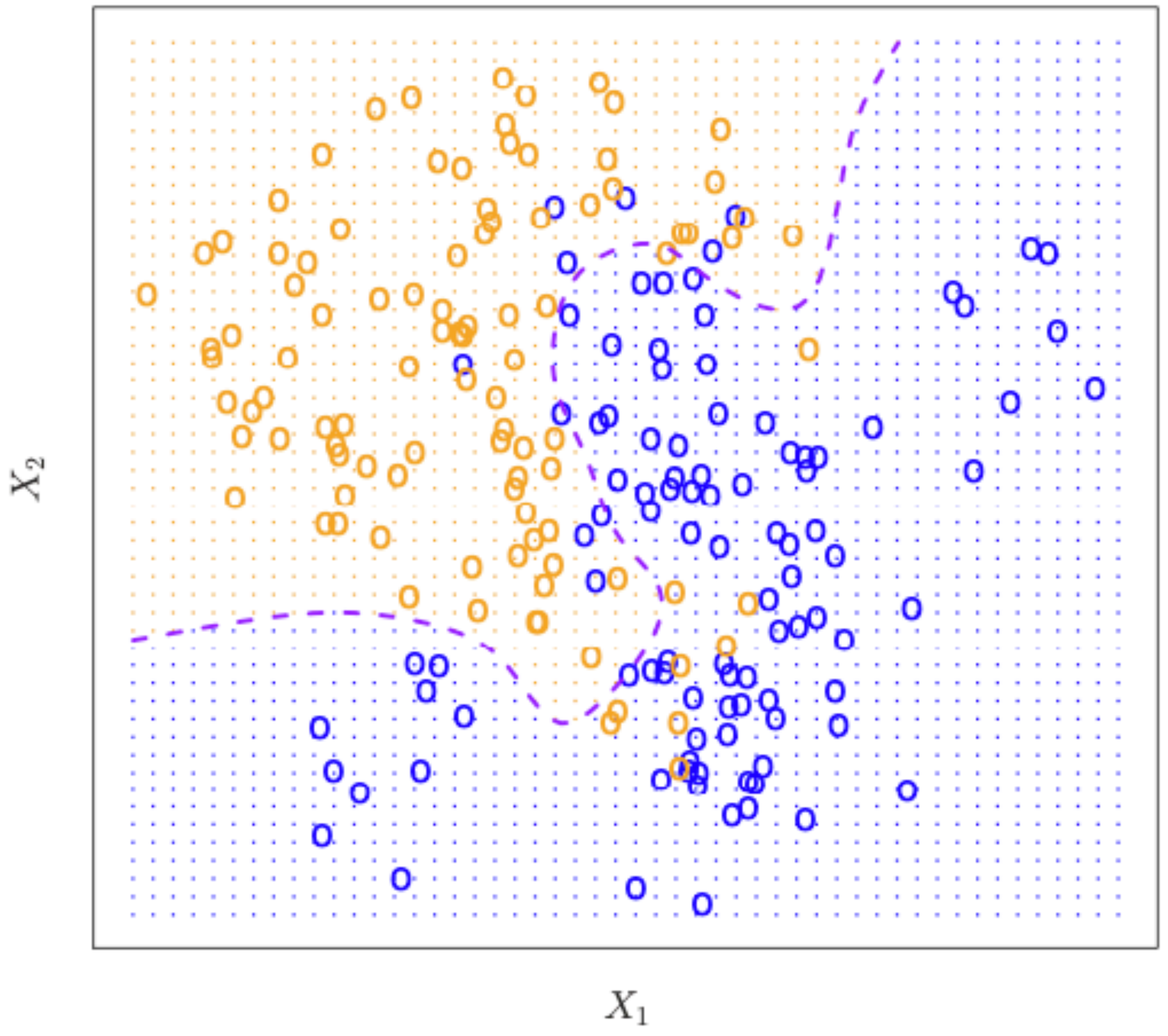
Then, the optimal classifier (called the Bayes classifier) is

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

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

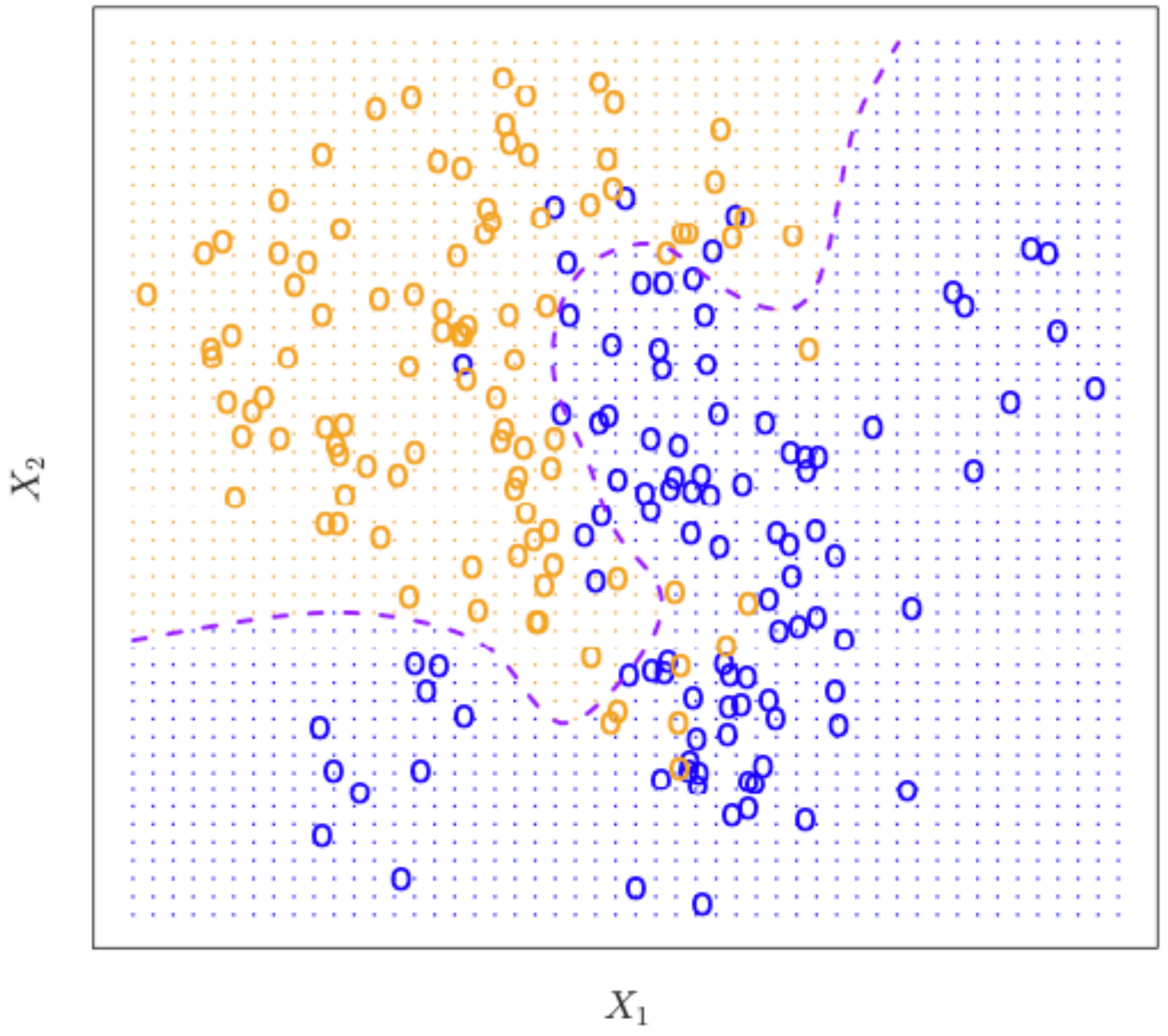
$$\hat{f}(X) = \begin{cases} 1, & \text{if } \widehat{p}(X) \geq 0.5; \\ 0 & \text{if } \widehat{p}(X) < 0.5. \end{cases}$$

Example: K-nearest neighbors

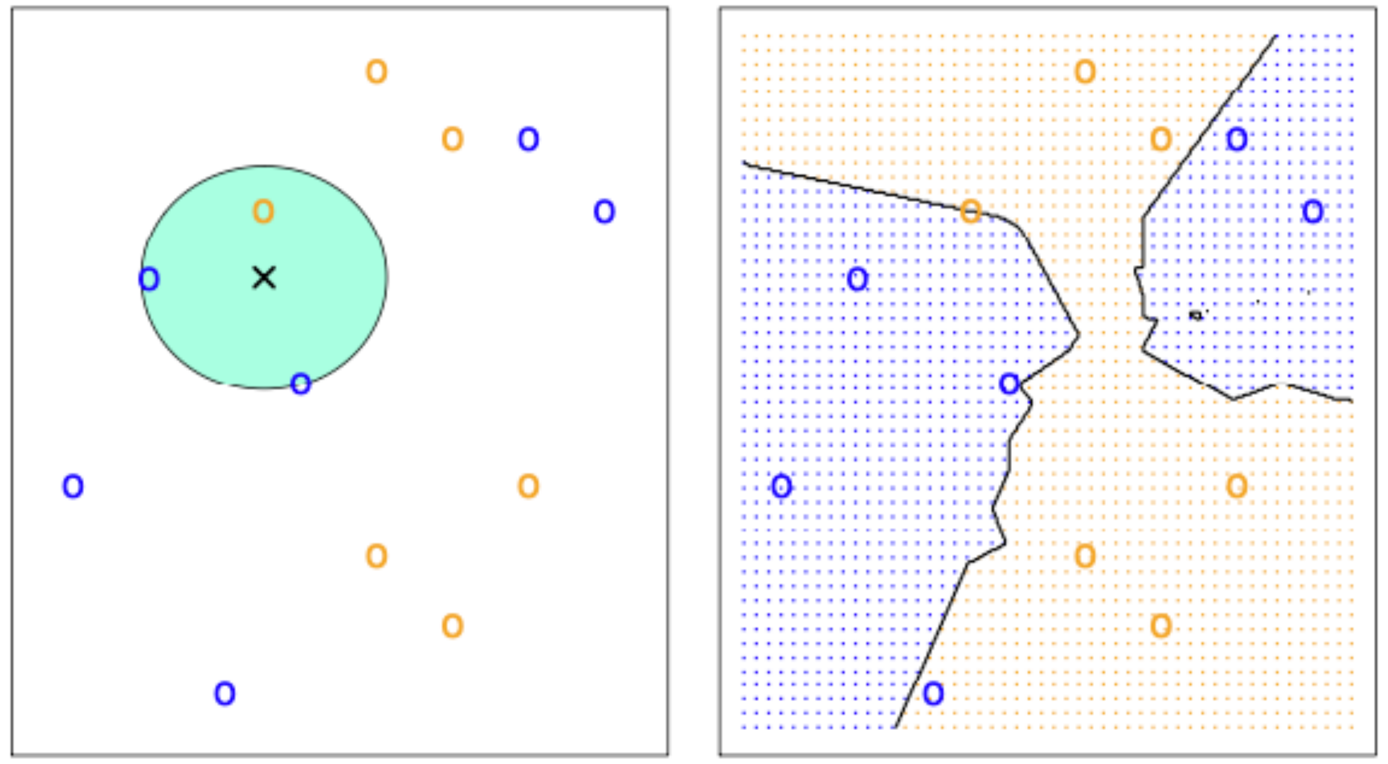


Simulated binary classification data.
Bayes classifier in purple.
E.g., color = stroke type, (X_1, X_2) = CT image.

