COMP2261 ARTIFICIAL INTELLIGENCE / MACHINE LEARNING

Classifier Evaluation

Dr Yang Long

Durham University

Previously

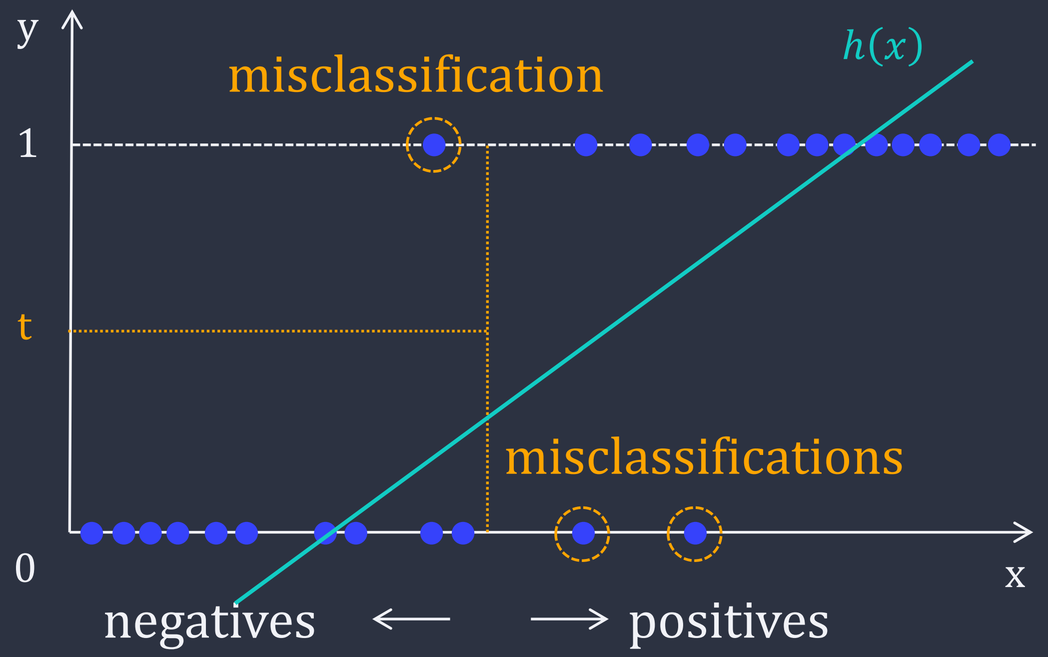
1. Problem Framing
2. Data Preparation
3. Algorithm Selection
4. Model Training
5. Model Testing
6. Hyperparameter Tuning
7. Inference / Prediction

Lecture Overview

1. Accuracy
2. Confusion Matrix
3. Sensitivity (Recall) & Specificity (Selectivity)
4. Positive Predictive Value (Precision) & Negative Predictive Value
5. ROC & AUC

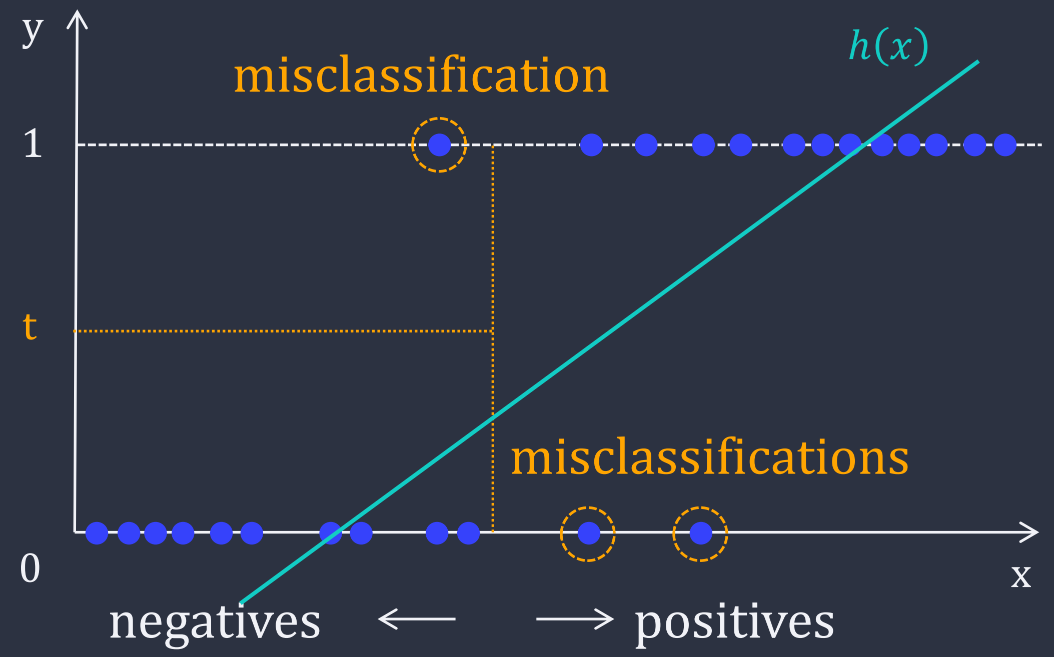
1. Accuracy

Accuracy



$$accuracy = \frac{correct\ predictions}{all\ the\ predictions}$$

Accuracy



$$accuracy = \frac{correct\ predictions}{all\ the\ predictions} = \frac{22}{25} = 88\%$$

Problem of Accuracy

EXAMPLE.

A classifier, which can achieve a very high accuracy of… 99% !

Is it really a good classifier?

Problem of Accuracy

EXAMPLE.

Class A
(positive class)

10 examples

Class B
(negative class)

990 examples

Q: how to train a classifier that can achieve such a high accuracy of 99%?

A: don't train, but just make a fake classifier and make it to predict all as Class B.

Problem of Accuracy

EXAMPLE.

Q: how to train a classifier that can achieve such a high accuracy of 99%?

A: don’t train, but just make a fake classifier and make it to predict all as Class B.

Accuracy could be misleading, especially when dataset is unbalanced.

2. Confusion Matrix

EXAMPLE.

x	y	$h(x)$	$\hat{y}$
	+	h_1	+
	+	h_1	-
	-	h_2	+
…	…	…	…
	-	h_m	-
feature	actual label	result of h(x)	predicted label

EXAMPLE.

x	y	$h(x)$	$\hat{y}$	
	+	h_1	+	correct
	+	h_1	-	
	-	h_2	+	
...	...	...	...	
	-	h_m	-	

EXAMPLE.

x	y	$h(x)$	$\hat{y}$	
	+	h_1	+	
	+	h_1	-	incorrect
	-	h_2	+	
...	...	...	...	
	-	h_m	-	

EXAMPLE.

x	y	$h(x)$	$\hat{y}$	
	+	h_1	+	
	+	h_1	-	
	-	h_2	+	incorrect
...	...	...	...	
	-	h_m	-	

EXAMPLE.

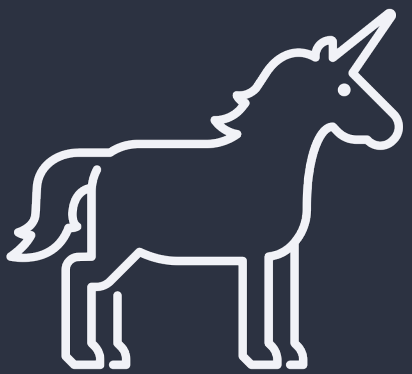
x	y	$h(x)$	$\hat{y}$		
	+	h_1	+	$+ \rightarrow +$	✓
	+	h_1	-	$+ \rightarrow -$	✗
	-	h_2	+	$- \rightarrow +$	✗
...	...	...	...		
	-	h_m	-	$- \rightarrow -$	✓

Confusion Matrix

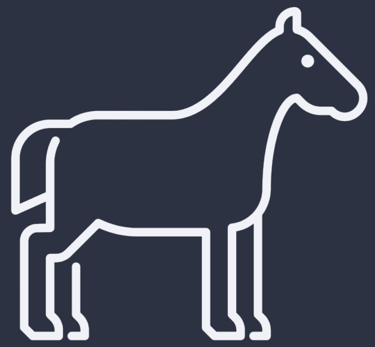
		actual class	
		+	–
predicted class	+	true positive (TP)	false positive (FP)
	–	false negative (FN)	true negative (TN)

Confusion Matrix

EXAMPLE.



