

# Model Evaluation

Classification quality metrics, prediction error, cross-validation

Machine Learning and Data Mining, 2025

Majid Sohrabi

National Research University Higher School of Economics



MMCP

October 01, 2025

# Classification quality metrics

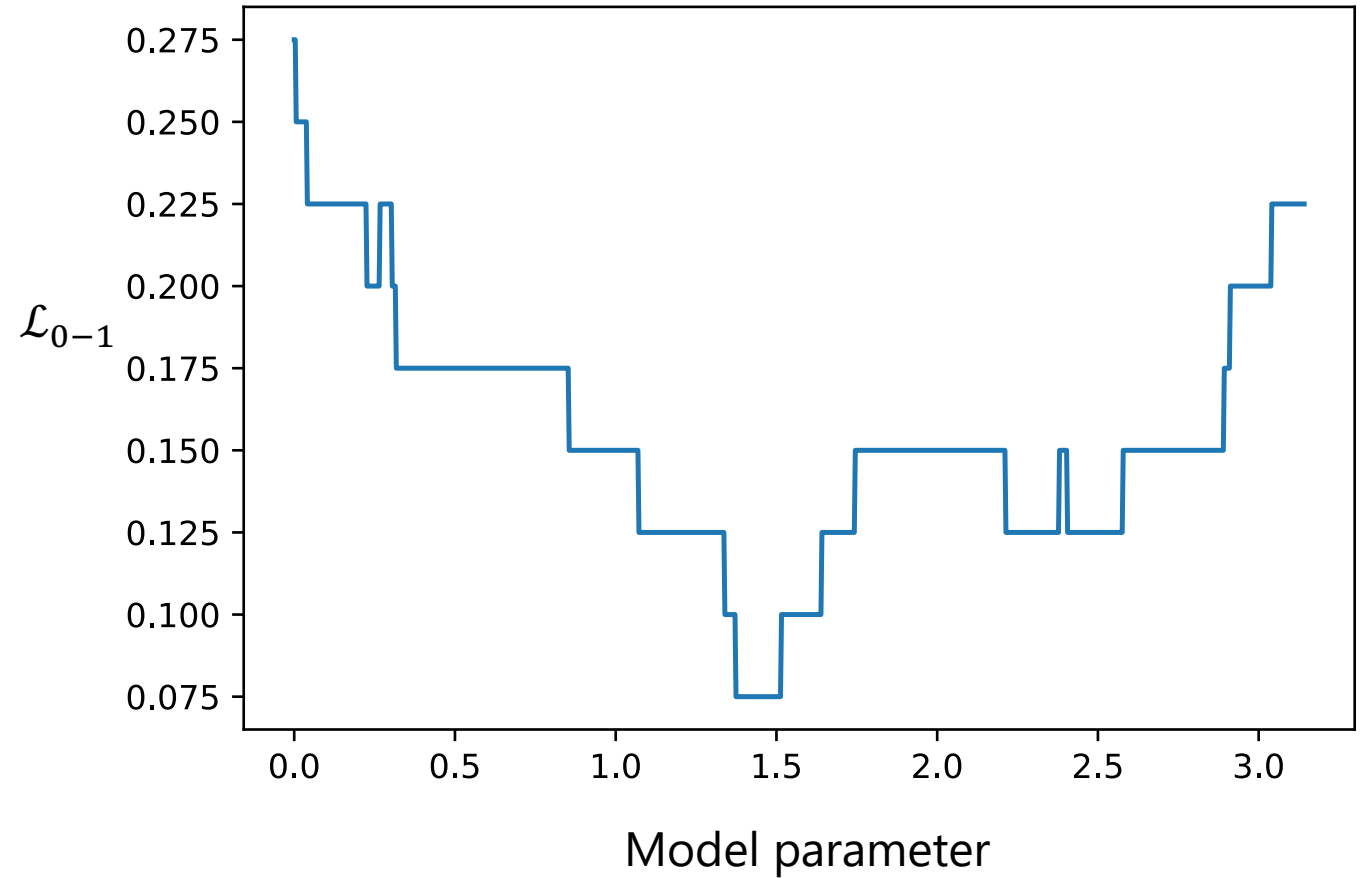


# How to evaluate a classifier?

# 0-1 Loss

Probability of an error (error rate):

$$\mathcal{L}_{0-1} = \frac{1}{N} \sum_{i=1 \dots N} \mathbb{I}(y_i \neq \hat{y}_i)$$



# 0-1 Loss

Probability of an error (error rate):