Example: K-nearest neighbors

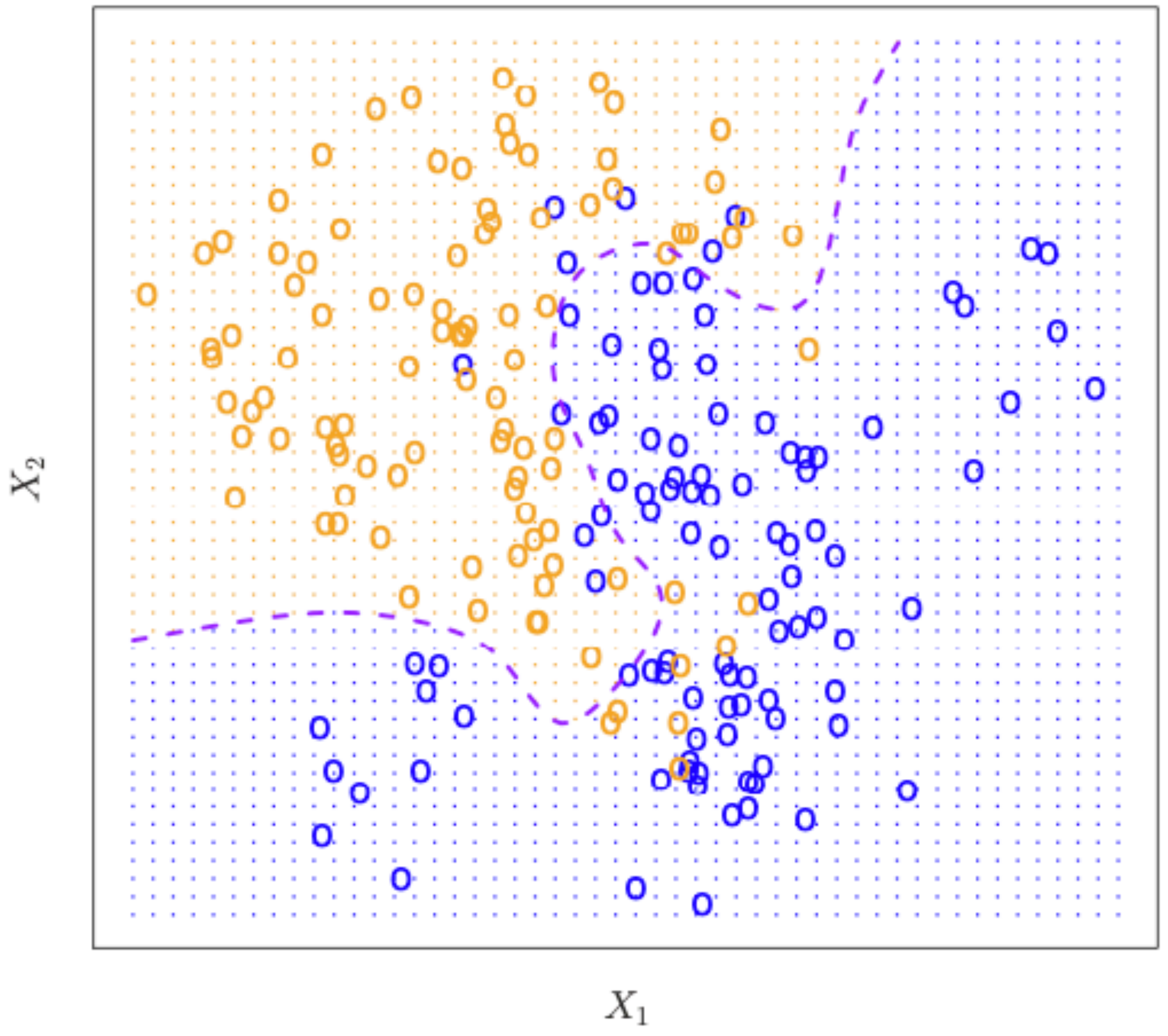


Simulated binary classification data.
Bayes classifier in purple.
E.g., color = stroke type, (X_1, X_2) = CT image.

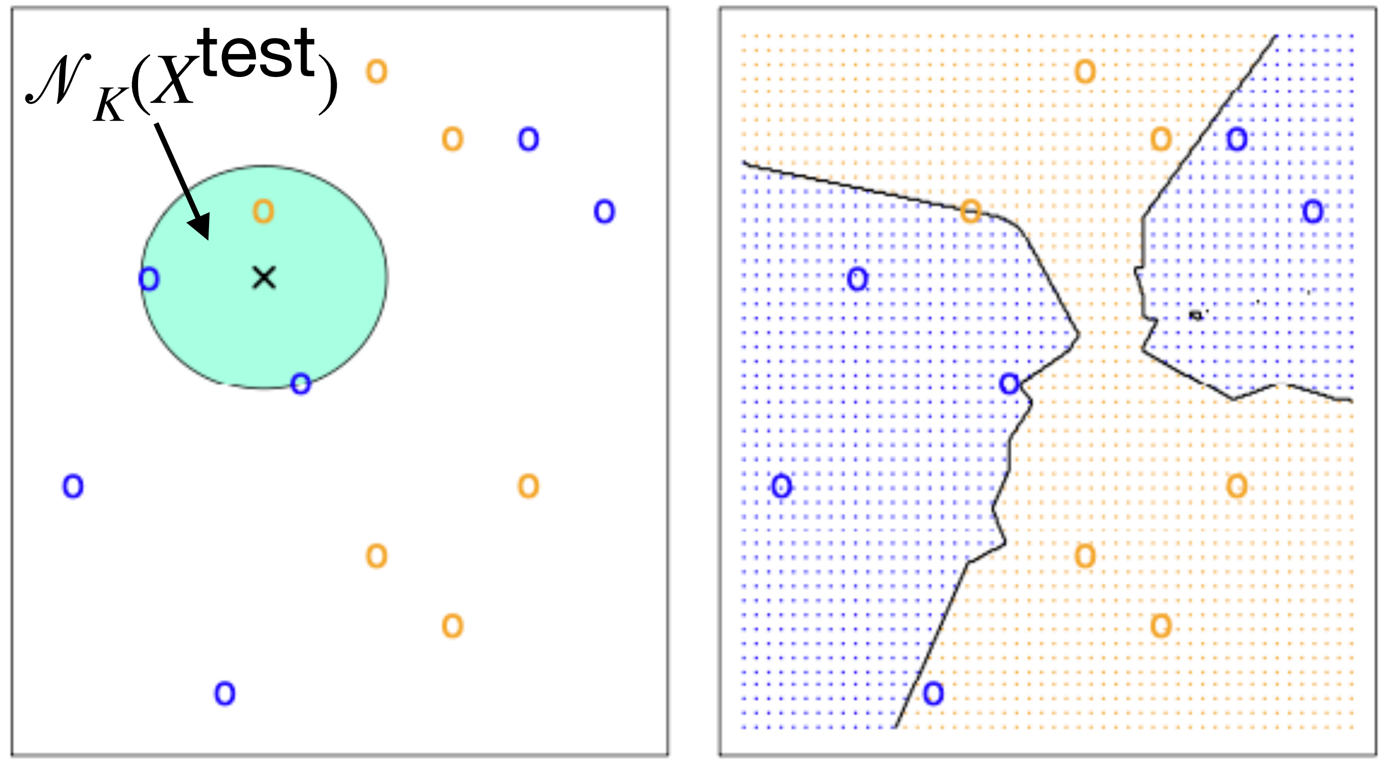


KNN illustration: Classify a test point based on majority vote among 3 nearest neighbors.

Example: K-nearest neighbors



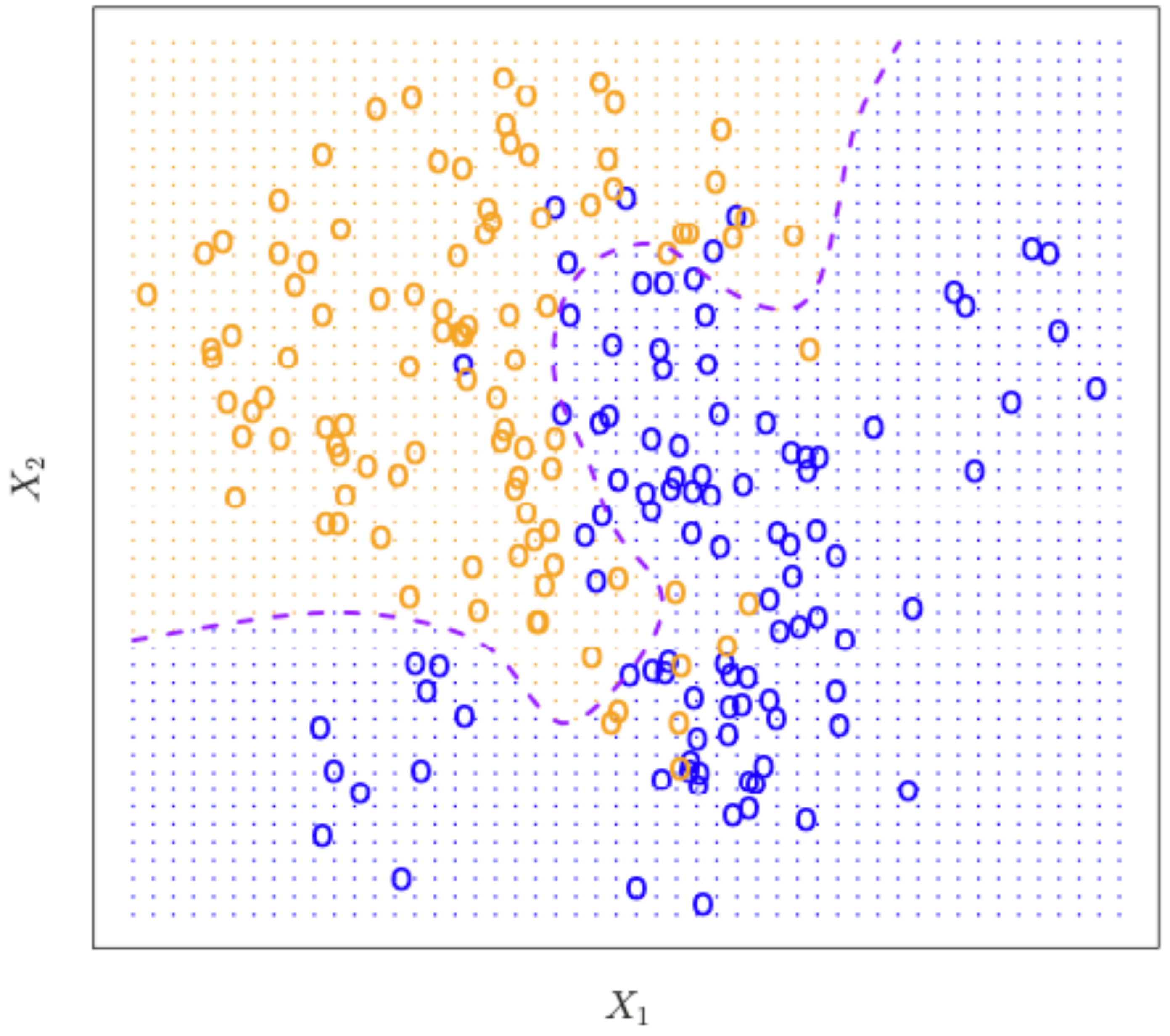
Simulated binary classification data.
Bayes classifier in purple.
E.g., color = stroke type, (X_1, X_2) = CT image.