Unicorn
(positive class)



Horse
(negative class)

20 photos: 8 unicorns and 12 horses

$$\text{Actual} = [+, +, +, +, +, +, +, +, -, -, -, -, -, -, -, -, -, -, -, -]$$

$$\text{Predicted} = [-, +, -, +, +, +, +, +, -, -, -, -, +, -, -, +, +, -, -, -]$$

Confusion Matrix

EXAMPLE.

$$\text{Actual} = [+, \boxed{+}, +, \boxed{+}, \boxed{+}, \boxed{+}, \boxed{+}, \boxed{+}, -, -, -, -, -, -, -, -, -, -, -, -]$$

$$\text{Predicted} = [-, \boxed{+}, -, \boxed{+}, \boxed{+}, \boxed{+}, \boxed{+}, \boxed{+}, -, -, -, -, +, -, -, +, +, -, -, -]$$

actual class

predicted class

	Unicorn	Horse
Unicorn	$\boxed{6\ (TP)}$	3 (FP)
Horse	2 (FN)	9 (TN)

Confusion Matrix

EXAMPLE.

$$\text{Actual} = [\boxed{+}, +, \boxed{+}, +, +, +, +, +, -, -, -, -, -, -, -, -, -, -, -, -]$$

$$\text{Predicted} = [\boxed{-}, +, \boxed{-}, +, +, +, +, +, -, -, -, -, +, -, -, +, +, -, -, -]$$

		actual class	
		Unicorn	Horse
predicted class	Unicorn	6 (TP)	3 (FP)
	Horse	2 (FN)	9 (TN)

Confusion Matrix

EXAMPLE.

$$\text{Actual} = [+, +, +, +, +, +, +, +, \boxed{-}, \boxed{-}, \boxed{-}, \boxed{-}, -, \boxed{-}, \boxed{-}, -, -, \boxed{-}, \boxed{-}, \boxed{-}]$$

$$\text{Predicted} = [-, +, -, +, +, +, +, +, \boxed{-}, \boxed{-}, \boxed{-}, \boxed{-}, +, \boxed{-}, \boxed{-}, +, +, \boxed{-}, \boxed{-}, \boxed{-}]$$

actual class

predicted class

	Unicorn	Horse
Unicorn	6 (TP)	3 (FP)
Horse	2 (FN)	$\boxed{9\ (TN)}$

Confusion Matrix

EXAMPLE.

$$\text{Actual} = [+, +, +, +, +, +, +, +, -, -, -, -, \boxed{-}, -, -, \boxed{-}, \boxed{-}, -, -, -]$$

$$\text{Predicted} = [-, +, -, +, +, +, +, +, -, -, -, -, \boxed{+}, -, -, \boxed{+}, \boxed{+}, -, -, -]$$

actual class

predicted class	Unicorn	Horse
Unicorn	6 (TP)	3 (FP)
Horse	2 (FN)	9 (TN)

Confusion Matrix

EXAMPLE.

$$\text{Actual} = [+, +, +, +, +, +, +, +, -, -, -, -, -, -, -, -, -, -, -, -]$$

$$\text{Predicted} = [-, +, -, +, +, +, +, +, -, -, -, -, +, -, -, +, +, -, -, -]$$

actual class

predicted class		Unicorn	Horse
	Unicorn	6 (TP)	3 (FP)
	Horse	2 (FN)	9 (TN)

correct predictions
(6+9=15)

Confusion Matrix

EXAMPLE.

$$\text{Actual} = [+, +, +, +, +, +, +, +, -, -, -, -, -, -, -, -, -, -, -, -]$$

$$\text{Predicted} = [-, +, -, +, +, +, +, +, -, -, -, -, +, -, -, +, +, -, -, -]$$

actual class

predicted class	Unicorn	Horse
Unicorn	6 (TP)	3 (FP)
Horse	2 (FN)	9 (TN)

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$= \frac{15}{20} = 75\%$$

3. Sensitivity and Specificity

EXAMPLE. COVID-19 Testing

Sensitivity (Recall)

whether the test is sensitive enough to detect when someone actually has the virus.

low sensitivity -> a lot of false negatives (the test is not sensitive enough to pick up the virus)

Specificity (Selectivity)

do we know we have this specific virus.

low specificity -> a lot of false positives where the test tells we have the virus but we actually don't.

EXAMPLE. COVID-19 Testing

	Has virus	Doesn't have virus
Tested positive	true positive (TP)	false positive (FP)
Tested negative	false negative (FN)	true negative (TN)

$$\uparrow Sensitivity = \frac{TP}{TP + \boxed{FN}} \downarrow$$

(recall / true positive rate (TPR))

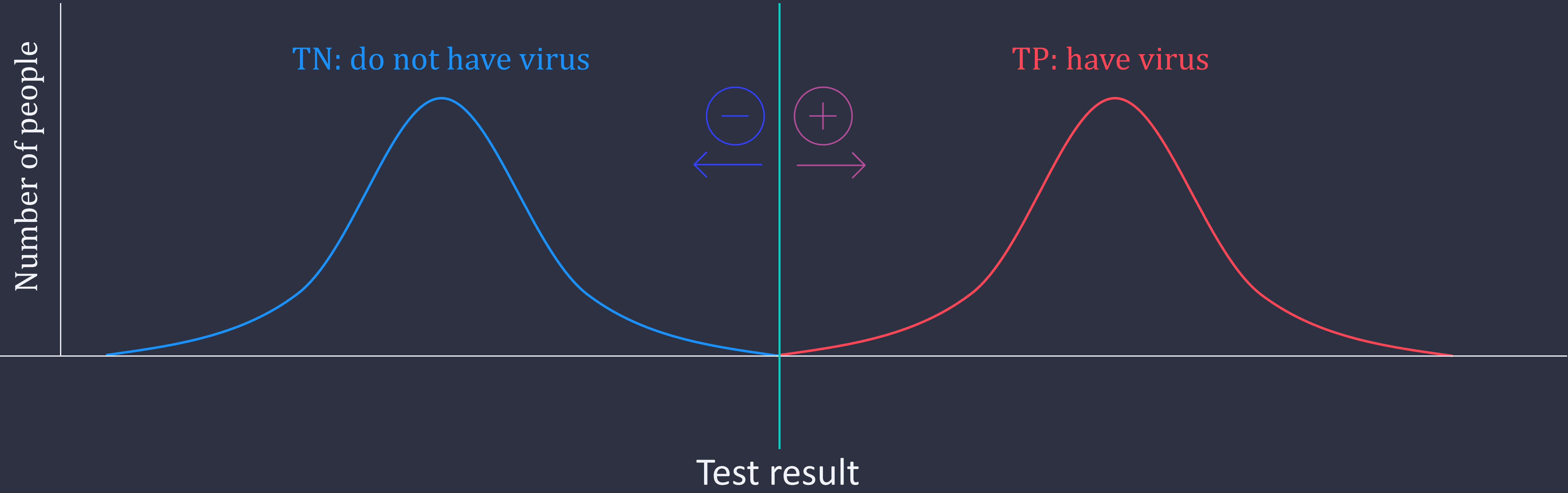
EXAMPLE. COVID-19 Testing

	Has virus	Doesn't have virus
Tested positive	true positive (TP)	false positive (FP)
Tested negative	false negative (FN)	true negative (TN)

$$Sensitivity = \frac{TP}{TP + FN}$$ (recall / true positive rate (TPR))

$$\uparrow Specificity = \frac{TN}{TN + \boxed{FP}} \downarrow$$ (selectivity / true negative rate (TNR))