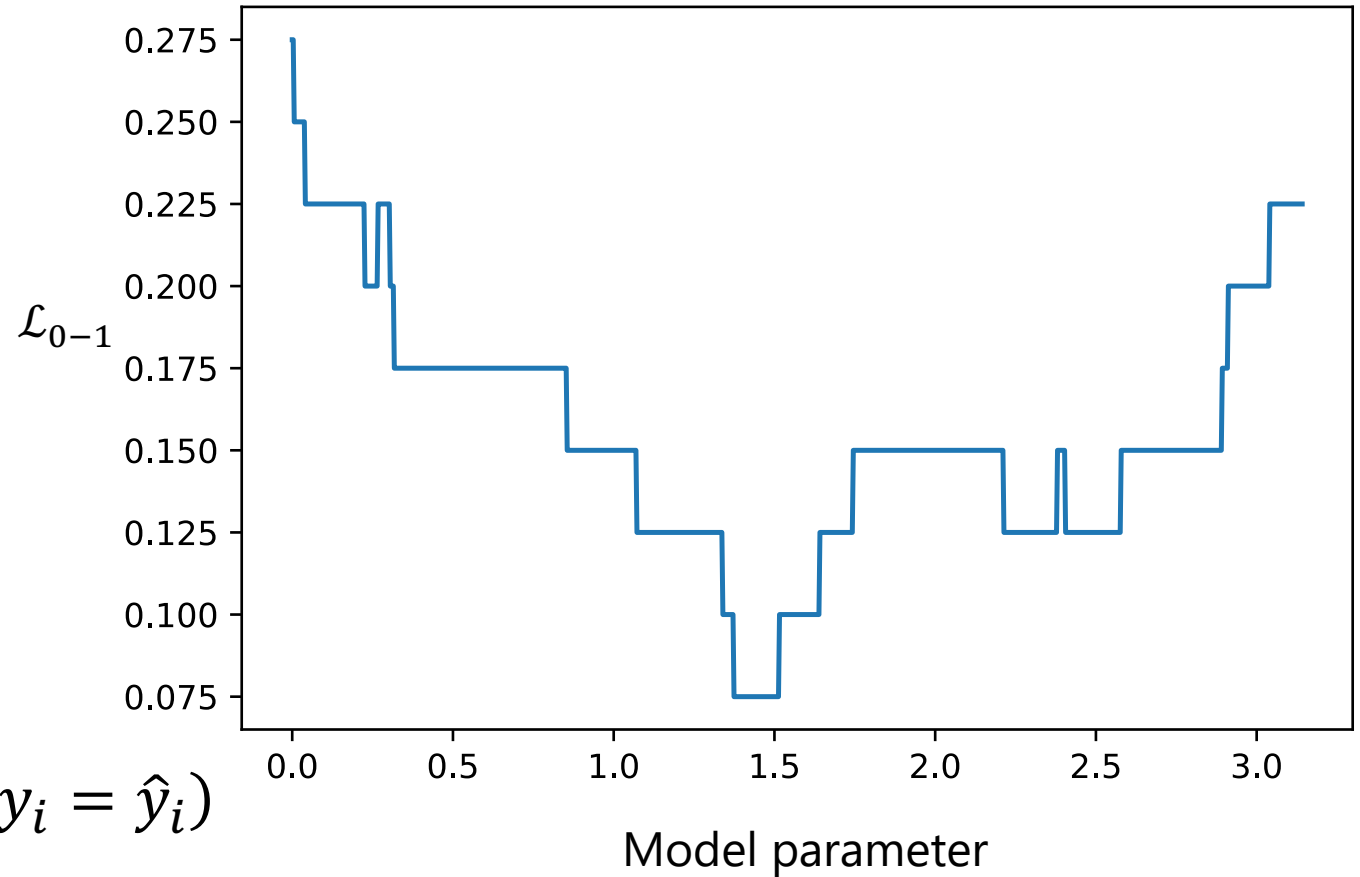
$$\mathcal{L}_{0-1} = \frac{1}{N} \sum_{i=1 \dots N} \mathbb{I}(y_i \neq \hat{y}_i)$$

Accuracy:

$$accuracy = 1 - \mathcal{L}_{0-1} = \frac{1}{N} \sum_{i=1 \dots N} \mathbb{I}(y_i = \hat{y}_i)$$

Not always a good quality measure

- E.g. when classes are imbalanced



# Confusion matrix

		Actual class			
		1	2	...	$C$
Predicted class	1	$n_{11}$	$n_{12}$	$\cdots$	$n_{1C}$
	2	$n_{21}$	$n_{22}$	$\cdots$	$n_{2C}$
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
	$C$	$n_{C1}$	$n_{C2}$	$\cdots$	$n_{CC}$

$n_{ij}$  – number of objects of class  $j$ , that were predicted as class  $i$

Diagonal elements – correct classifications

Off-diagonal elements – incorrect classifications

# Binary case

		Actual class	
		+	−
Predicted class	+	<b>TP</b> (True Positives)	<b>FP</b> (False Positives)
	−	<b>FN</b> (False Negatives)	<b>TN</b> (True Negatives)

# Binary case

		Actual class	
		+	-
Predicted class	+	<b>TP</b> (True Positives)	<b>FP</b> (False Positives)
	-	<b>FN</b> (False Negatives)	<b>TN</b> (True Negatives)

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$



# Binary case

		Actual class	
		+	-
Predicted class	+	<b>TP</b> (True Positives)	<b>FP</b> (False Positives)
	-	<b>FN</b> (False Negatives)	<b>TN</b> (True Negatives)

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{True positive rate, TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

# Binary case

		Actual class	
		+	-
Predicted class	+	<b>TP</b> (True Positives)	<b>FP</b> (False Positives)
	-	<b>FN</b> (False Negatives)	<b>TN</b> (True Negatives)

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{True positive rate, TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{False positive rate, FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

# Binary case

		Actual class	
		+	-
Predicted class	+	<b>TP</b> (True Positives)	<b>FP</b> (False Positives)
	-	<b>FN</b> (False Negatives)	<b>TN</b> (True Negatives)

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{True positive rate, TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{False positive rate, FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

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

# Binary case

		Actual class	
		+	-
Predicted class	+	<b>TP</b> (True Positives)	<b>FP</b> (False Positives)
	-	<b>FN</b> (False Negatives)	<b>TN</b> (True Negatives)