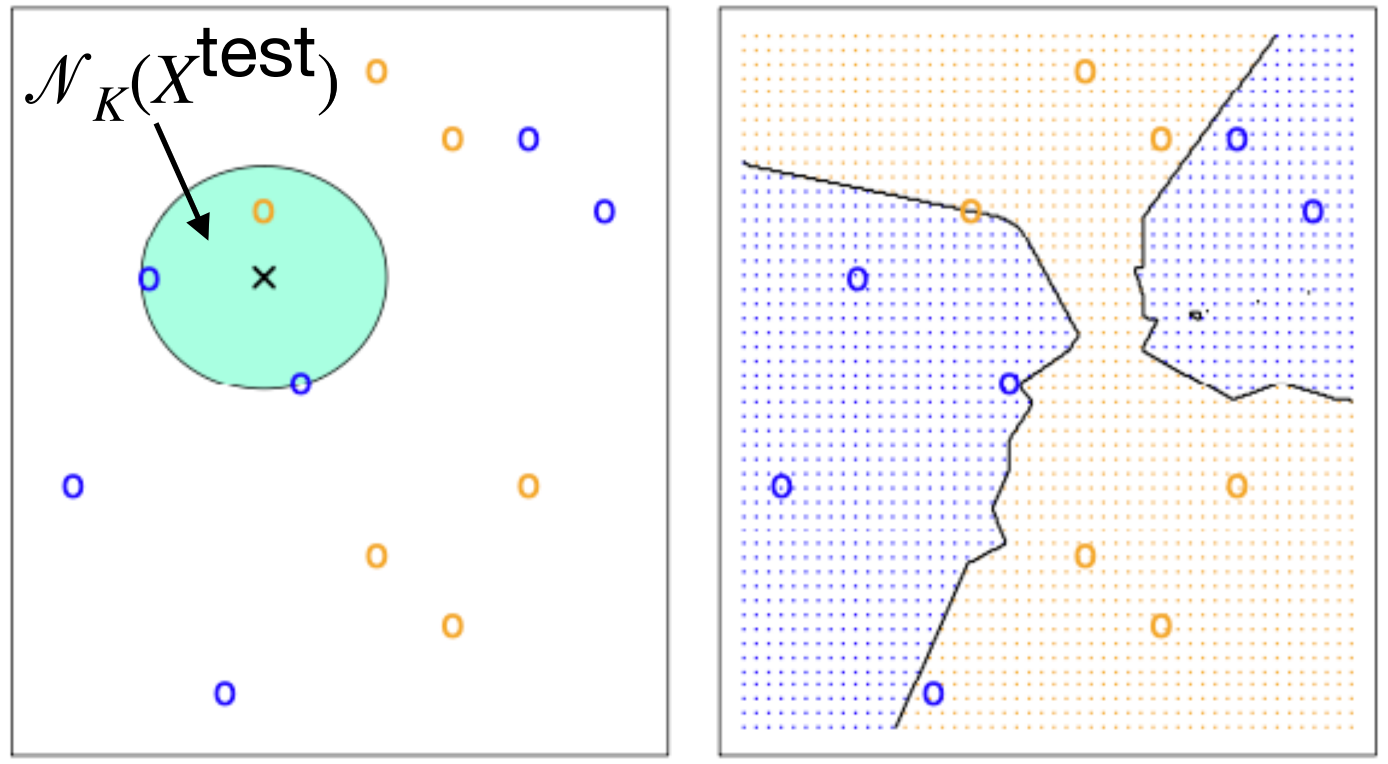
KNN illustration: Classify a test point based on majority vote among 3 nearest neighbors.

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

Example: K-nearest neighbors

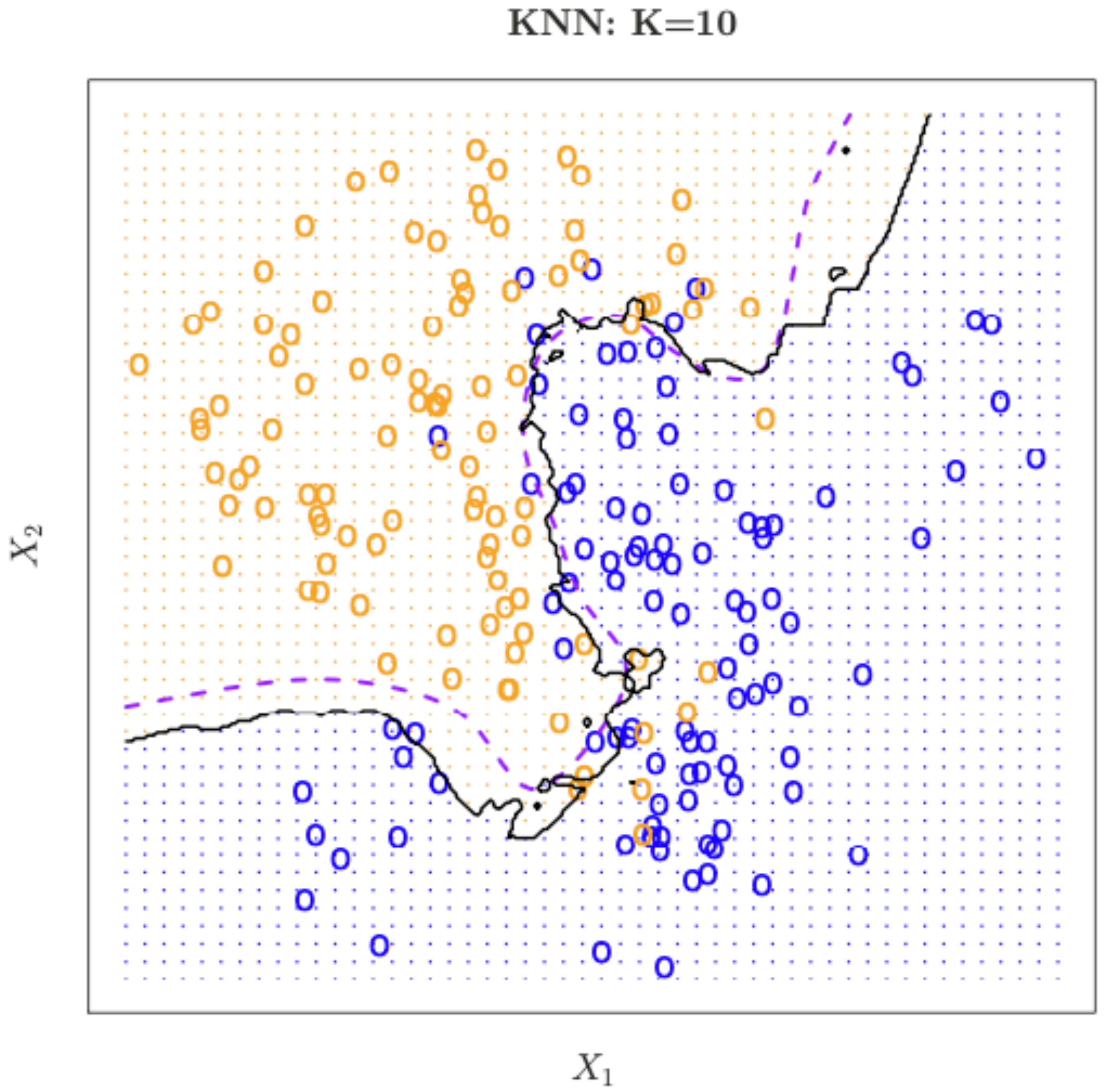


Simulated binary classification data.
Bayes classifier in purple.
E.g., color = stroke type, (X_1, X_2) = CT image.



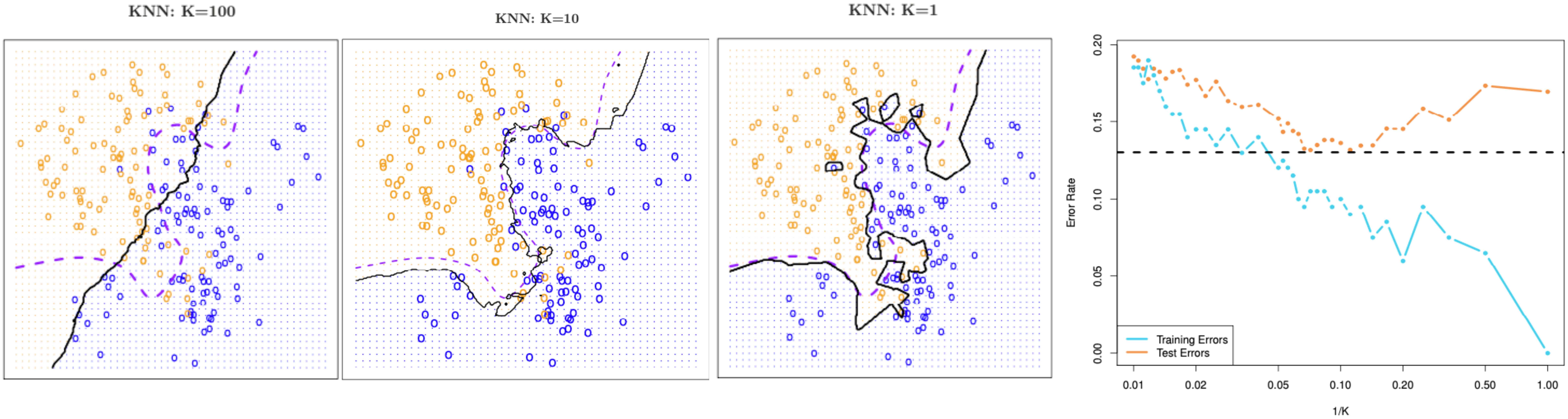
KNN illustration: Classify a test point based on majority vote among 3 nearest neighbors.

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



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

Bias-variance tradeoff

Mathematically: Applies only to continuous response variables and MSE.

Bias-variance tradeoff

Mathematically: Applies only to continuous response variables and MSE.

Intuitively: Applies to any prediction problem, including classification.

Bias-variance tradeoff

Mathematically: Applies only to continuous response variables and MSE.

Intuitively: Applies to any prediction problem, including classification.

For the estimate $\widehat{p}(X)$

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

Bias-variance tradeoff

Mathematically: Applies only to continuous response variables and MSE.

Intuitively: Applies to any prediction problem, including classification.

For the estimate $\widehat{p}(X)$

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

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

Bias-variance tradeoff

Mathematically: Applies only to continuous response variables and MSE.

Intuitively: Applies to any prediction problem, including classification.

For the estimate $\widehat{p}(X)$

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

- Variance: Var$[\widehat{p}(X)]$

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

Bias-variance tradeoff

Mathematically: Applies only to continuous response variables and MSE.

Intuitively: Applies to any prediction problem, including classification.

For the estimate $\widehat{p}(X)$

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

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

- Bias: Predict wrong class on average, to the extent $\widehat{p}$ on wrong side of 0.5

Bias-variance tradeoff

Mathematically: Applies only to continuous response variables and MSE.

Intuitively: Applies to any prediction problem, including classification.

For the estimate $\widehat{p}(X)$

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

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

- 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

Bias-variance tradeoff

Mathematically: Applies only to continuous response variables and MSE.

Intuitively: Applies to any prediction problem, including classification.