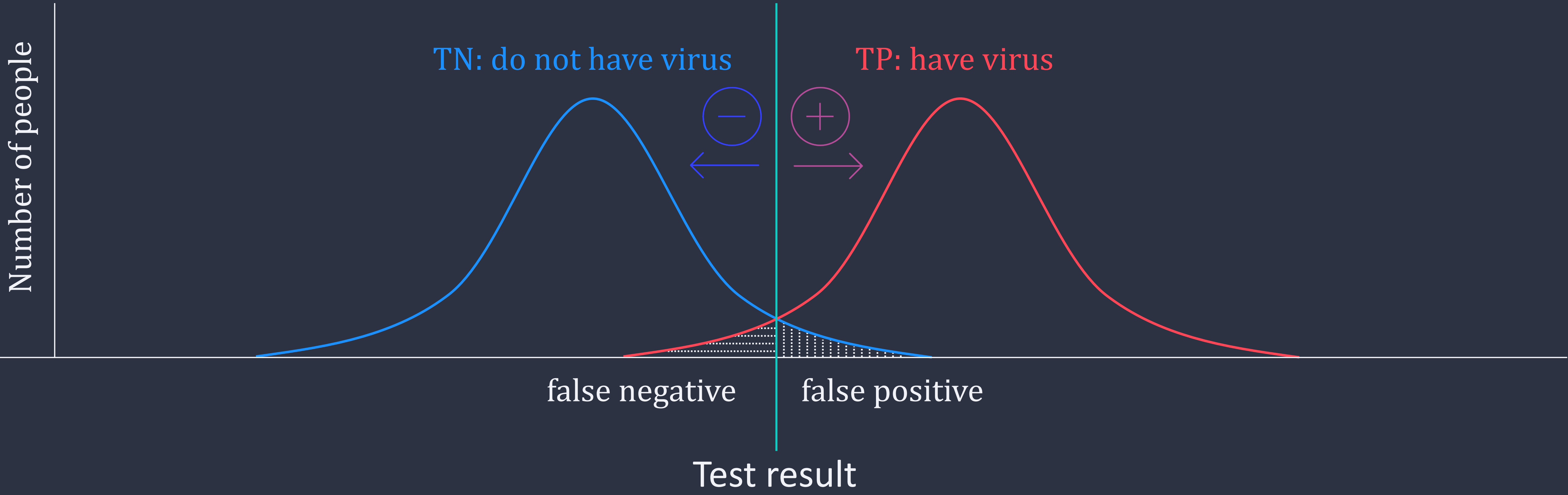
EXAMPLE. **COVID-19 Testing**



$$Specificity = \frac{TN}{TN + FP} = \frac{TN}{TN + 0} = 100\%$$

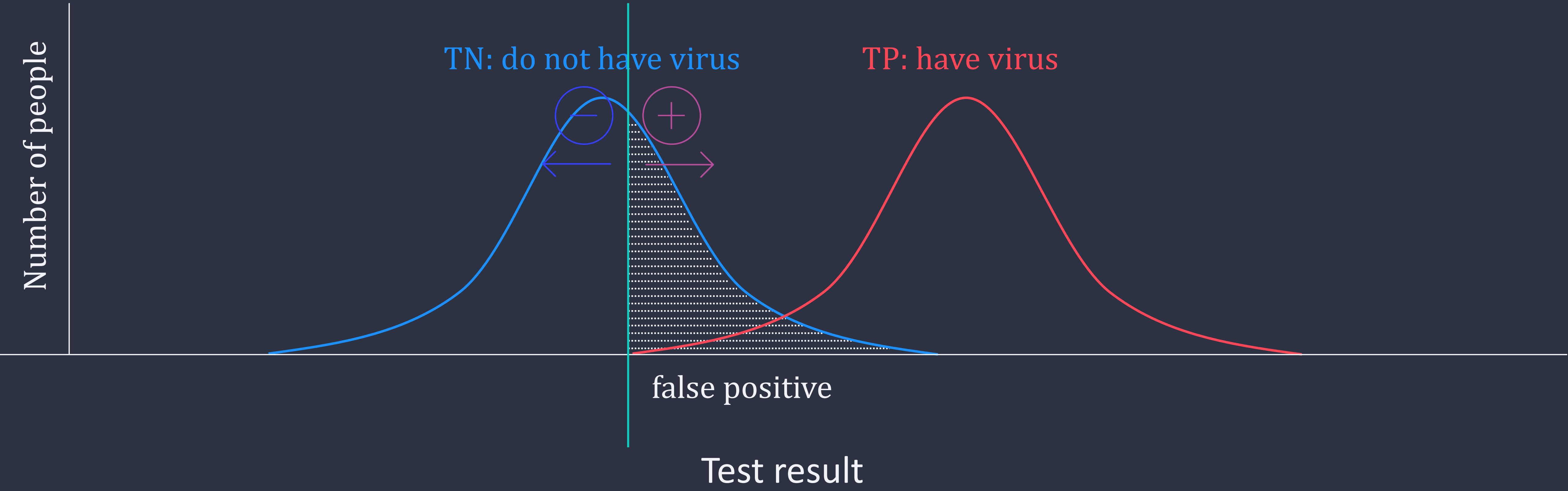
$$Sensitivity = \frac{TP}{TP + FN} = \frac{TP}{TP + 0} = 100\%$$

EXAMPLE. COVID-19 Testing



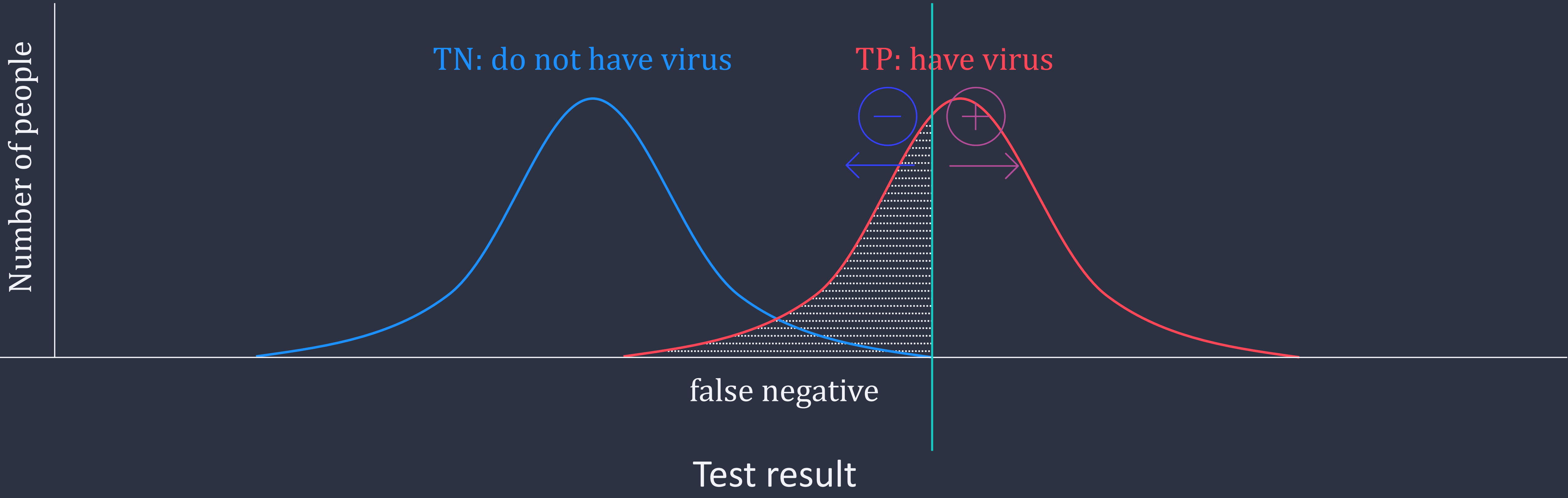
$$Specificity = \frac{TN}{TN + FP} < 100\% \qquad Sensitivity = \frac{TP}{TP + FN} < 100\%$$