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

$$\text{True positive rate, TPR} = \frac{TP}{TP+FN}$$

$$\text{False positive rate, FPR} = \frac{FP}{FP+TN}$$

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

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

# Binary case

		Actual class	
		+	-
Predicted class	+	<b>TP</b> (True Positives)	<b>FP</b> (False Positives)
	-	<b>FN</b> (False Negatives)	<b>TN</b> (True Negatives)

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{True positive rate, TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{False positive rate, FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

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

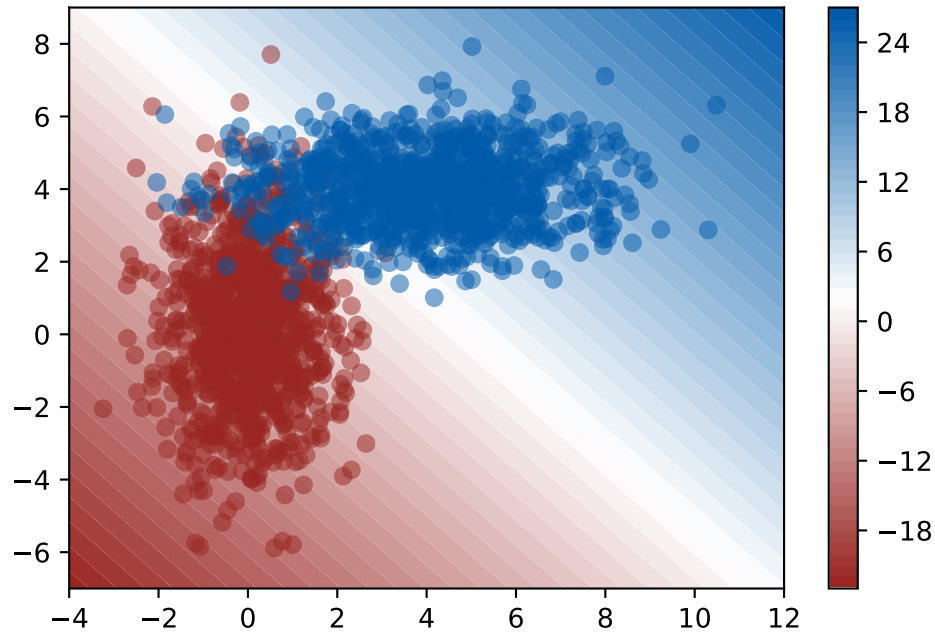
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$F_1\text{-score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