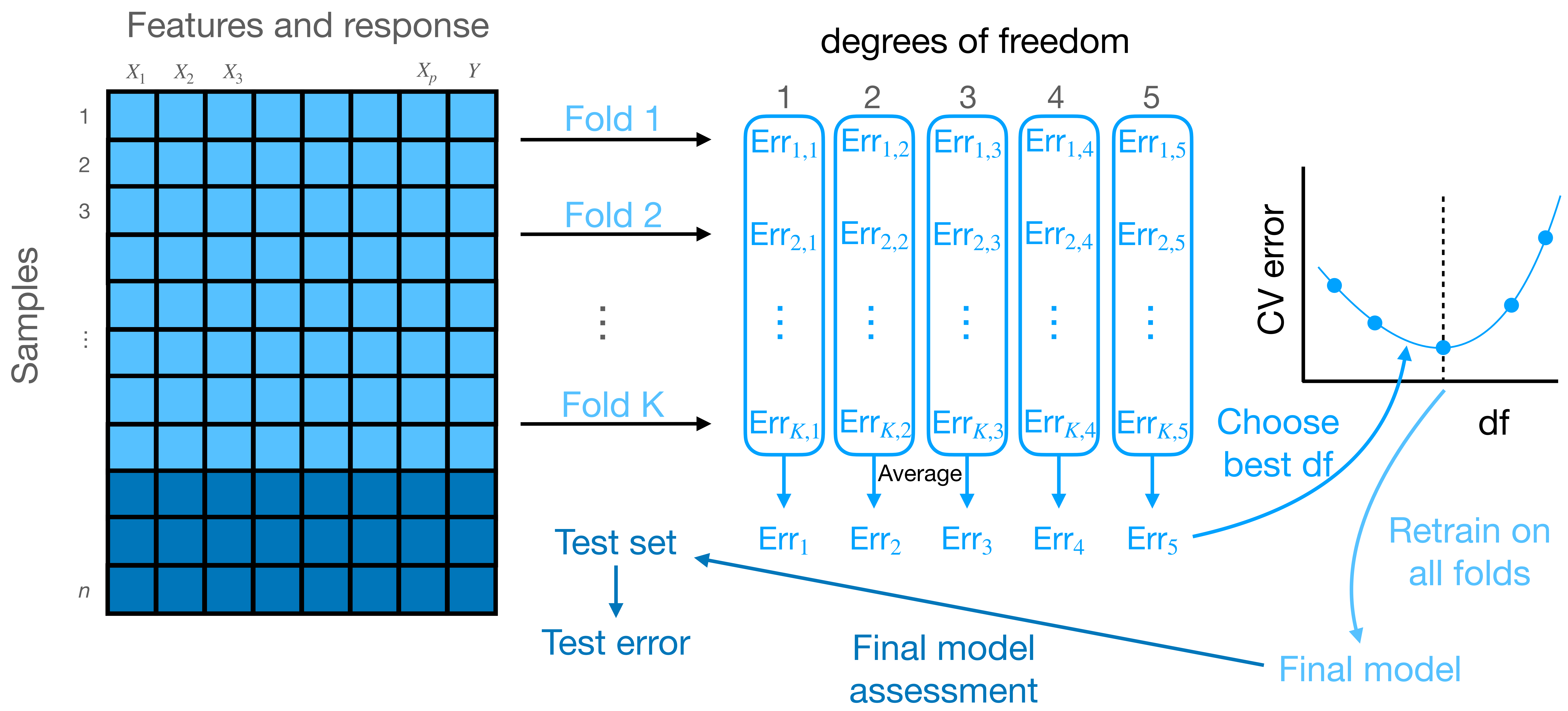
For the estimate $\widehat{p}(X)$

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

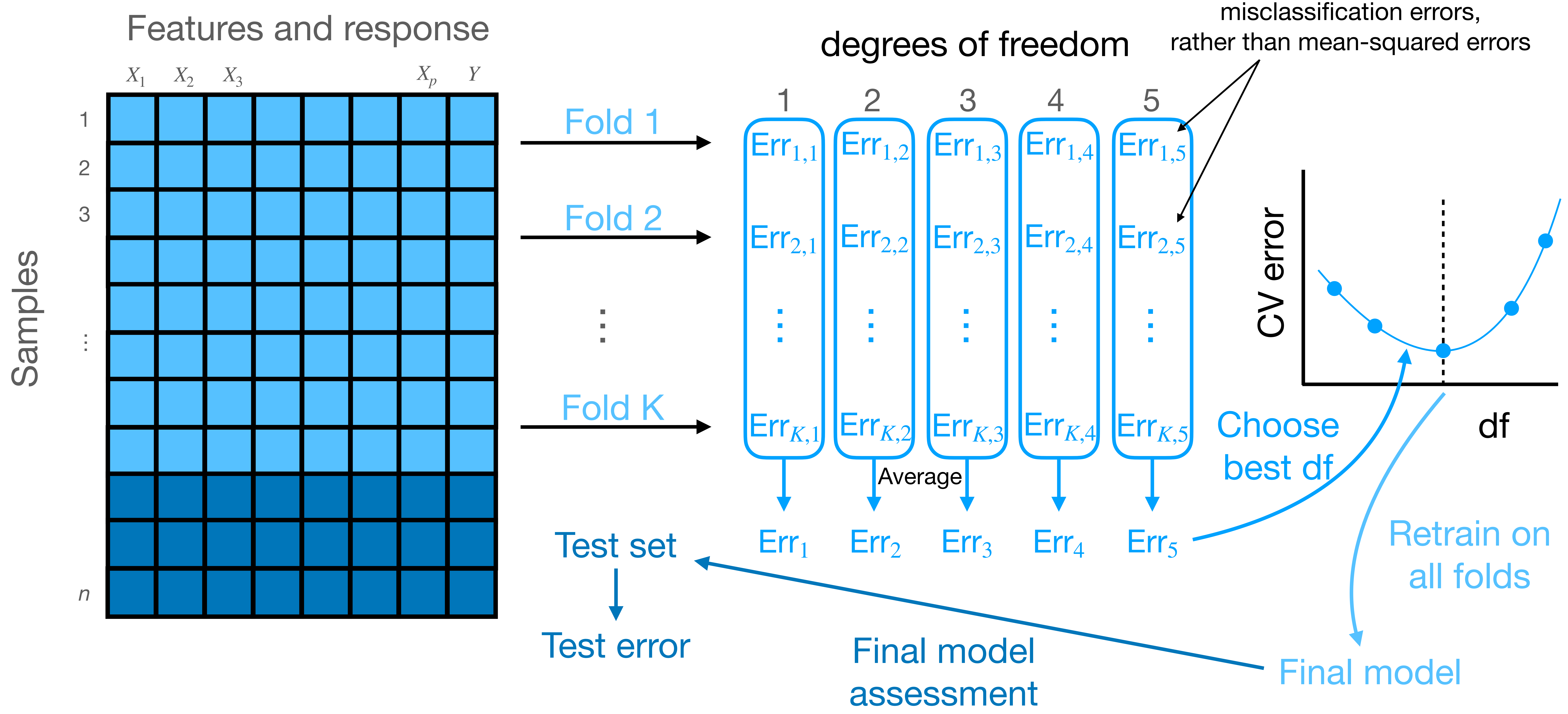
For classifying $\widehat{Y} = I(p(X) \geq 0.5)$

- 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 $0 < \mathbb{P}[Y = 1 \,|\, X] < 1$.

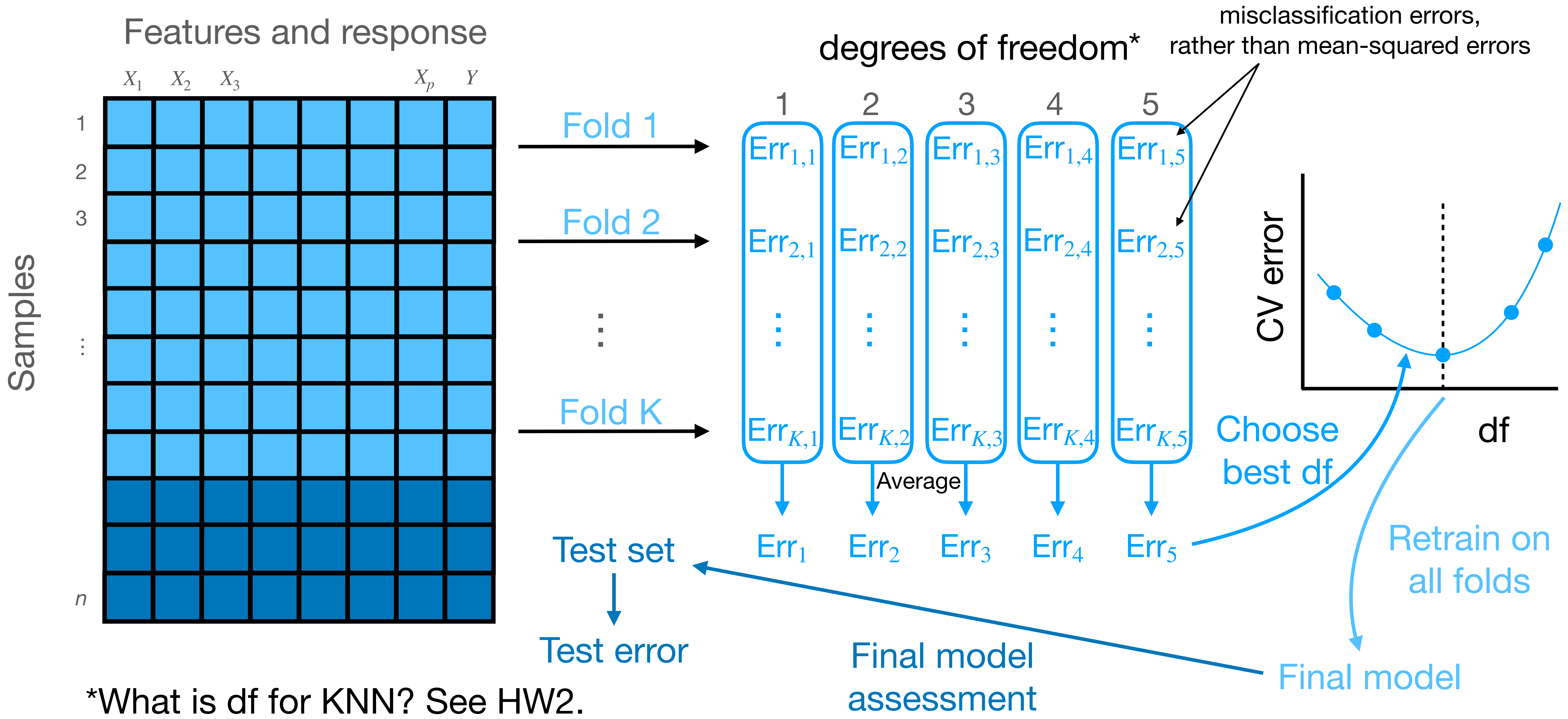
Cross-validation based on misclassification error
(otherwise same as before)



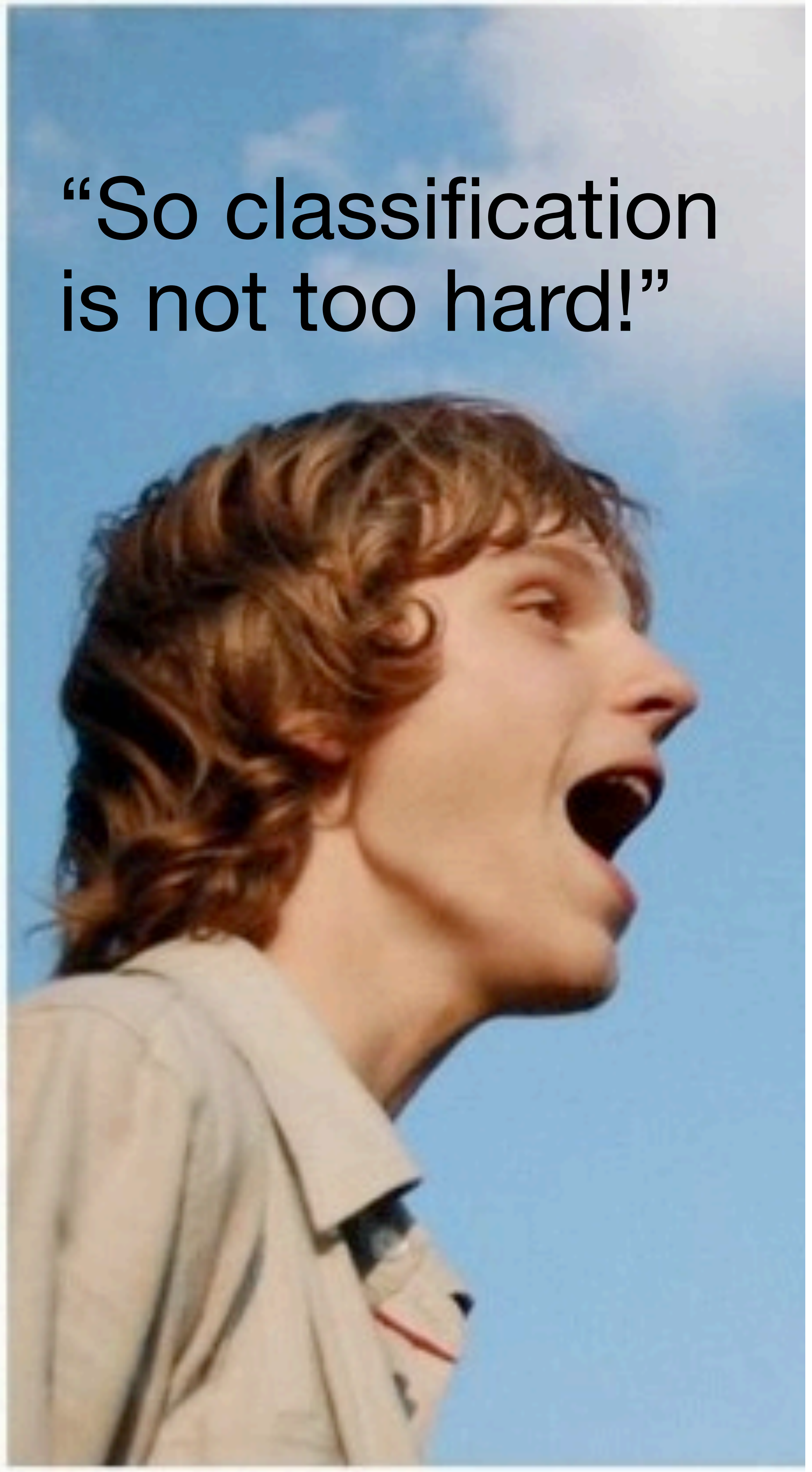
Cross-validation based on misclassification error
(otherwise same as before)