EXAMPLE. COVID-19 Testing



$$Specificity = \frac{TN}{TN + FP} \ll 100\% \downarrow \quad Sensitivity = \frac{TP}{TP + FN} = 100\%$$

EXAMPLE. COVID-19 Testing



$$Specificity = \frac{TN}{TN + FP} = 100\% \qquad Sensitivity = \frac{TP}{TP + FN} \ll 100\% \downarrow$$

4. Positive Predictive Value (PPV)
and
Negative Predictive Value (NPV)

Positive Predictive Value (PPV) Negative Predictive Value (NPV)

actual class

predicted class

	+	–
+	TP	FP
–	FN	TN

Positive Predictive Value (PPV)

$$PPV = \frac{TP}{\boxed{TP + FP}}$$

Negative Predictive Value (NPV)

$$NPV = \frac{TN}{\boxed{TN + FN}}$$

		actual class	
		+	−
predicted class	+	TP	FP
	−	FN	TN

predicted class \ actual class	+	−
+	TP	FP
−	FN	TN

$$PPV = \frac{TP}{TP + FP}$$ (precision)

$$NPV = \frac{TN}{TN + FN}$$

$$Sensitivity = \frac{TP}{TP + FN}$$ (recall)

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

actual class

predicted class	+	−
+	TP	FP
−	FN	TN

$$PPV = \frac{TP}{TP + FP} \qquad \text{(precision)}$$

$$NPV = \frac{TN}{TN + FN}$$

$$Sensitivity = \frac{TP}{TP + FN} \qquad \text{(recall)}$$

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

F-score / F-measure

$$F_1 = \frac{2}{precision^{-1} + recall^{-1}} = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

(harmonic mean of *precision* & *recall*)

F-score / F-measure

$$F_1 = \frac{2}{precision^{-1} + recall^{-1}} = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{2TP}{2TP + FP + FN}$$

(ignores *TN* → misleading for unbalanced classes)

(gives equal importance to precision & recall → misleading in practice)

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

Commonly used β

$$\beta = 0.5 \quad F_{0.5} = 1.25 \cdot \frac{precision \cdot recall}{0.25 \cdot precision + recall}$$

$$\beta = 2 \quad F_2 = 5 \cdot \frac{precision \cdot recall}{4 \cdot precision + recall}$$

When $\beta = 1,\ F_\beta$ becomes F_1.

actual class

predicted class

	+	−
+	TP	FP
−	FN	TN

Performance Measures

actual class

predicted class	+	−	measures	
+	TP	FP	Positive predictive value (PPV) (precision)	$\frac{TP}{TP+FP}$
−	FN	TN	Negative predictive value (NPV)	$\frac{TN}{TN+FN}$
measures	Sensitivity (recall) $\frac{TP}{TP+FN}$	Specificity (selectivity) $\frac{TN}{TN+FP}$	Accuracy (ACC) $\frac{TP+TN}{TP+TN+FP+FN}$	F 1 score $2 \cdot \frac{precision \cdot recall}{precision + recall}$

5. ROC & AUC

Binary Classification

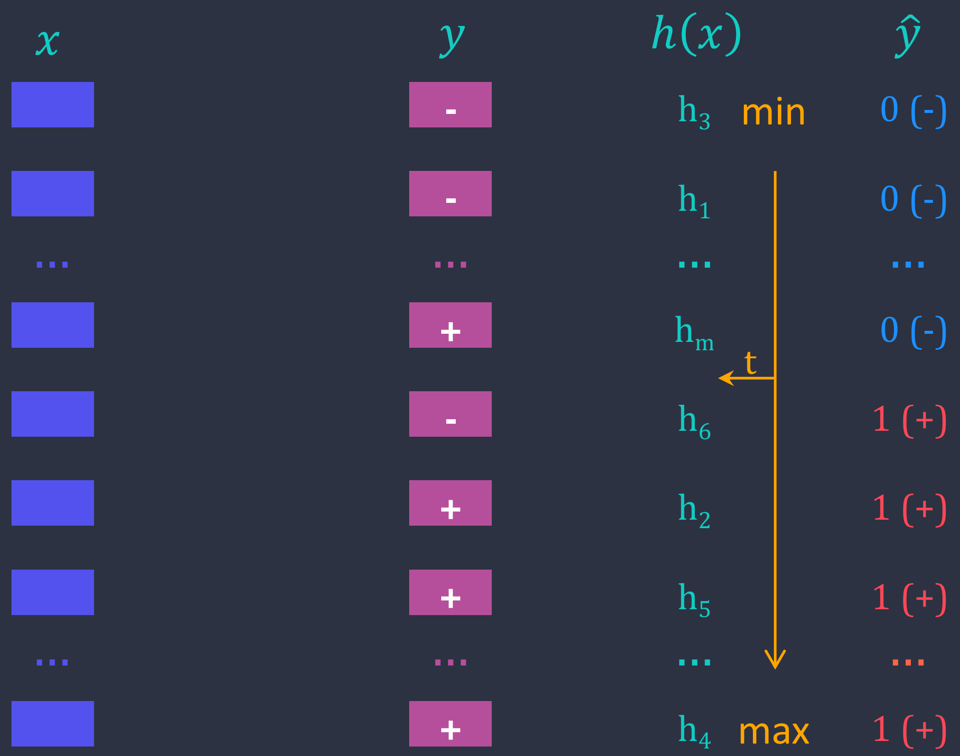
- Learning for an approximation function $h(X)$ mapping input X to output $\hat{y}$
- Discrete output $\hat{y}$ with only two possible values + and –
- Decision rule: $\hat{y} = \begin{cases} +, & h(X) \geq t \\ -, & otherwise \end{cases}$

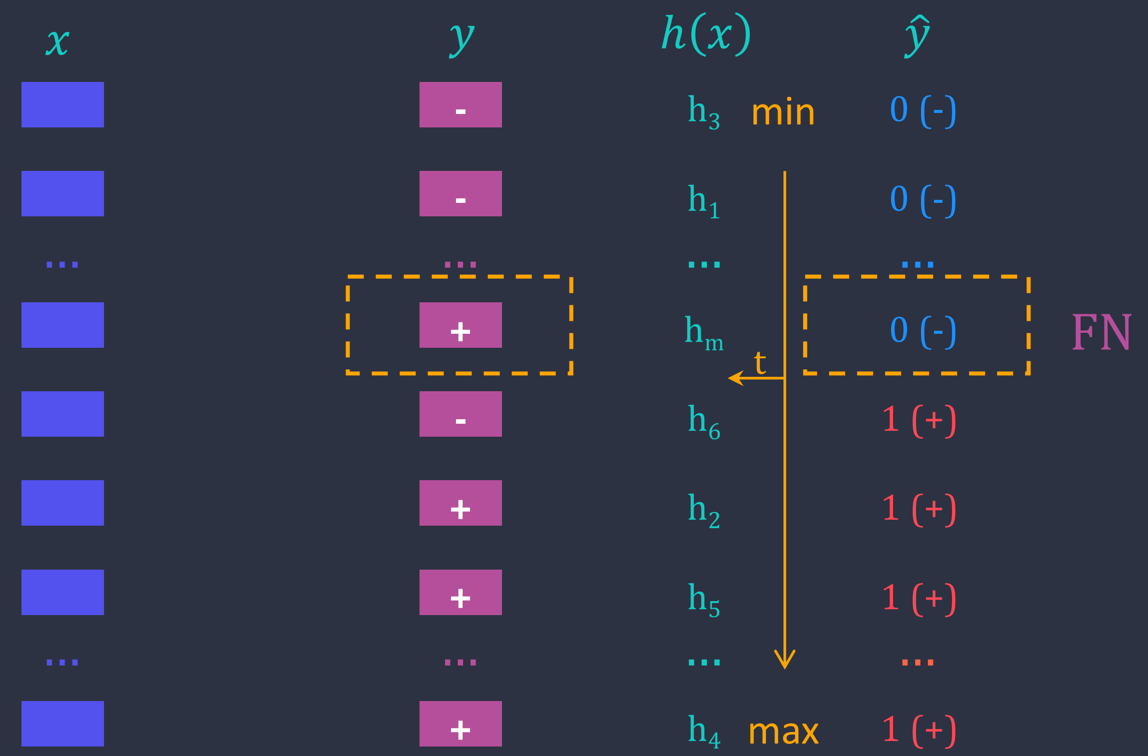
x	y	$h(x)$	$\hat{y}$
	-	h_1	0 (-)
	+	h_2	1 (+)
	-	h_3	0 (-)
	+	h_4	1 (+)
	+	h_5	1 (+)
	-	h_6	1 (+)
...	...	...	...
	+	h_m	0 (-)