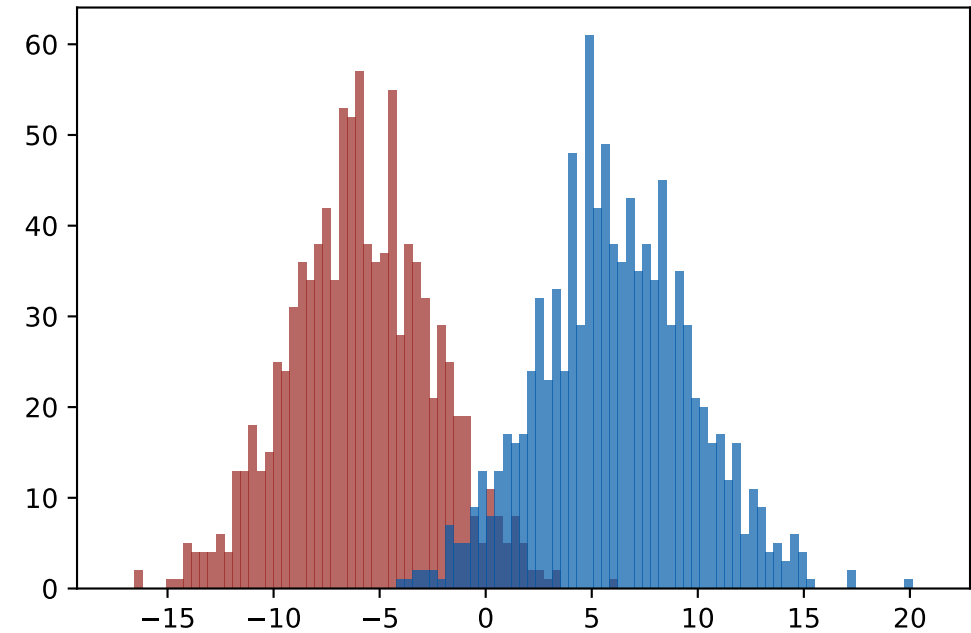
# Continuous predictions

Many classification algorithms work with continuous scoring functions

- E.g. log odds in Logistic Regression, or scoring function of an SVM model



Model prediction

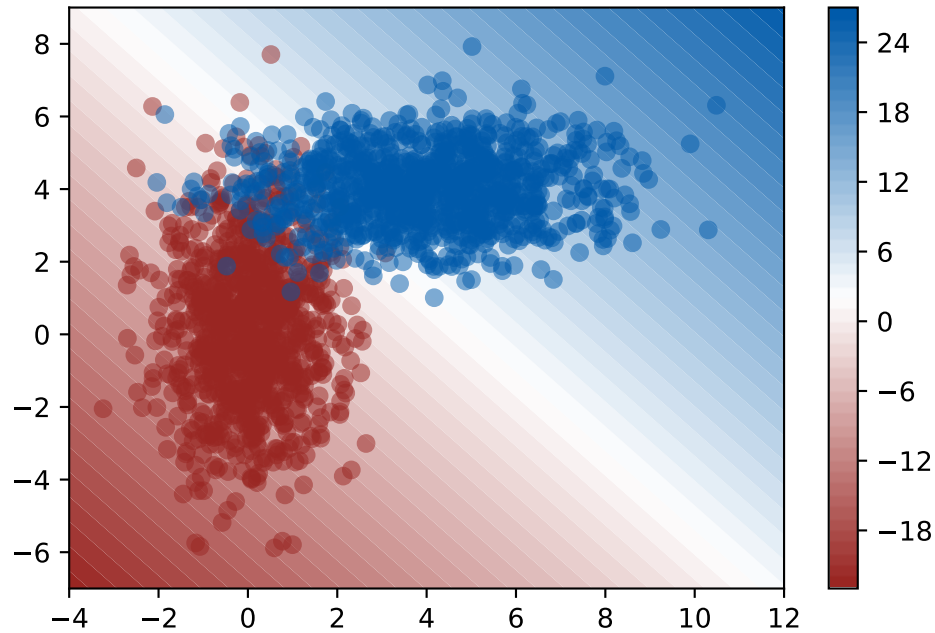


Scoring function values on data

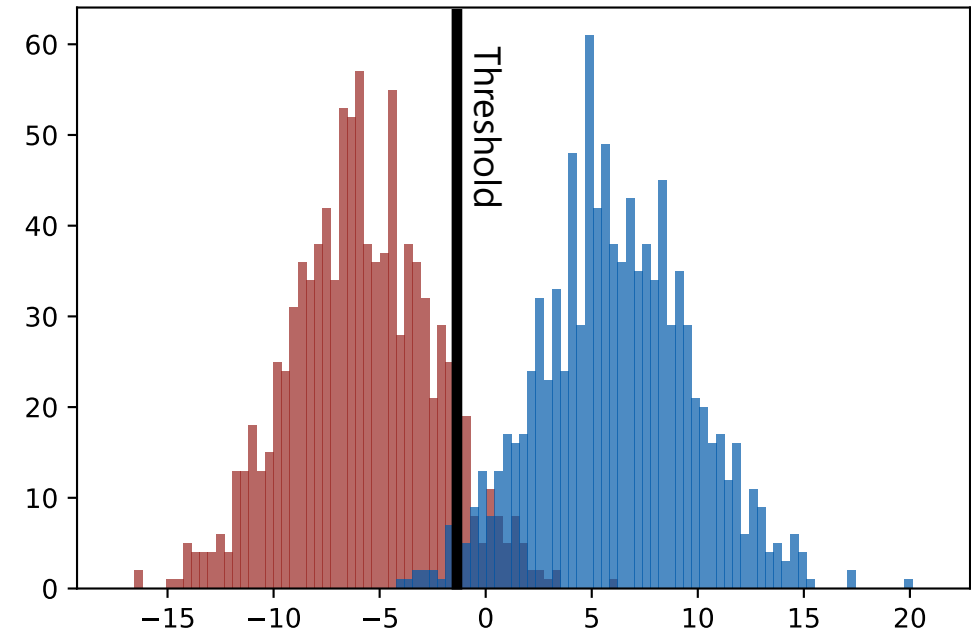
# Continuous predictions

Many classification algorithms work with continuous scoring functions

- E.g. log odds in Logistic Regression, or scoring function of an SVM model



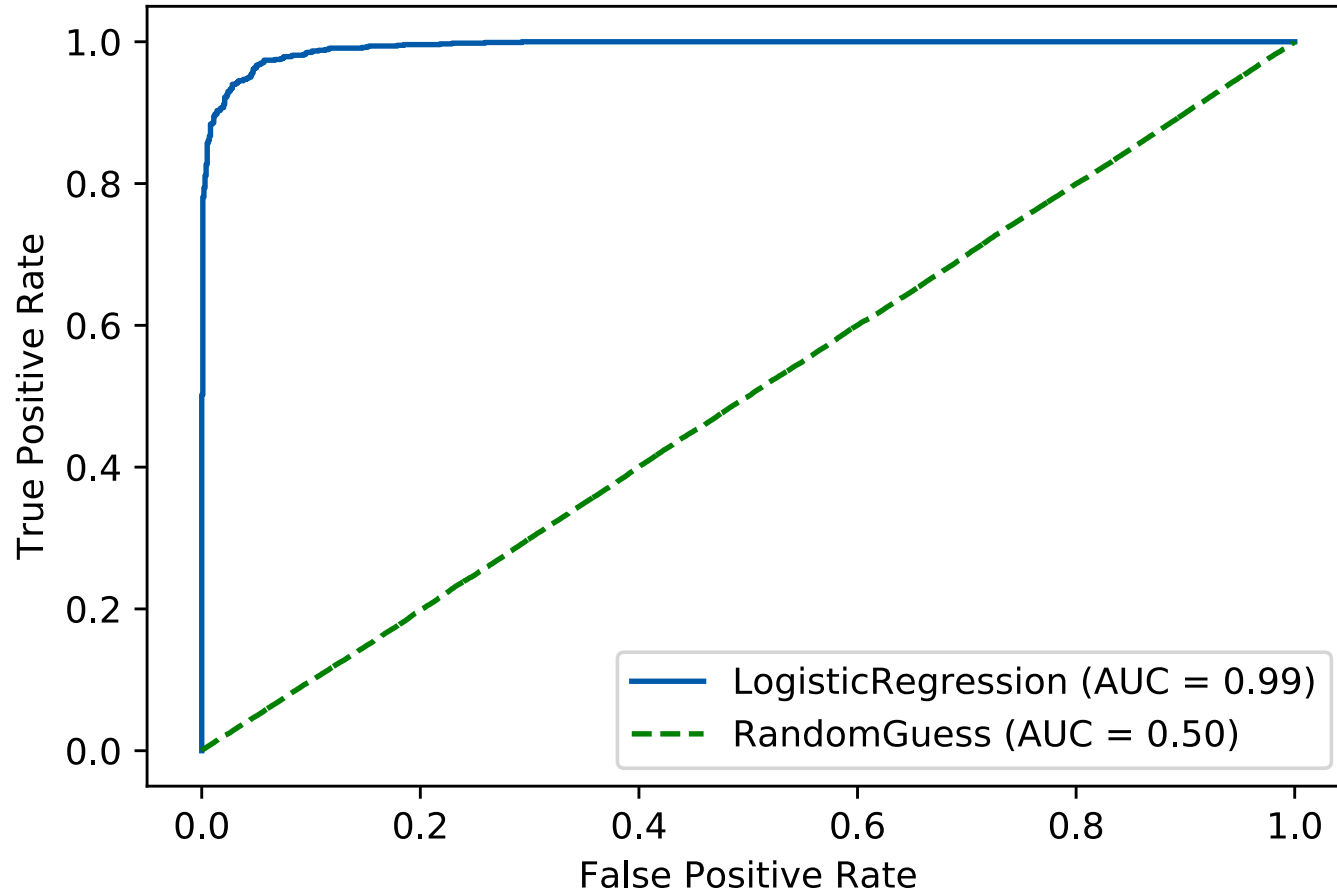
Model prediction



Scoring function values on data