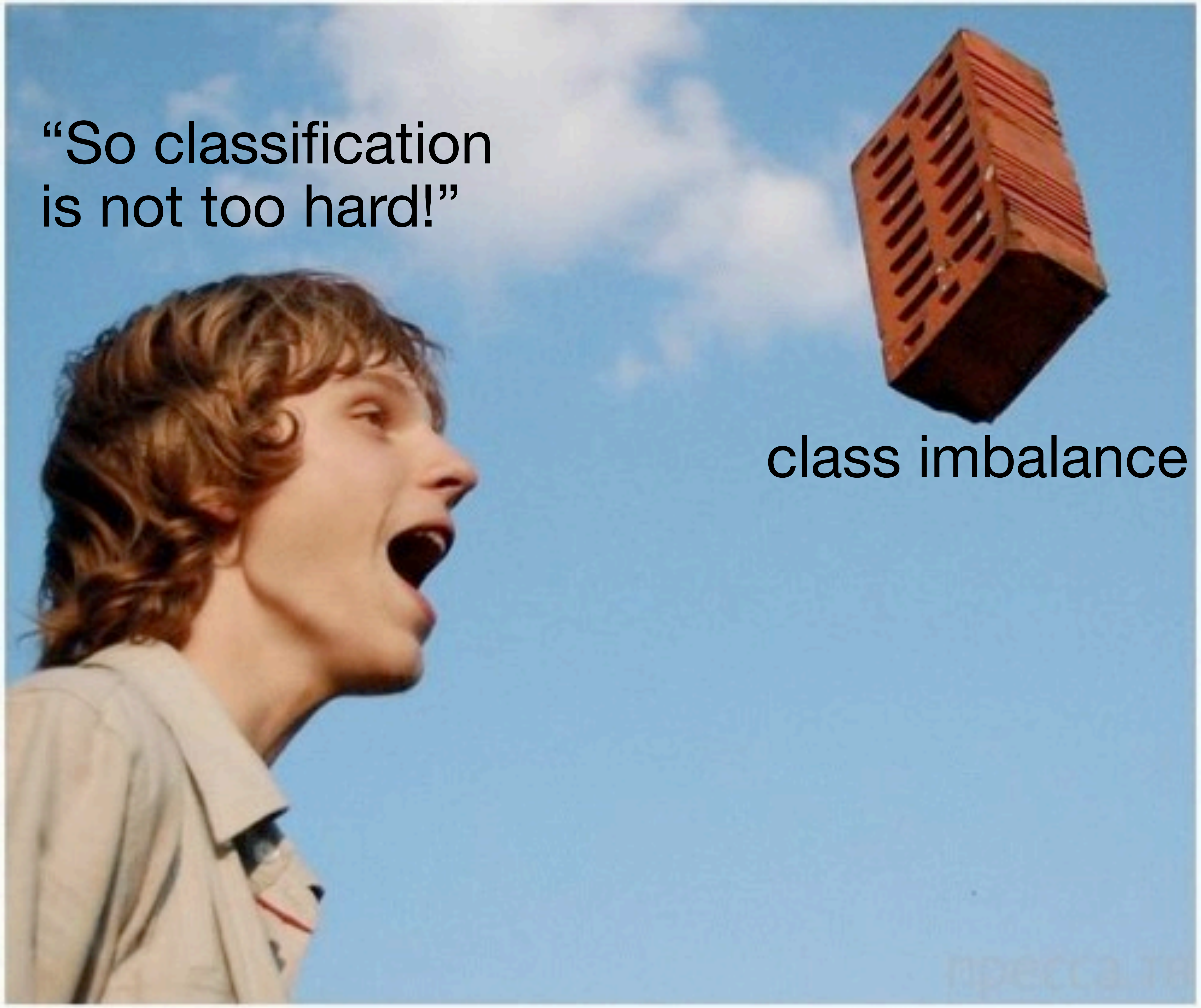


Cross-validation based on misclassification error
(otherwise same as before)



*What is df for KNN? See HW2.


“So classification
is not too hard!”


"So classification
is not too hard!"
class imbalance

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

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.

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%.

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.

Binary classification terminology

Positive: $Y = 1$
(e.g. COVID positive)

Negative: $Y = 0$
(e.g. COVID negative)

Binary classification terminology

Positive: $Y = 1$
(e.g. COVID positive)

Negative: $Y = 0$
(e.g. COVID negative)

	Actually Positive	**Actually Negative**
Predicted Positive	True Positive (TP) (E.g. Sick person testing positive)	False Positive (FP) (E.g. Healthy person testing positive)
Predicted Negative	False negative (FN) (E.g. Sick person testing negative)	True Negative (TN) (E.g. Healthy person testing negative)

Thinking about misclassification costs

Thinking about misclassification costs

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

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)
  $\rightarrow$ drained bank account. Cost: C_{FN} = \$10,000.

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

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_{FN} = \$10,000.

- False alarm of fraud (false positive)
  → annoying text message and/or replaced credit card. Cost: C_{FP} = \$10.

Weighted misclassification error:

$$\frac{1}{N}\sum_{i=1}^{N} w_i \cdot I(\widehat{Y}_i^{\text{test}} \neq Y_i^{\text{test}}), \qquad \text{where} \quad w_i = \begin{cases} C_{\mathrm{FP}} & \text{if } Y_i^{\text{test}} = 0 \\ C_{\mathrm{FN}} & \text{if } Y_i^{\text{test}} = 1 \end{cases}.$$

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:

$$\frac{1}{N}\sum_{i=1}^{N} w_i \cdot I(\hat{Y}_i^{\text{test}} \neq Y_i^{\text{test}}), \qquad \text{where} \quad w_i = \begin{cases} C_{\text{FP}} & \text{if } Y_i^{\text{test}} = 0 \\ C_{\text{FN}} & \text{if } Y_i^{\text{test}} = 1 \end{cases}.$$

w_i are called observation weights; increase penalty for misclassifying positives.

Building observation weights into training

Building observation weights into training

Many machine learning algorithms accommodate observation weights, i.e. seek to optimize the weighted misclassification error.

Building observation weights into training

Many machine learning algorithms accommodate observation weights, i.e. seek to optimize the weighted misclassification error.

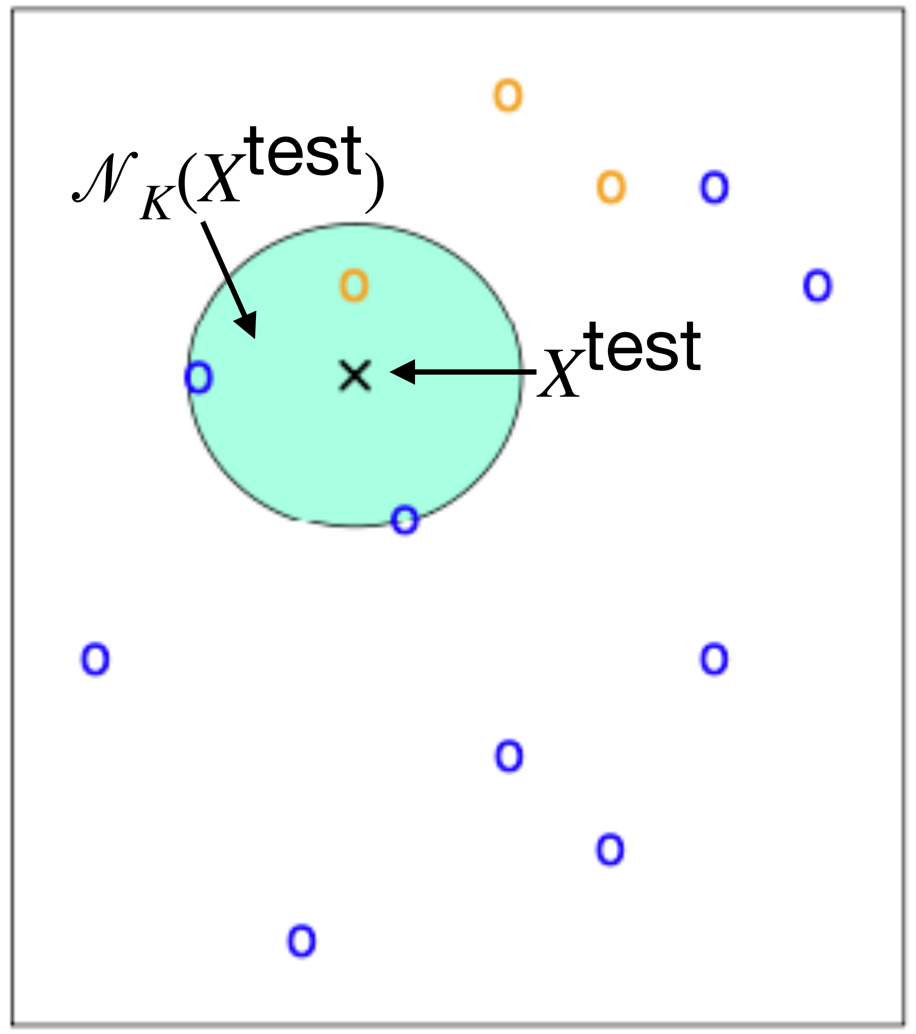
For example, consider weighted K-nearest neighbors:

Building observation weights into training

Many machine learning algorithms accommodate observation weights, i.e. seek to optimize the weighted misclassification error.

For example, consider weighted K-nearest neighbors:

Positive (rare) training points
Negative (common) training points

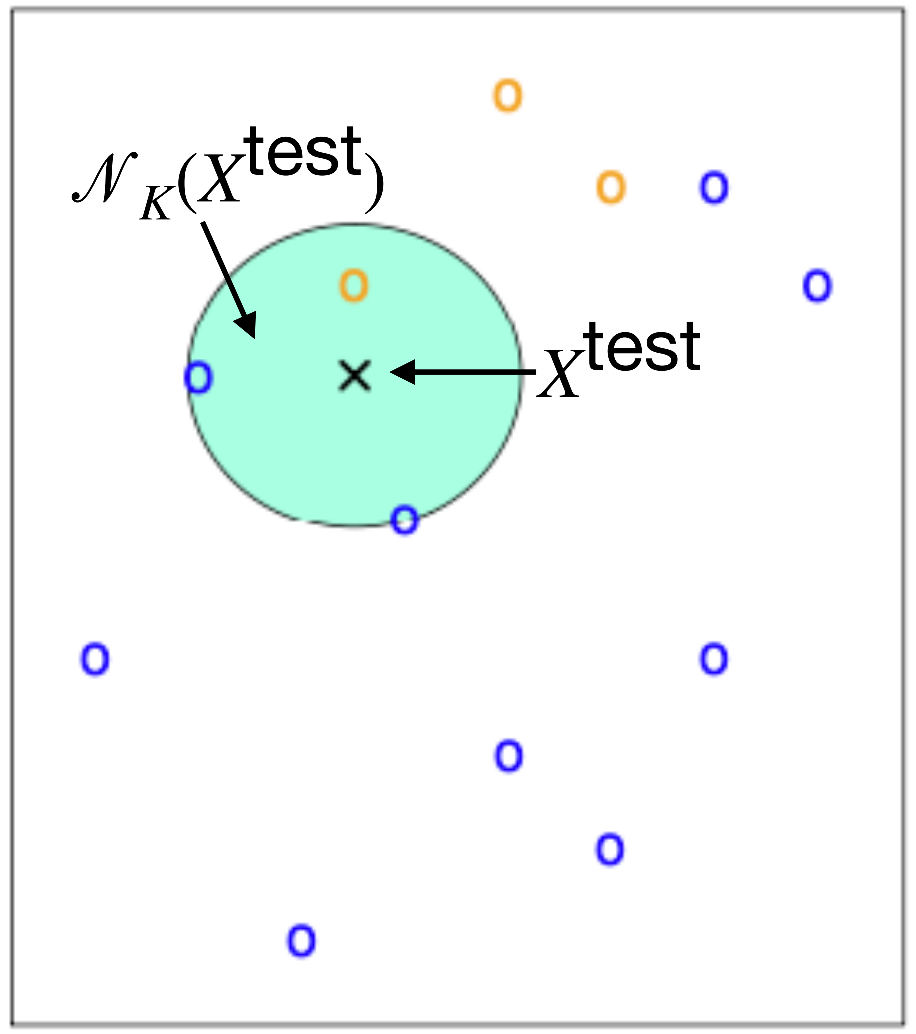


Building observation weights into training

Many machine learning algorithms accommodate observation weights, i.e. seek to optimize the weighted misclassification error.