Binary Classification

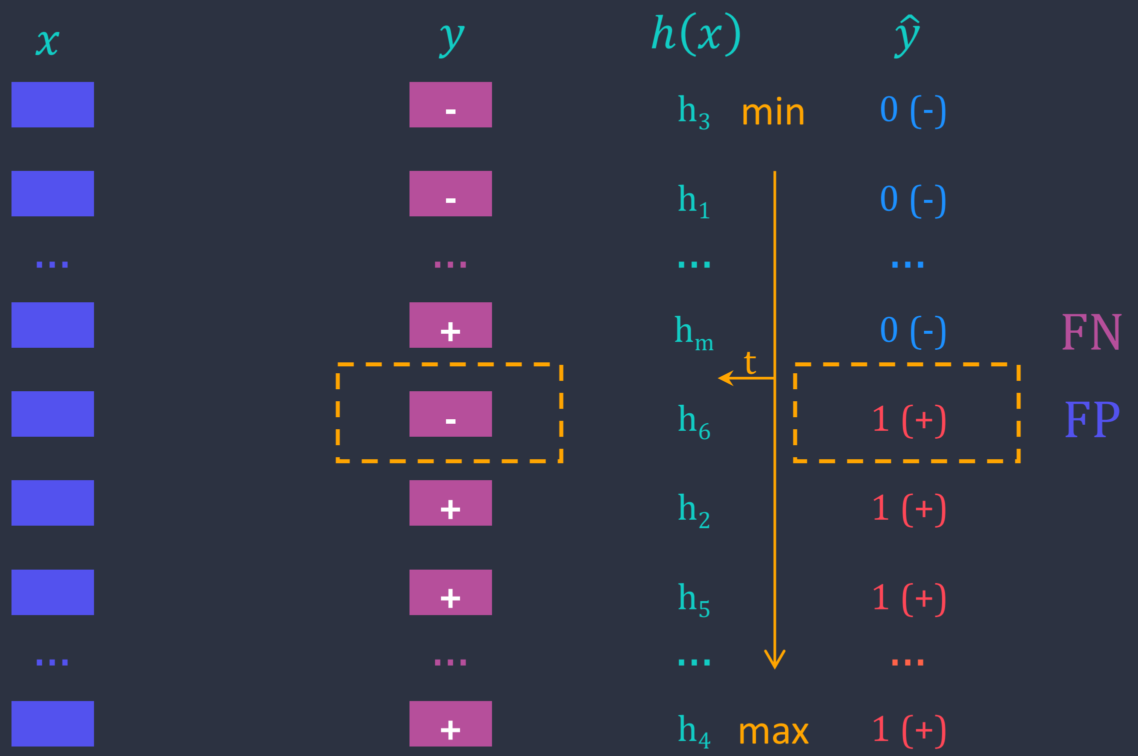
- Learning for an approximation function $h(X)$ mapping input X to output $\hat{y}$
- Discrete output $\hat{y}$ with only two possible values + and –
- Decision rule: $\hat{y} = \begin{cases} +, & h(X) \geq t \\ -, & otherwise \end{cases}$

x	y	$h(x)$	$\hat{y}$
	-	h_1	0 (-)
	+	h_2	1 (+)
	-	h_3	0 (-)
	+	h_4	1 (+)
	+	h_5	1 (+)
	-	h_6	1 (+)
…	…	…	…
	+	h_m	0 (-)


x
y
$h(x)$
$\hat{y}$
-
-
...
+
-
+
+
...
+
h_3 min
h_1
...
h_m
t
h_6
h_2
h_5
...
h_4 max
0 (-)
0 (-)
...
0 (-)
1 (+)
1 (+)
1 (+)
...
1 (+)



x
y
$h(x)$
$\hat{y}$
-
-
+
-
+
+
+
h_3 min
h_1
h_m
t
h_6
h_2
h_5
h_4 max
0 (-)
0 (-)
0 (-)
1 (+)
1 (+)
1 (+)
1 (+)
FN



x
y
$h(x)$
$\hat{y}$
-
-
+
-
+
+
+
h_3 min
h_1
h_m
t
h_6
h_2
h_5
h_4 max
0 (-)
0 (-)
0 (-)
1 (+)
1 (+)
1 (+)
1 (+)
FN
FP

x	y	$h(x)$		$\hat{y}$	
	-	h_3	min	0 (-)	
	-	h_1		0 (-)	
...	...	...		...	
	+	h_m		0 (-)	$\downarrow$FN
			t		
	-	h_6		1 (+)	$\downarrow$FP
	+	h_2		1 (+)	
	+	h_5		1 (+)	
	...	...		...	
	+	h_4	max	1 (+)	

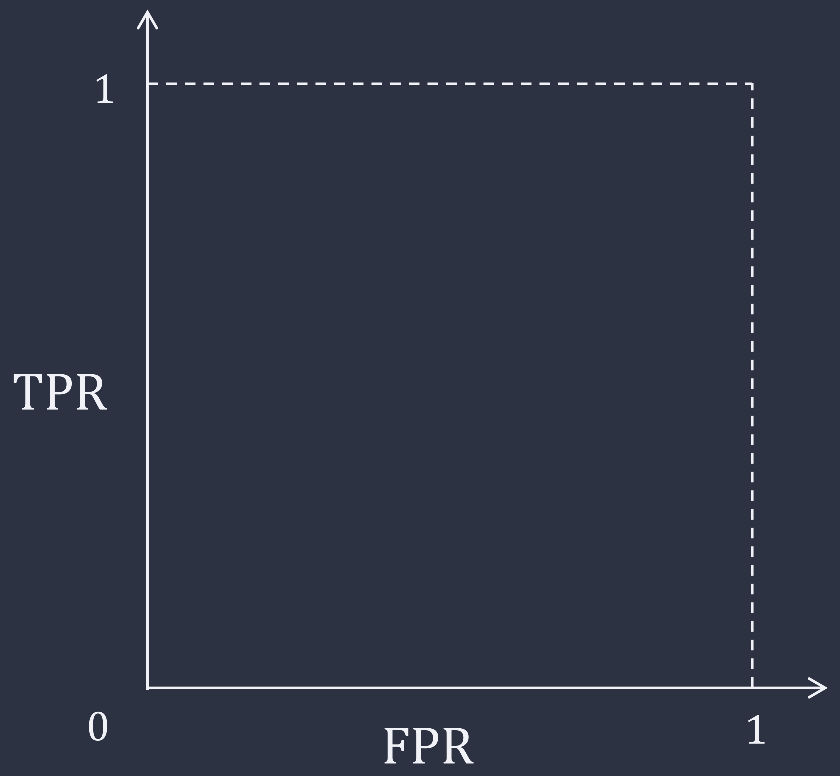
$$\uparrow TPR = \frac{TP}{TP + \downarrow FN}$$