# ROC-curve

Receiver operating characteristic = TPR as a function of FPR



## History [\[ edit \]](#)

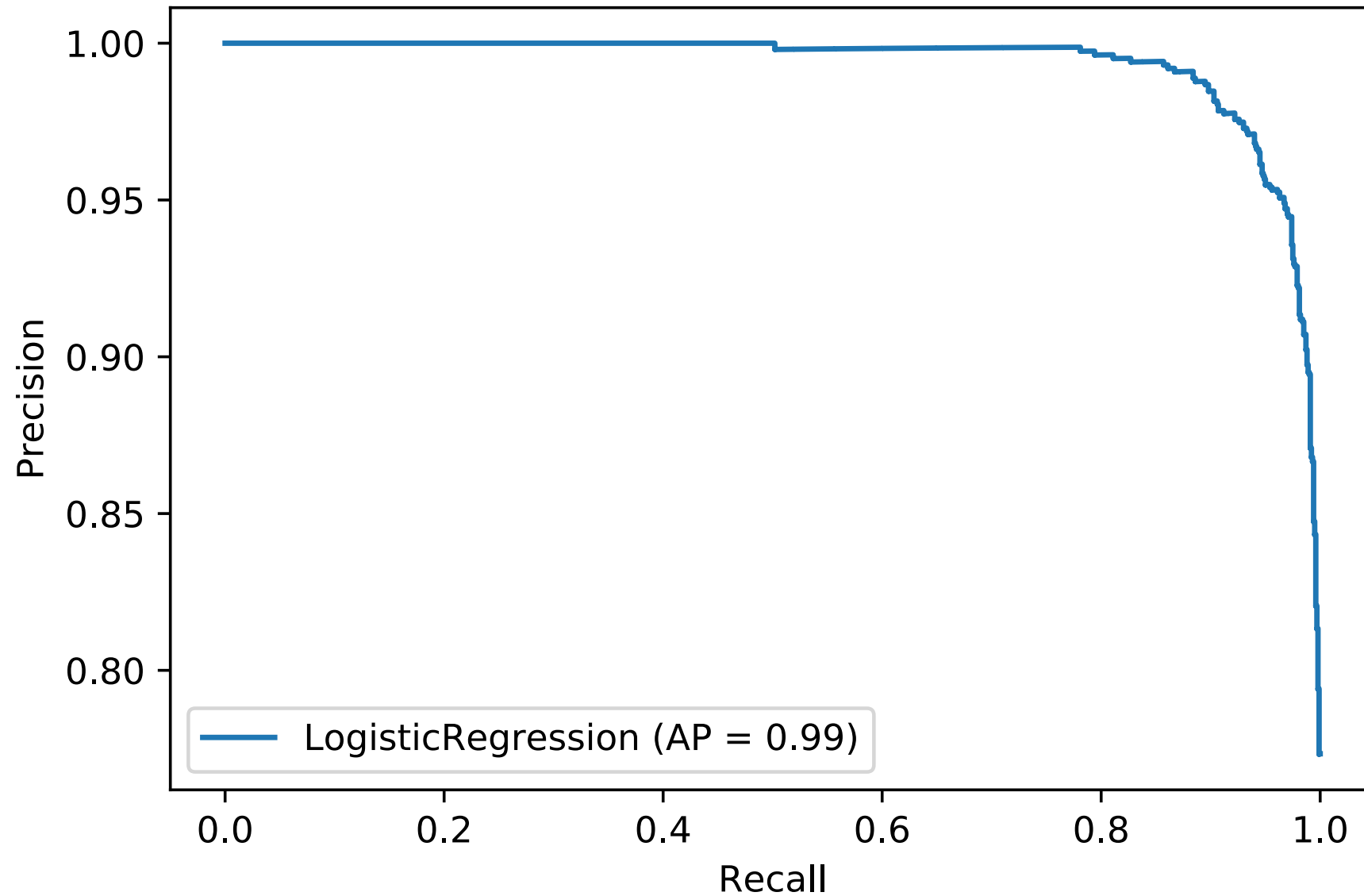
The ROC curve was first used during [World War II](#) for the analysis of [radar signals](#) before it was employed in [signal detection theory](#).<sup>[45]</sup> Following the [attack on Pearl Harbor](#) in 1941, the United States army began new research to increase the prediction of correctly detected Japanese aircraft from their radar signals. For these purposes they measured the ability of a radar receiver operator to make these important distinctions, which was called the Receiver Operating Characteristic.<sup>[46]</sup>