For example, consider weighted K-nearest neighbors:

Positive (rare) training points
Negative (common) training points

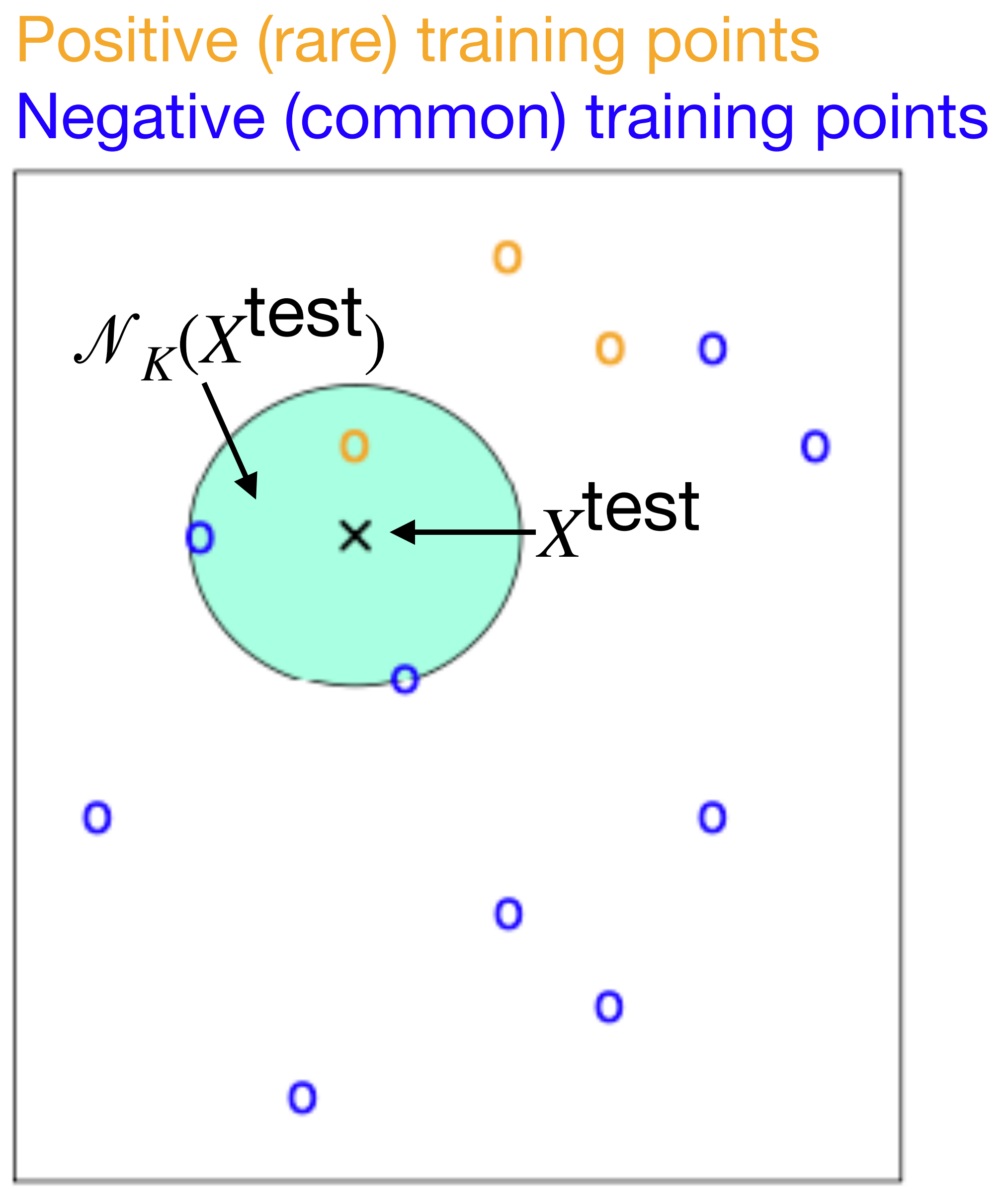


$$\hat{p}(X^{\text{test}}) = \frac{\sum_{i \in \mathcal{N}_K} w_i \cdot I(Y_i^{\text{train}} = 1)}{\sum_{i \in \mathcal{N}_K} w_i}$$

Building observation weights into training

Many machine learning algorithms accommodate observation weights, i.e. seek to optimize the weighted misclassification error.

For example, consider weighted K-nearest neighbors:



$$w_i = \begin{cases} C_{\text{FP}} & \text{if } Y_i^{\text{train}} = 0 \\ C_{\text{FN}} & \text{if } Y_i^{\text{train}} = 1 \end{cases}$$

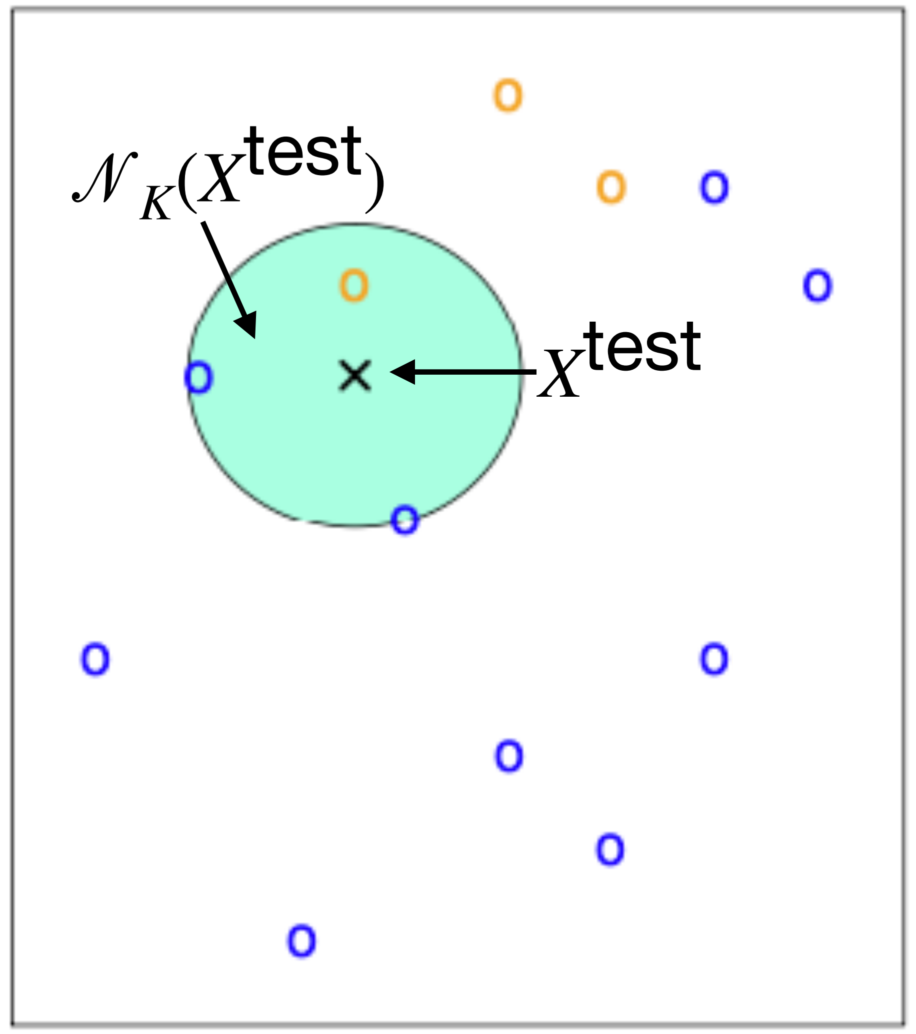
$$\hat{p}(X^{\text{test}}) = \frac{\sum_{i \in \mathcal{N}_K} w_i \cdot I(Y_i^{\text{train}} = 1)}{\sum_{i \in \mathcal{N}_K} w_i}$$

Building observation weights into training

Many machine learning algorithms accommodate observation weights, i.e. seek to optimize the weighted misclassification error.

For example, consider weighted K-nearest neighbors:

Positive (rare) training points
Negative (common) training points



$$w_i = \begin{cases} C_{\text{FP}} & \text{if } Y_i^{\text{train}} = 0 \\ C_{\text{FN}} & \text{if } Y_i^{\text{train}} = 1 \end{cases}$$

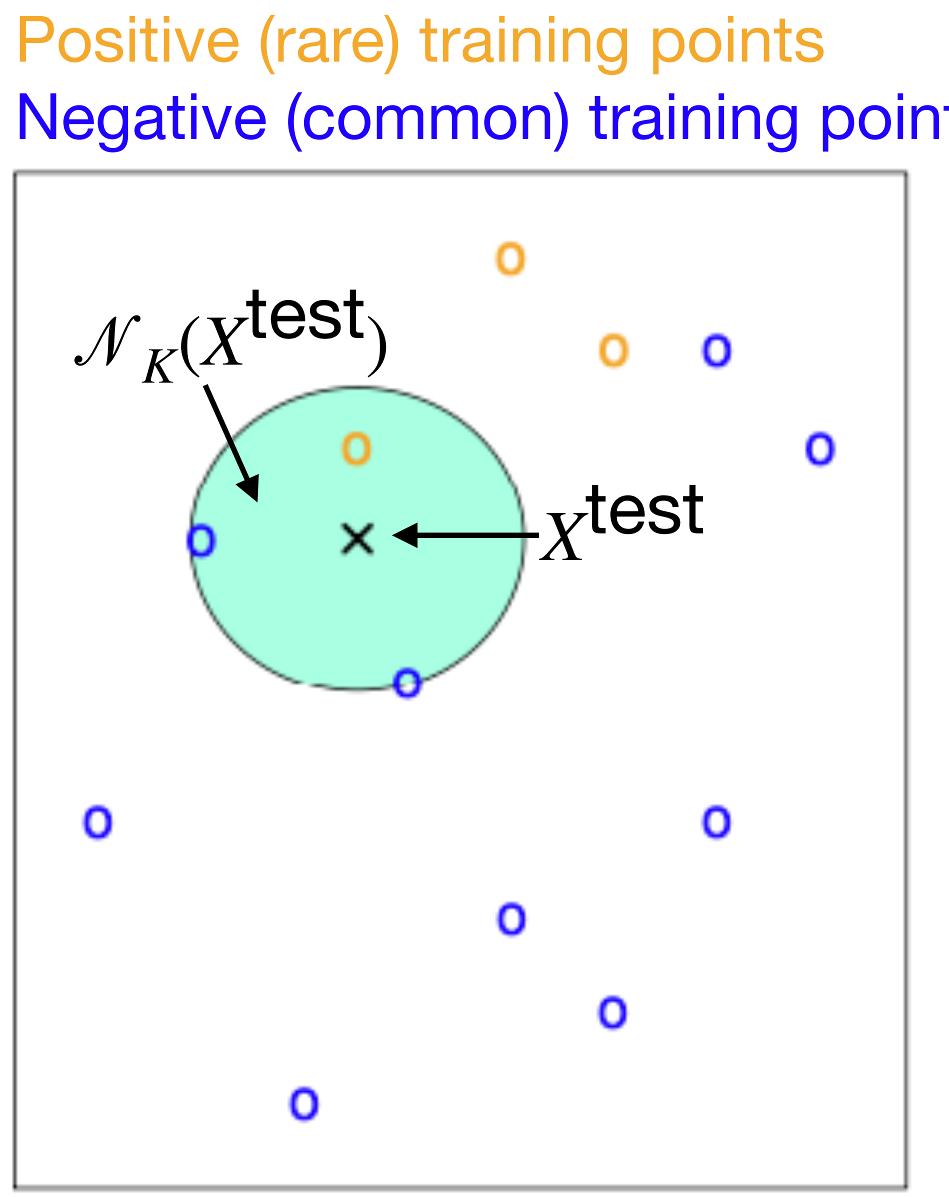
Upweight positive (rare) training cases so that the classifier predicts positive more often.

$$\hat{p}(X^{\text{test}}) = \frac{\sum_{i \in \mathcal{N}_K} w_i \cdot I(Y_i^{\text{train}} = 1)}{\sum_{i \in \mathcal{N}_K} w_i}$$

Building observation weights into training

Many machine learning algorithms accommodate observation weights, i.e. seek to optimize the weighted misclassification error.