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

x	y	$h(x)$		$\hat{y}$	
	-	h_3	min	0 (-)	
	-	h_1		0 (-)	
...	...	...		...	
	+	h_m		0 (-)	FN
			t		
	-	h_6		1 (+)	FP
	+	h_2		1 (+)	
	+	h_5		1 (+)	
...	...	...		...	
	+	h_4	max	1 (+)	

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

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



x	y	$h(x)$	t	$\hat{y}$	
	-	h_3	min	1 (+)	FP
	-	h_1		1 (+)	FP
...	...	...		...	
	+	h_m		1 (+)	
	-	h_6		1 (+)	FP
	+	h_2		1 (+)	
	+	h_5		1 (+)	
...	...	...		...	
	+	h_4	max	1 (+)	

$$TPR = \frac{TP}{TP + \cancel{FN}} = 1$$

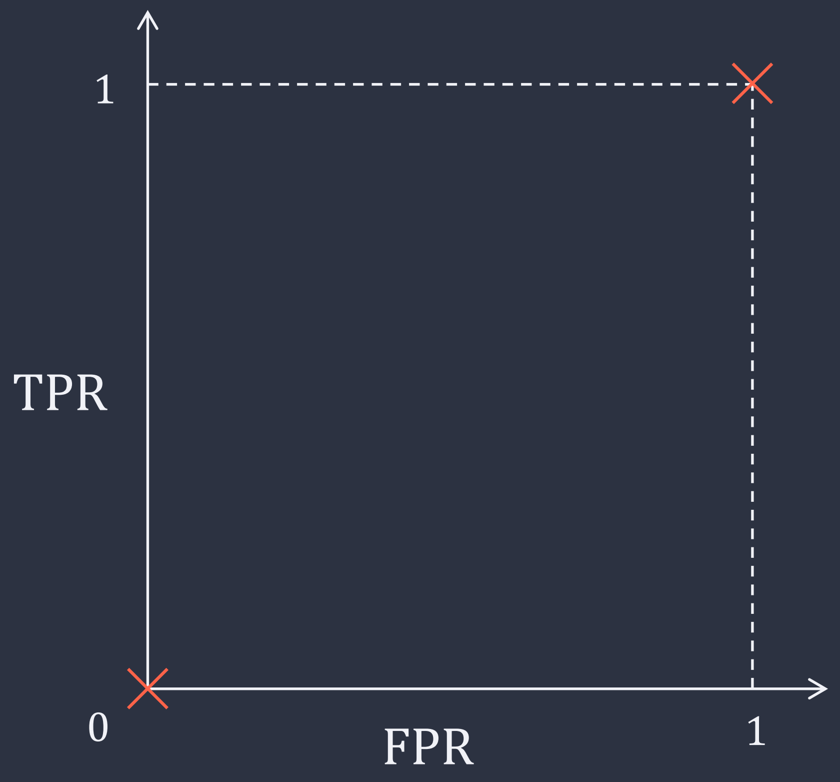
$$FPR = \frac{FP}{FP + \cancel{TN}} = 1$$

x	y	$h(x)$		$\hat{y}$	
	-	h_3	min	0 (-)	
	-	h_1		0 (-)	
...	...	...		...	
	+	h_m		0 (-)	FN
	-	h_6		0 (-)	
	+	h_2		0 (-)	FN
	+	h_5		0 (-)	FN
...	...	...		...	
	+	h_4	max	0 (-)	FN

t

$$TPR = \frac{\cancel{TP}}{\cancel{TP} + FN} = 0$$

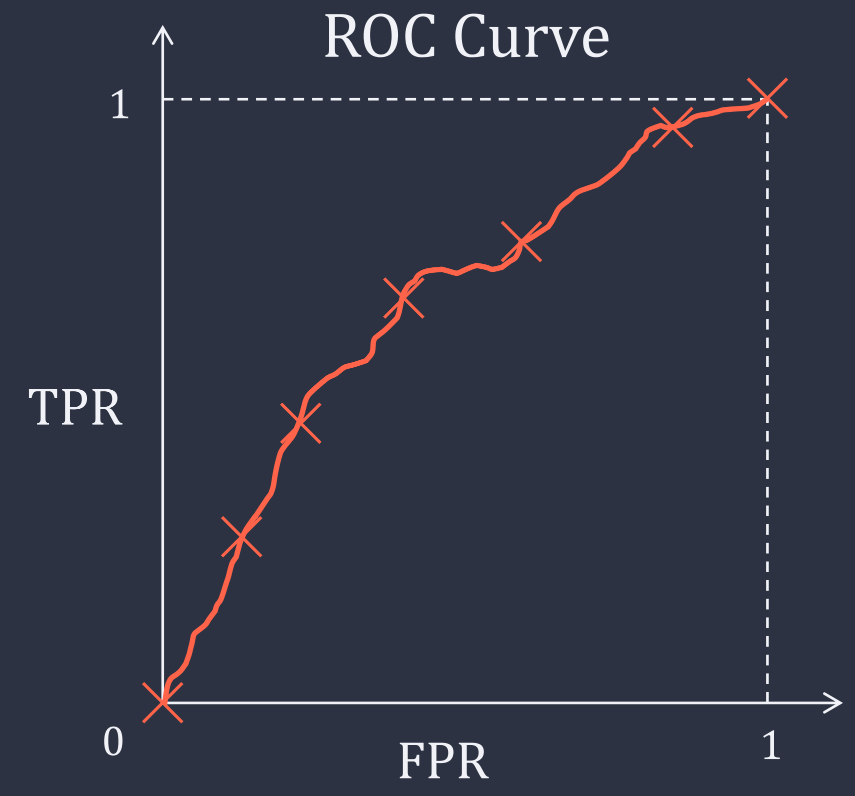
$$FPR = \frac{\cancel{FP}}{\cancel{FP} + TN} = 0$$



x	y	$h(x)$		$\hat{y}$	
	-	h_3	min	0 (-)	
	-	h_1		0 (-)	
...	...	...		...	
	+	h_m		0 (-)	FN
	-	h_6	t	0 (-)	
	+	h_2		1 (+)	
	+	h_5		1 (+)	
...	...	...		...	
	+	h_4	max	1 (+)	

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

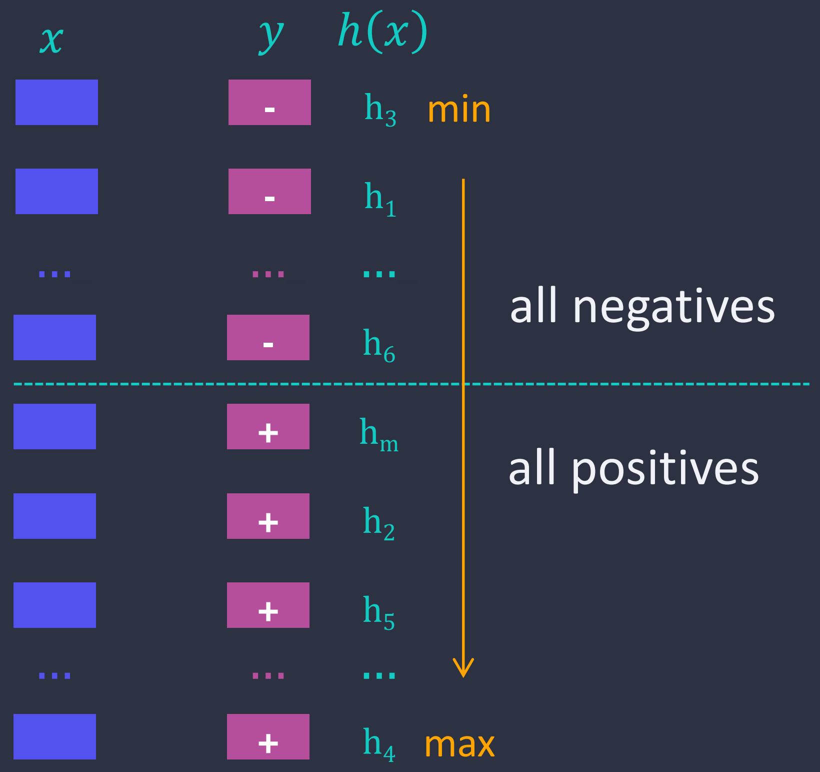
$$FPR = \frac{FP}{FP + TN}$$



(ROC: receiver operating characteristic)