[https://en.wikipedia.org/wiki/Receiver\\_operating\\_characteristic](https://en.wikipedia.org/wiki/Receiver_operating_characteristic)

Nice demo: <http://arogozhnikov.github.io/2015/10/05/roc-curve.html>



# Precision-recall curve



Prediction error  
vs  
expected prediction error



# Two scenarios

# Two scenarios

A:

- Trained a model  $\hat{f}_\tau(x)$  on a particular dataset  $\tau$
- Want to know, how well this particular  $\hat{f}_\tau(x)$  will perform on new data

# Two scenarios

A:

- Trained a model  $\hat{f}_\tau(x)$  on a particular dataset  $\tau$
- Want to know, how well this particular  $\hat{f}_\tau(x)$  will perform on new data

B:

- Chose a particular algorithm  $\mathcal{A}: \tau \rightarrow \hat{f}_\tau$ , for a particular problem – defined by the unknown population distribution  $P(x, y)$
- Want to know, how well this algorithm performs on this problem

# Two scenarios

A:

- Trained a model  $\hat{f}_\tau(x)$  on a particular dataset  $\tau$
- Want to know, how well this particular  $\hat{f}_\tau(x)$  will perform on new data
- **Prediction error:**

$$\text{Err}_\tau = \mathbb{E}_{x,y} \left[ L \left( y, \hat{f}_\tau(x) \right) \right]$$

B:

- Chose a particular algorithm  $\mathcal{A}: \tau \rightarrow \hat{f}_\tau$ , for a particular problem – defined by the unknown population distribution  $P(x, y)$
- Want to know, how well this algorithm performs on this problem

# Two scenarios

A:

- Trained a model  $\hat{f}_\tau(x)$  on a particular dataset  $\tau$
- Want to know, how well this particular  $\hat{f}_\tau(x)$  will perform on new data
- **Prediction error:**

$$\text{Err}_\tau = \mathbb{E}_{x,y} \left[ L \left( y, \hat{f}_\tau(x) \right) \right]$$

B:

- Chose a particular algorithm  $\mathcal{A}: \tau \rightarrow \hat{f}_\tau$ , for a particular problem – defined by the unknown population distribution  $P(x, y)$
- Want to know, how well this algorithm performs on this problem
- **Expected prediction error:**

$$\text{Err} = \mathbb{E}_{x,y,\tau} \left[ L \left( y, \hat{f}_\tau(x) \right) \right] = \mathbb{E}_\tau [\text{Err}_\tau]$$

# Splitting to train and test



What kind of error do we estimate here?



# Splitting to train and test



What kind of error do we estimate here?

How to estimate its variance?

# Splitting to train, validation and test

When we do model selection, we use the left-out data to estimate the prediction error and minimize it (e.g., wrt the hyperparameters)

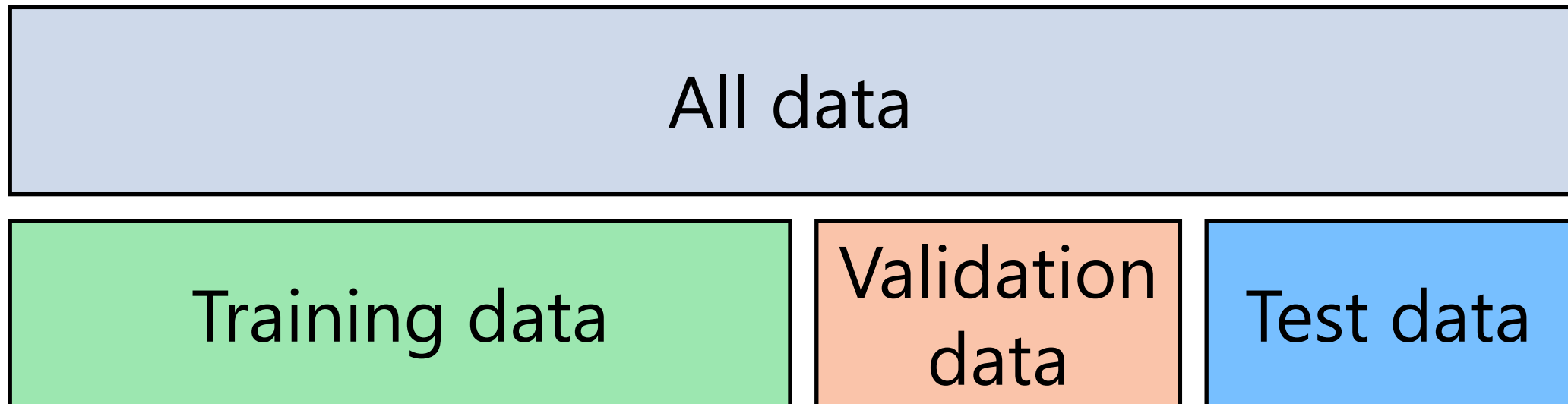
May 'overfit to test', so the resulting minimized error is not a good estimate of the prediction error