For example, consider weighted K-nearest neighbors:



$$w_i = \begin{cases} C_{\text{FP}} & \text{if } Y_i^{\text{train}} = 0 \\ C_{\text{FN}} & \text{if } Y_i^{\text{train}} = 1 \end{cases}$$

Upweight positive (rare) training cases so that the classifier predicts positive more often.

$$\hat{p}(X^{\text{test}}) = \frac{\sum_{i \in \mathcal{N}_K} w_i \cdot I(Y_i^{\text{train}} = 1)}{\sum_{i \in \mathcal{N}_K} w_i}$$

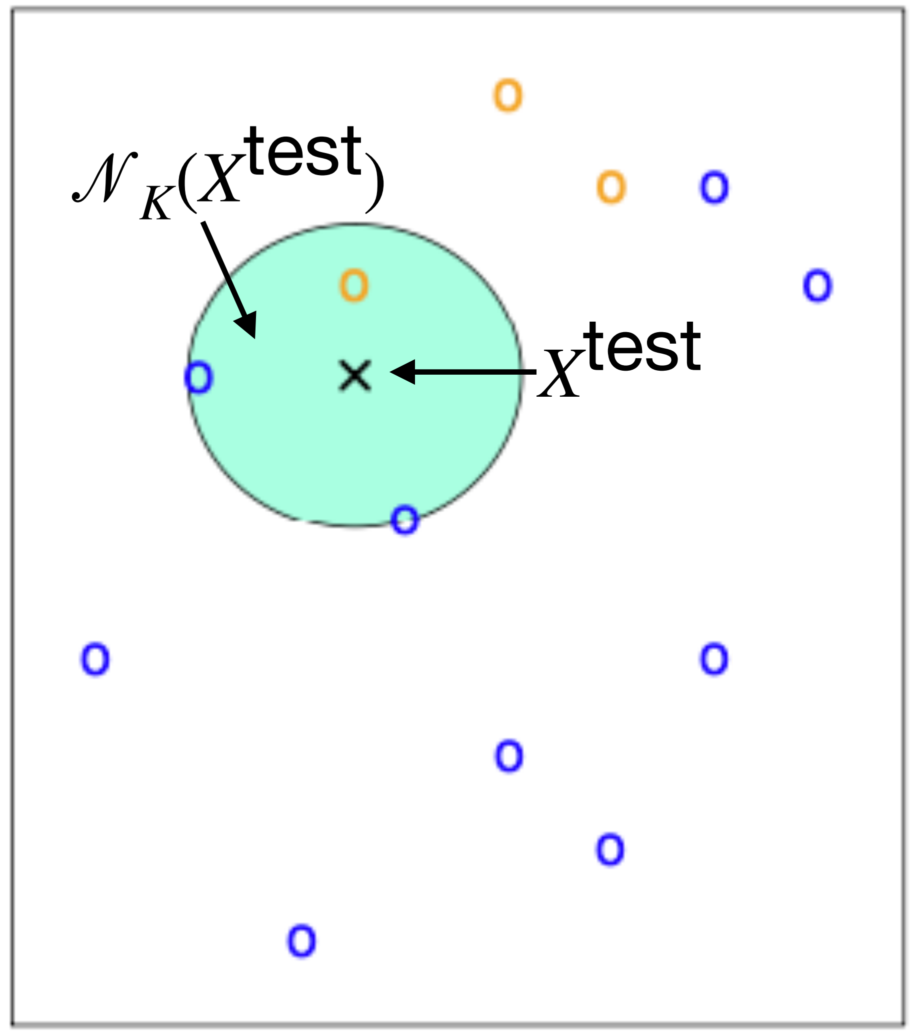
w_{blue}	w_{yellow}	$\hat{p}(X_{\text{test}})$	Predicted class
1	1	1/3	Blue

Building observation weights into training

Many machine learning algorithms accommodate observation weights, i.e. seek to optimize the weighted misclassification error.

For example, consider weighted K-nearest neighbors:

Positive (rare) training points
Negative (common) training points



$$w_i = \begin{cases} C_{\text{FP}} & \text{if } Y_i^{\text{train}} = 0 \\ C_{\text{FN}} & \text{if } Y_i^{\text{train}} = 1 \end{cases}$$

Upweight positive (rare) training cases so that the classifier predicts positive more often.

$$\hat{p}(X^{\text{test}}) = \frac{\sum_{i \in \mathcal{N}_K} w_i \cdot I(Y_i^{\text{train}} = 1)}{\sum_{i \in \mathcal{N}_K} w_i}$$

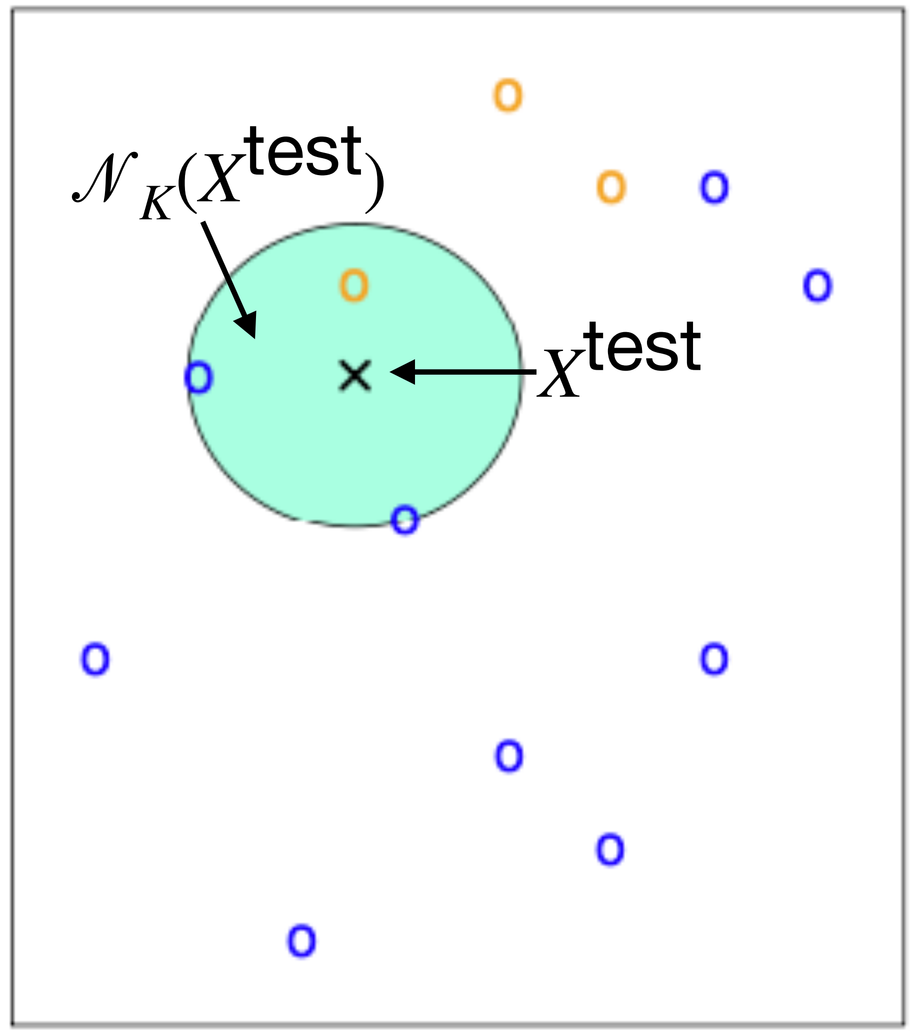
w_{blue}	w_{yellow}	$\hat{p}(X_{\text{test}})$	Predicted class
1	1	1/3	Blue
1	2	1/2	Yellow

Building observation weights into training

Many machine learning algorithms accommodate observation weights, i.e. seek to optimize the weighted misclassification error.

For example, consider weighted K-nearest neighbors:

Positive (rare) training points
Negative (common) training points



$$w_i = \begin{cases} C_{\text{FP}} & \text{if } Y_i^{\text{train}} = 0 \\ C_{\text{FN}} & \text{if } Y_i^{\text{train}} = 1 \end{cases}$$

Upweight positive (rare) training cases so that the classifier predicts positive more often.

$$\hat{p}(X^{\text{test}}) = \frac{\sum_{i \in \mathcal{N}_K} w_i \cdot I(Y_i^{\text{train}} = 1)}{\sum_{i \in \mathcal{N}_K} w_i}$$

w_{blue}	w_{yellow}	$\hat{p}(X_{\text{test}})$	Predicted class
1	1	1/3	Blue
1	2	1/2	Yellow

(As though two yellows in circle)

Building observation weights into training

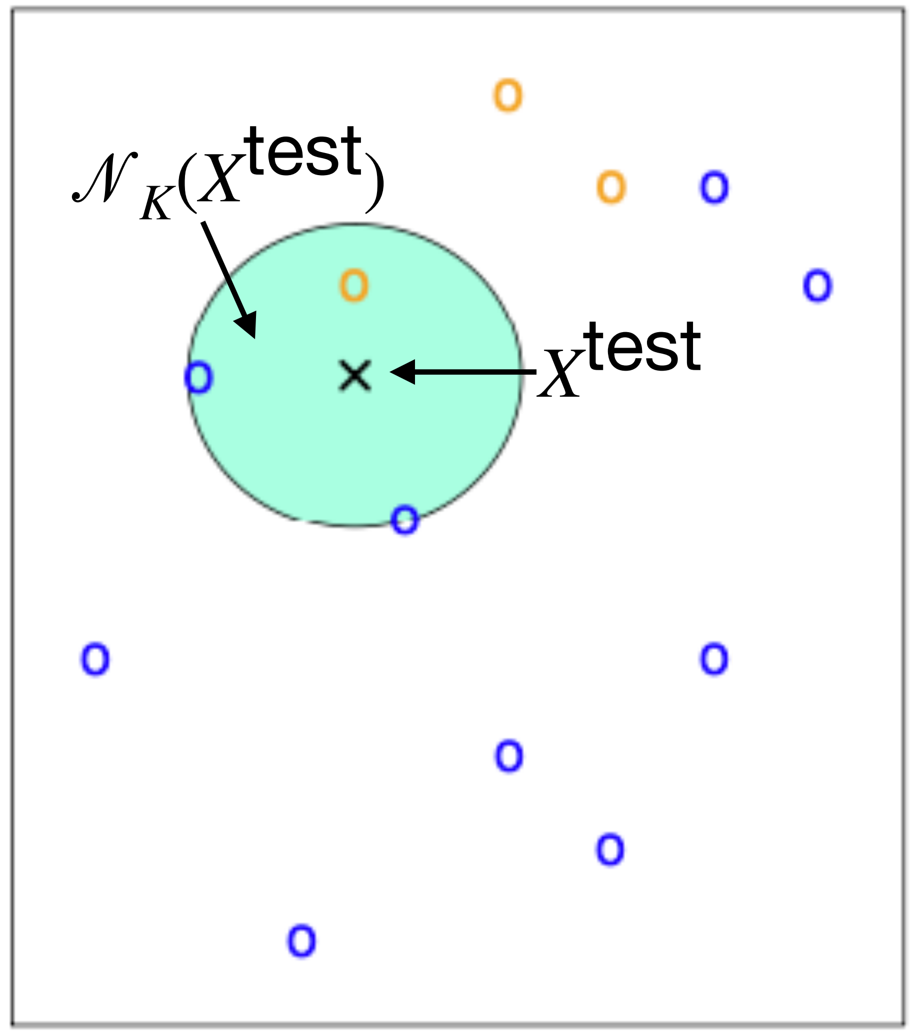
Many machine learning algorithms accommodate observation weights, i.e. seek to optimize the weighted misclassification error.