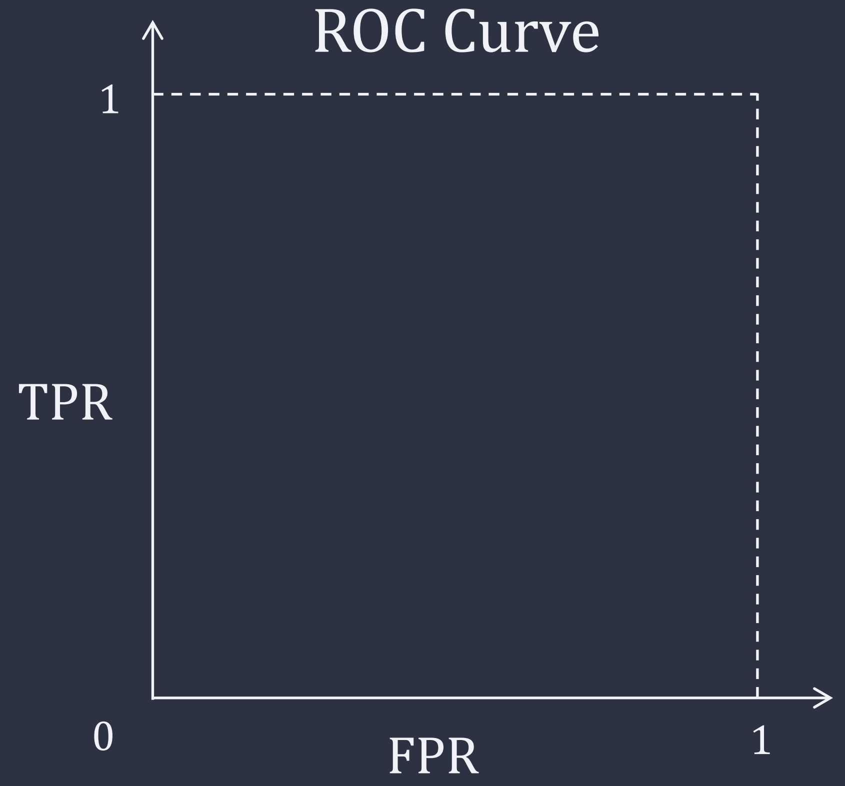
The best (idealistic) $h(X)$

All positive examples receive higher scores than all negative examples.



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

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



(ROC: receiver operating characteristic)

The best (idealistic) $h(X)$

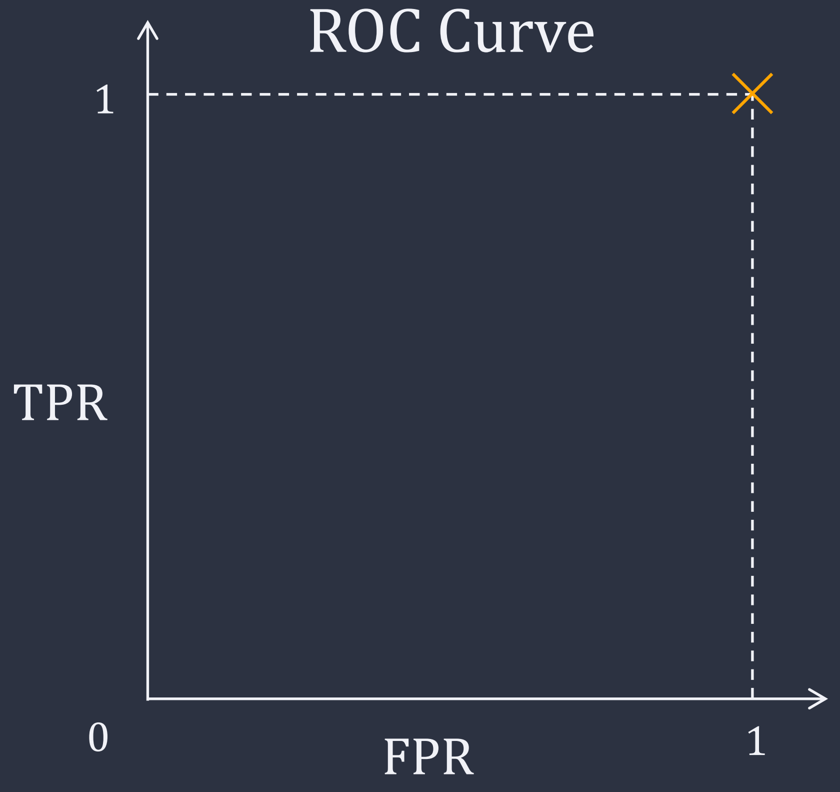
All positive examples receive higher scores than all negative examples.

x	y	$h(x)$		$\hat{y}$	
	-	h_3	min	1 (+)	FP
	-	h_1		1 (+)	FP
…	…	…		…	…
	-	h_6		1 (+)	FP
	+	h_m		1 (+)	TP
	+	h_2		1 (+)	TP
	+	h_5		1 (+)	TP
…	…	…		…	…
	+	h_4	max	1 (+)	TP

t

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

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



(ROC: receiver operating characteristic)

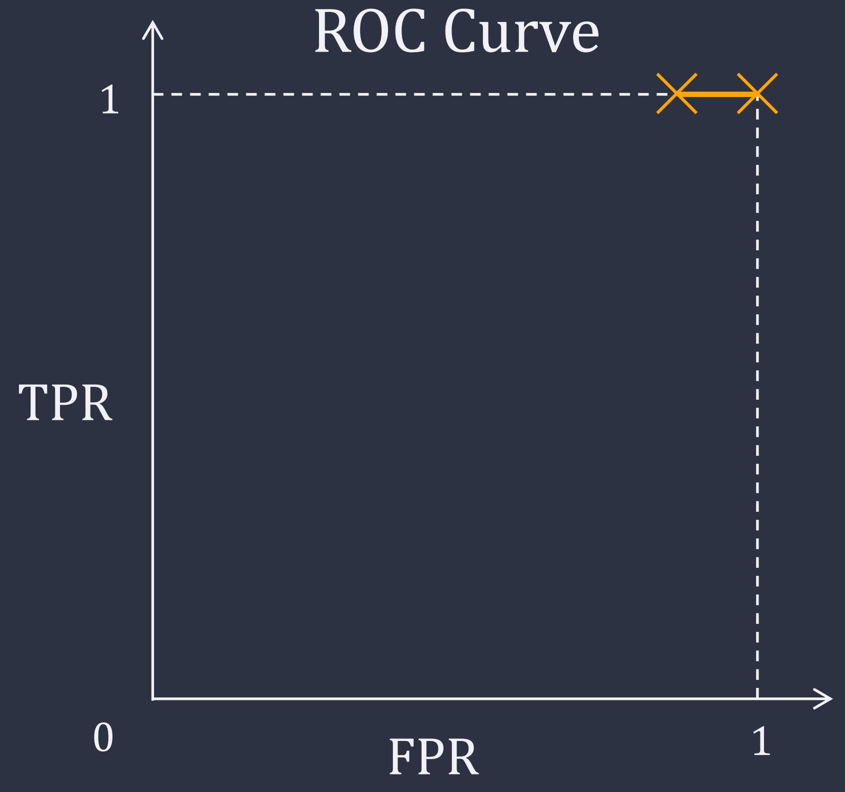
The best (idealistic) $h(X)$

All positive examples receive higher scores than all negative examples.