# Splitting to train, validation and test

When we do model selection, we use the left-out data to estimate the prediction error and minimize it (e.g., wrt the hyperparameters)

May 'overfit to test', so the resulting minimized error is not a good estimate of the prediction error

Solution: tune on the **validation data**, do the final evaluation on the **test data**

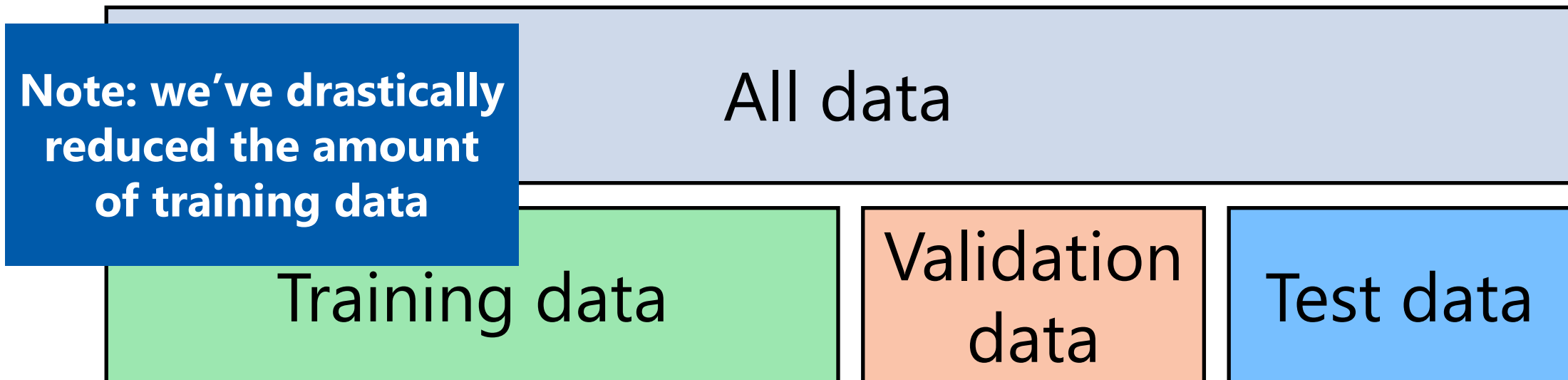


# Splitting to train, validation and test

When we do model selection, we use the left-out data to estimate the prediction error and minimize it (e.g., wrt the hyperparameters)

May 'overfit to test', so the resulting minimized error is not a good estimate of the prediction error

Solution: tune on the **validation data**, do the final evaluation on the **test data**



# Cross-validation



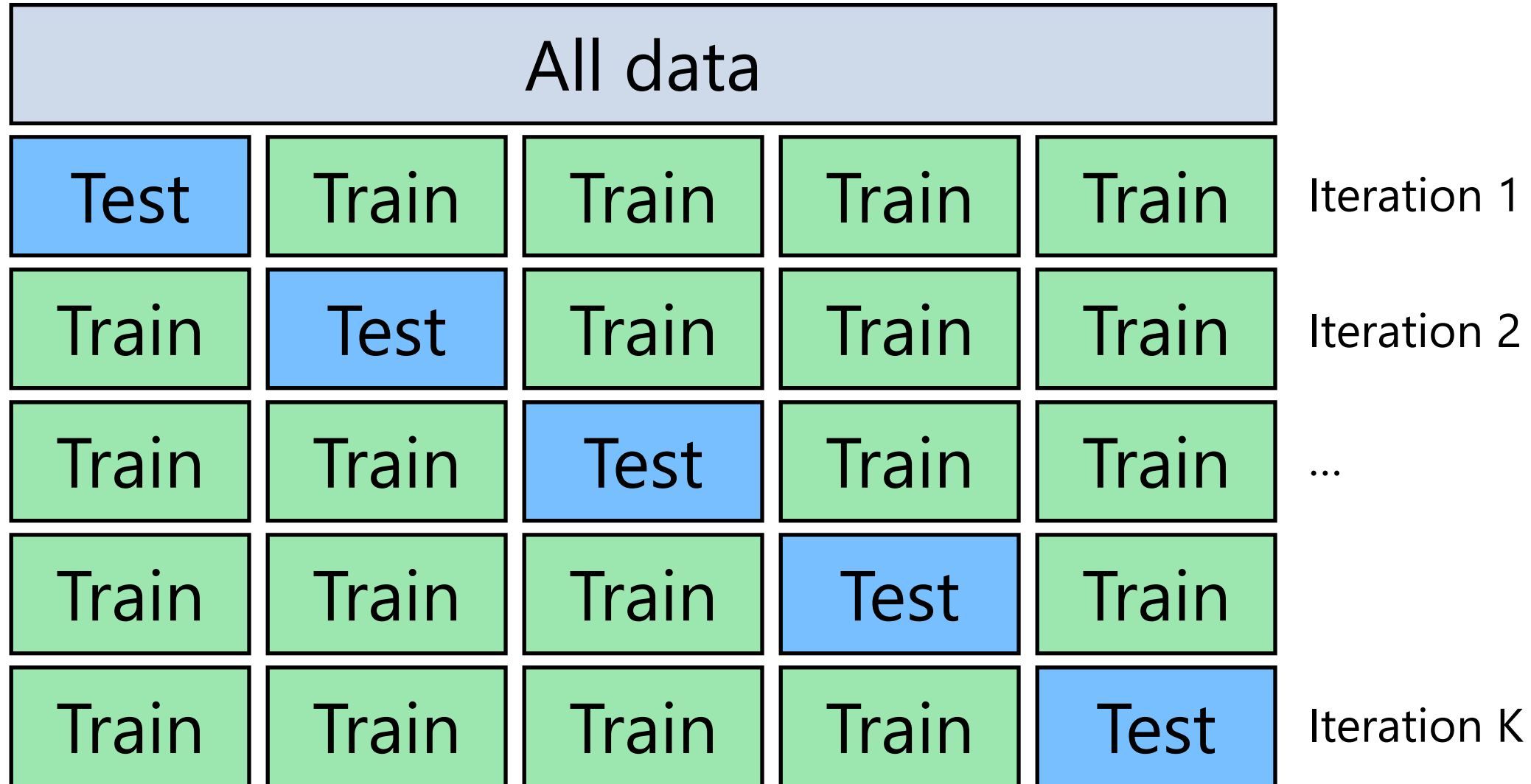
# How to estimate the expected prediction error?