For example, consider weighted K-nearest neighbors:

$$\begin{cases} C_{\text{FP}} & \text{if } Y_i^{\text{train}} = 0 \\ C_{\text{FN}} & \text{if } Y_i^{\text{train}} = 1 \end{cases}$$

Upweight positive (rare) training cases so that the classifier predicts positive more often.

Positive (rare) training points
Negative (common) training points



$$\hat{p}(X^{\text{test}}) = \frac{\sum_{i \in \mathcal{N}_K} w_i \cdot I(Y_i^{\text{train}} = 1)}{\sum_{i \in \mathcal{N}_K} w_i}$$

w_{blue}	w_{yellow}	$\hat{p}(X_{\text{test}})$	Predicted class
1	1	1/3	Blue
1	2	1/2	Yellow
1	3	3/5	Yellow

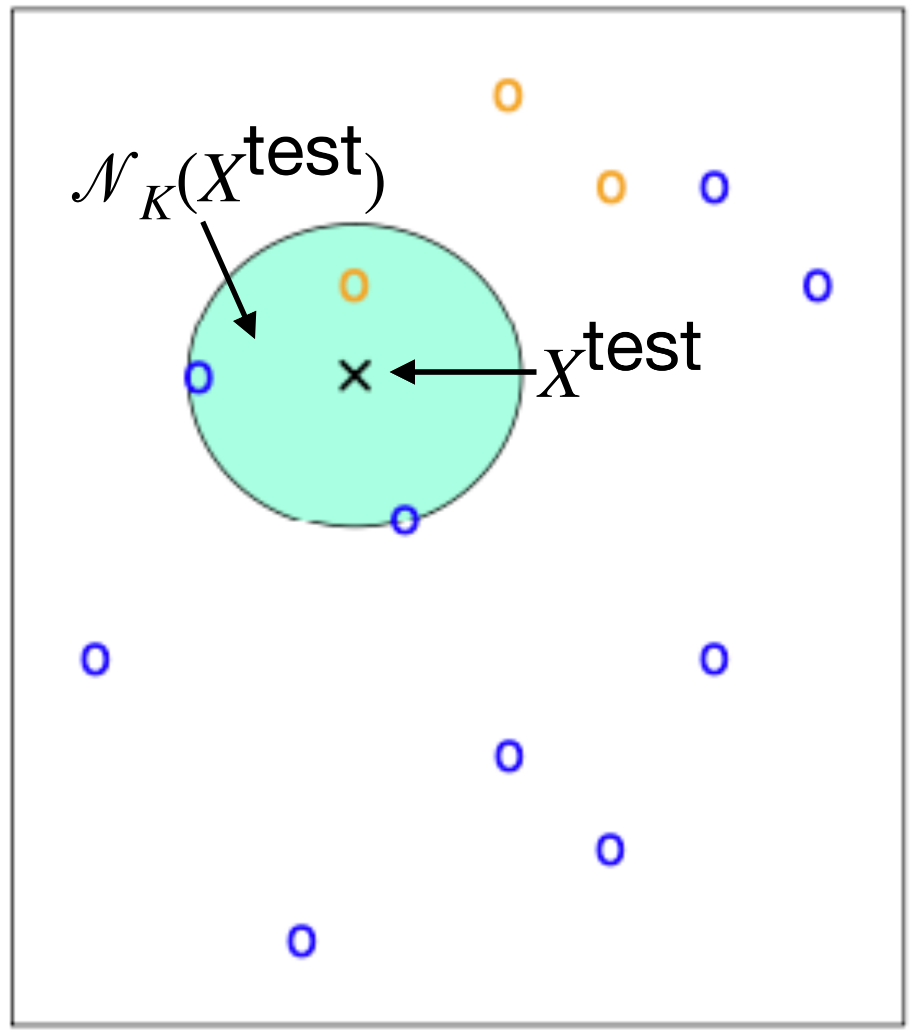
(As though two yellows in circle)

Building observation weights into training

Many machine learning algorithms accommodate observation weights, i.e. seek to optimize the weighted misclassification error.

For example, consider weighted K-nearest neighbors:

Positive (rare) training points
Negative (common) training points



$$w_i = \begin{cases} C_{\text{FP}} & \text{if } Y_i^{\text{train}} = 0 \\ C_{\text{FN}} & \text{if } Y_i^{\text{train}} = 1 \end{cases}$$

Upweight positive (rare) training cases so that the classifier predicts positive more often.

$$\hat{p}(X^{\text{test}}) = \frac{\sum_{i \in \mathcal{N}_K} w_i \cdot I(Y_i^{\text{train}} = 1)}{\sum_{i \in \mathcal{N}_K} w_i}$$

w_{blue}	w_{yellow}	$\hat{p}(X_{\text{test}})$	Predicted class	
1	1	1/3	Blue	
1	2	1/2	Yellow	(As though two yellows in circle)
1	3	3/5	Yellow	(As though three yellows in circle)

Cross-validation with imbalanced classes

Cross-validation with imbalanced classes

- Split the data into folds after stratifying by the response class.

Cross-validation with imbalanced classes

- Split the data into folds after stratifying by the response class.
- Use weighted misclassification error when assessing models on in-fold data.

Evaluating classification errors on a test set

Evaluating classification errors on a test set

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

Evaluating classification errors on a test set

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

Another way of assessing classification performance—without quantifying costs—is the confusion matrix and associated metrics (e.g. precision and recall).

The confusion matrix and associated metrics

Confusion matrix

	Actually Positive	Actually Negative
Predicted Positive	10 True Positives (TP) (E.g. Sick person testing positive)	20 False Positives (FP) (E.g. Healthy person testing positive)
Predicted Negative	40 False negatives (FN) (E.g. Sick person testing negative)	30 True Negatives (TN) (E.g. Healthy person testing negative)

The confusion matrix and associated metrics

Confusion matrix

	Actually Positive	Actually Negative
Predicted Positive	10 True Positives (TP) (E.g. Sick person testing positive)	20 False Positives (FP) (E.g. Healthy person testing positive)
Predicted Negative	40 False negatives (FN) (E.g. Sick person testing negative)	30 True Negatives (TN) (E.g. Healthy person testing negative)

Metrics based on confusion matrix

The confusion matrix and associated metrics

Confusion matrix

	Actually Positive	Actually Negative
Predicted Positive	10 True Positives (TP) (E.g. Sick person testing positive)	20 False Positives (FP) (E.g. Healthy person testing positive)
Predicted Negative	40 False negatives (FN) (E.g. Sick person testing negative)	30 True Negatives (TN) (E.g. Healthy person testing negative)

Metrics based on confusion matrix

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{10}{30}$$

The confusion matrix and associated metrics

Confusion matrix

	Actually Positive	Actually Negative
Predicted Positive	10 True Positives (TP) (E.g. Sick person testing positive)	20 False Positives (FP) (E.g. Healthy person testing positive)
Predicted Negative	40 False negatives (FN) (E.g. Sick person testing negative)	30 True Negatives (TN) (E.g. Healthy person testing negative)

Metrics based on confusion matrix

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{10}{30}$$

What fraction of predicted positives are actually positive?

The confusion matrix and associated metrics

Confusion matrix

	Actually Positive	Actually Negative
Predicted Positive	10 True Positives (TP) (E.g. Sick person testing positive)	20 False Positives (FP) (E.g. Healthy person testing positive)
Predicted Negative	40 False negatives (FN) (E.g. Sick person testing negative)	30 True Negatives (TN) (E.g. Healthy person testing negative)

Metrics based on confusion matrix

$$\text{Precision} = \frac{\text{TP}}{\text{TP + FP}} = \frac{10}{30}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP + FN}} = \frac{10}{50}$$

The confusion matrix and associated metrics

Confusion matrix

	Actually Positive	Actually Negative
Predicted Positive	10 True Positives (TP) (E.g. Sick person testing positive)	20 False Positives (FP) (E.g. Healthy person testing positive)
Predicted Negative	40 False negatives (FN) (E.g. Sick person testing negative)	30 True Negatives (TN) (E.g. Healthy person testing negative)

Metrics based on confusion matrix

$$\text{Precision} = \frac{\text{TP}}{\text{TP + FP}} = \frac{10}{30}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP + FN}} = \frac{10}{50}$$

What fraction of actual positives are predicted positive?

The confusion matrix and associated metrics

Confusion matrix

	Actually Positive	Actually Negative
Predicted Positive	10 True Positives (TP) (E.g. Sick person testing positive)	20 False Positives (FP) (E.g. Healthy person testing positive)
Predicted Negative	40 False negatives (FN) (E.g. Sick person testing negative)	30 True Negatives (TN) (E.g. Healthy person testing negative)

Metrics based on confusion matrix

$$\text{Precision} = \frac{\text{TP}}{\text{TP + FP}} = \frac{10}{30}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP + FN}} = \frac{10}{50}$$

$$\text{F-score} = \left(\frac{1}{2}\left(\frac{1}{\text{prec.}} + \frac{1}{\text{rec.}}\right)\right)^{-1} = \frac{1}{4}$$

The confusion matrix and associated metrics

Confusion matrix

	Actually Positive	Actually Negative
Predicted Positive	10 True Positives (TP) (E.g. Sick person testing positive)	20 False Positives (FP) (E.g. Healthy person testing positive)
Predicted Negative	40 False negatives (FN) (E.g. Sick person testing negative)	30 True Negatives (TN) (E.g. Healthy person testing negative)

Metrics based on confusion matrix

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{10}{30}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{10}{50}$$

$$\text{F-score} = \left(\frac{1}{2}\left(\frac{1}{\text{prec.}} + \frac{1}{\text{rec.}}\right)\right)^{-1} = \frac{1}{4}$$

One number summarizing performance of the classifier.

Summary

Summary

- Classification problem is similar in some ways to regression; different in others.

Summary

- Classification problem is similar in some ways to regression; different in others.
- Classification done by estimating $\mathbb{P}[Y = 1 \mid X]$, thresholding at 0.5 (e.g. KNN).

Summary

- Classification problem is similar in some ways to regression; different in others.
- Classification 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.

Summary

- Classification problem is similar in some ways to regression; different in others.
- Classification done by estimating $\mathbb{P}[Y = 1 \,|\, X]$, thresholding at 0.5 (e.g. KNN).
- The bias-variance tradeoff carries over intuitively, but not mathematically.
- The misclassification error not a good metric for problems when different misclassifications have different costs; often the case with imbalanced classes.

Summary

- Classification problem is similar in some ways to regression; different in others.
- Classification 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.
- The misclassification error not a good metric for problems when different misclassifications have different costs; often the case with imbalanced classes.
- Differential misclassification costs can be remedied by building observations weights into training.

Summary

- Classification problem is similar in some ways to regression; different in others.
- Classification done by estimating $\mathbb{P}[Y = 1 \,|\, X]$, thresholding at 0.5 (e.g. KNN).
- The bias-variance tradeoff carries over intuitively, but not mathematically.
- The misclassification error not a good metric for problems when different misclassifications have different costs; often the case with imbalanced classes.
- Differential misclassification costs can be remedied by building observations weights into training.
- Performance metrics for classifiers include the weighted misclassification error and confusion matrix based metrics like precision and recall.