$$\text{Err} = \mathbb{E}_{x,y,\tau} \left[ L \left( y, \hat{f}_{\tau}(x) \right) \right] = \mathbb{E}_{\tau} [\text{Err}_{\tau}]$$

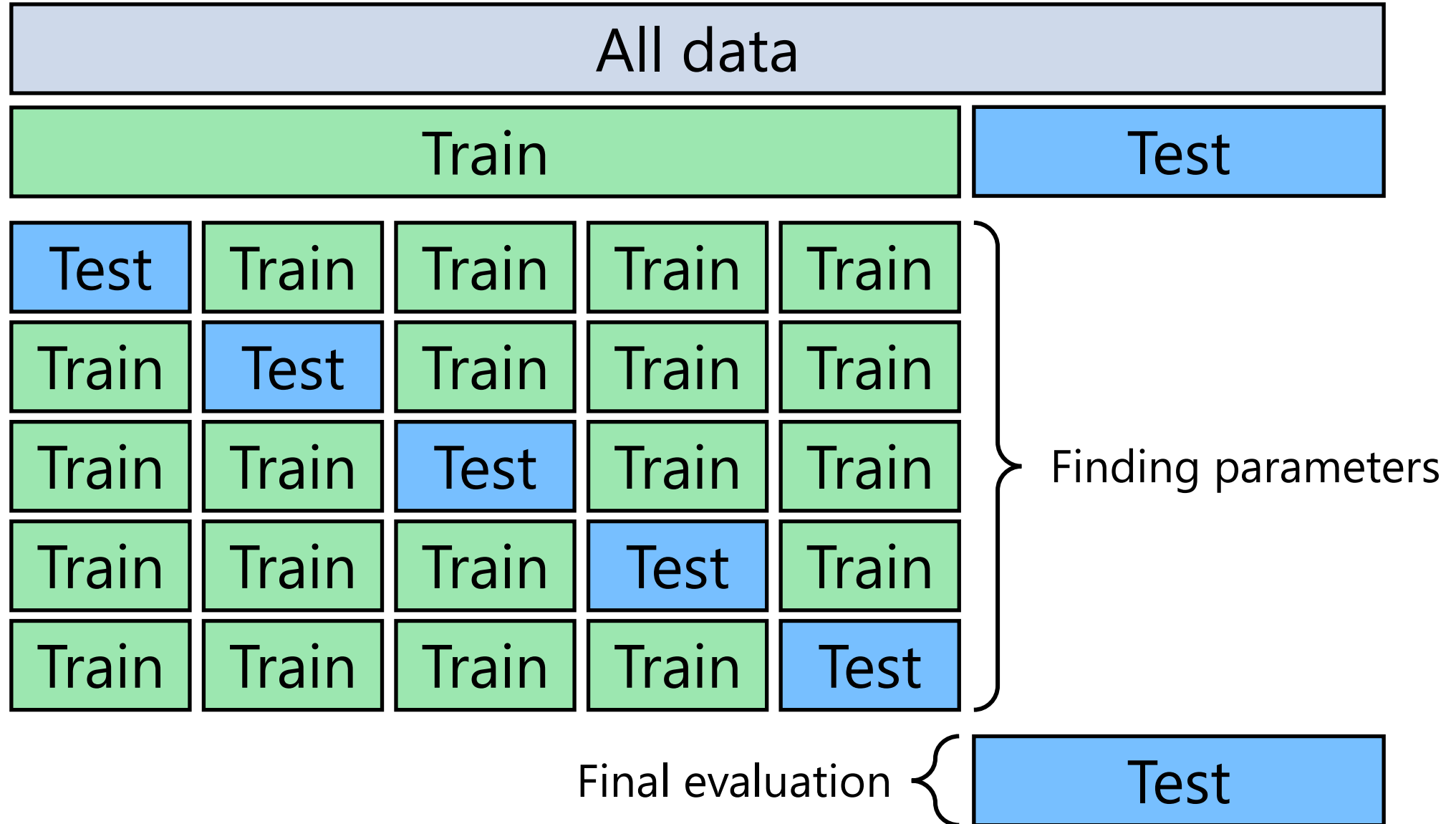
We can't just sample new training sets!  
(unless there's **really** a lot of data available to us)

Note:  $\text{Err}_{\tau}$  is by itself an estimate of  $\text{Err}$ ,  
but since it's just a single observation we  
know nothing about its variance

# K-fold cross-validation



# Hyperparameter tuning





# Thank you!

Majid Sohrabi



[msohrabi@hse.ru](mailto:msohrabi@hse.ru)