x	y	$h(x)$		$\hat{y}$	
	-	h_3	t min	0 (-)	TN
	-	h_1		1 (+)	FP
…	…	…		…	…
	-	h_6		1 (+)	FP
	+	h_m		1 (+)	TP
	+	h_2		1 (+)	TP
	+	h_5		1 (+)	TP
…	…	…		…	…
	+	h_4	max	1 (+)	TP

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

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



(ROC: receiver operating characteristic)

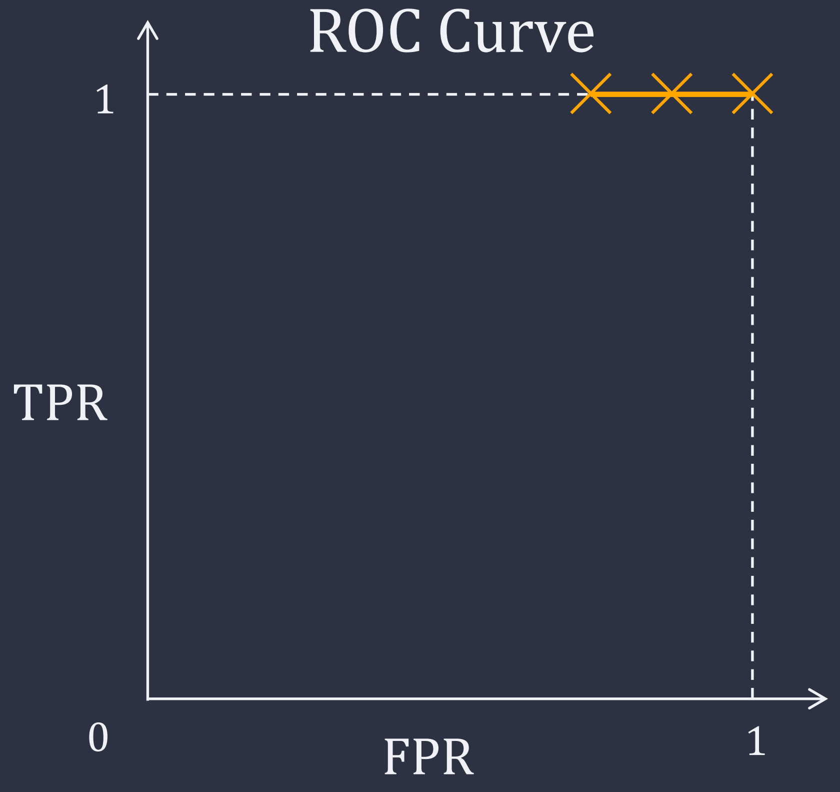
The best (idealistic) $h(X)$

All positive examples receive higher scores than all negative examples.

x	y	$h(x)$		$\hat{y}$	
	-	h_3	min	0 (-)	TN
	-	h_1	t	0 (-)	TN
…	…	…		…	…
	-	h_6		1 (+)	FP
	+	h_m		1 (+)	TP
	+	h_2		1 (+)	TP
	+	h_5		1 (+)	TP
…	…	…		…	…
	+	h_4	max	1 (+)	TP

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

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



(ROC: receiver operating characteristic)

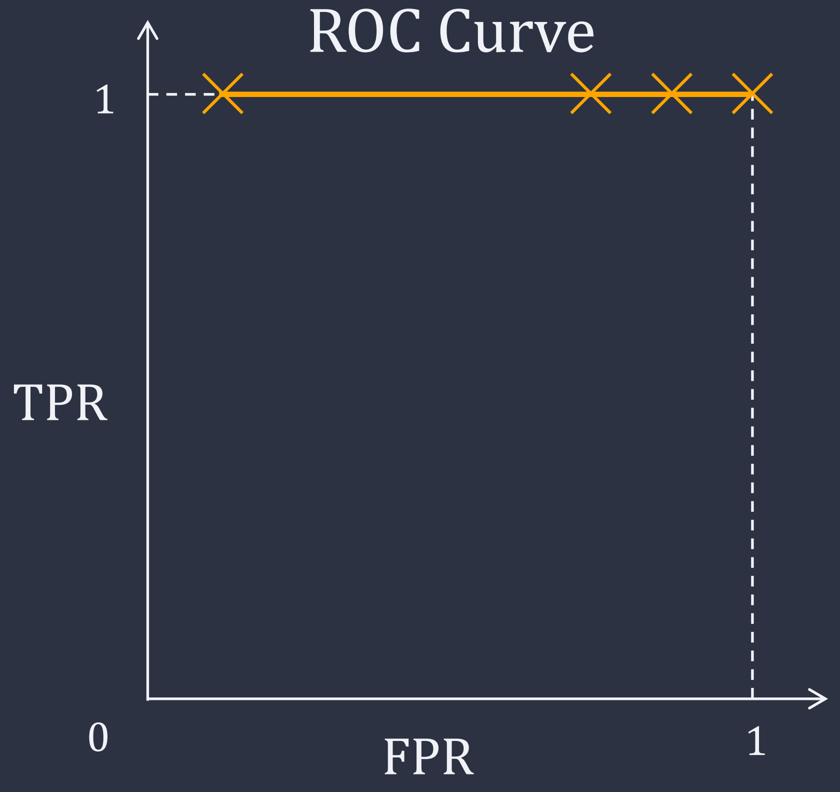
The best (idealistic) $h(X)$

All positive examples receive higher scores than all negative examples.

x	y	$h(x)$		$\hat{y}$	
	-	h_3	min	0 (-)	TN
	-	h_1		0 (-)	TN
…	…	…	t	…	…
	-	h_6		1 (+)	FP
	+	h_m		1 (+)	TP
	+	h_2		1 (+)	TP
	+	h_5		1 (+)	TP
…	…	…		…	…
	+	h_4	max	1 (+)	TP

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

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



(ROC: receiver operating characteristic)

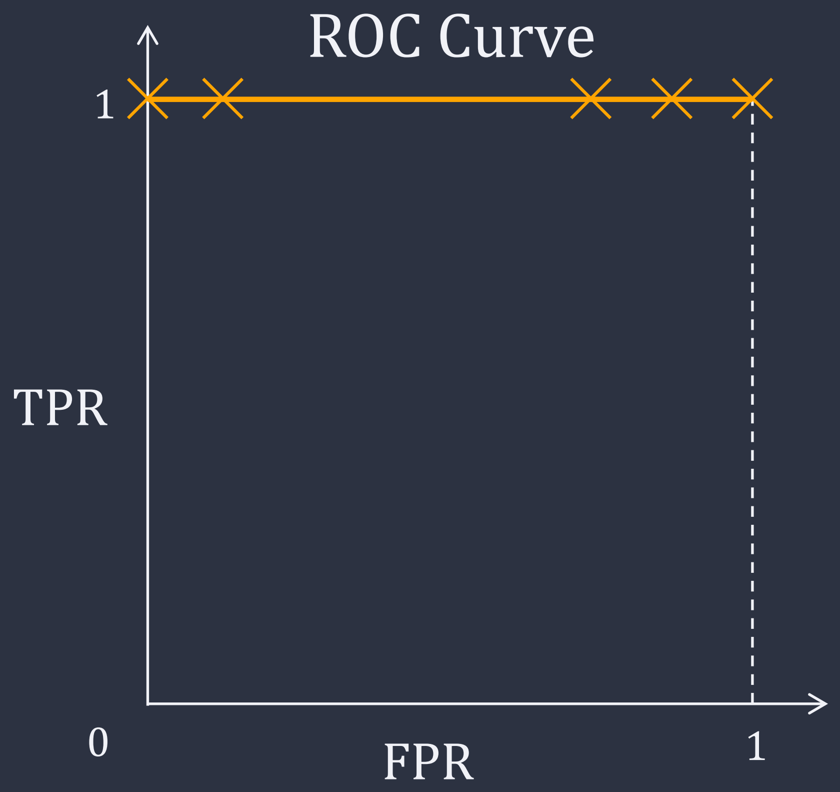
The best (idealistic) $h(X)$

All positive examples receive higher scores than all negative examples.

x	y	$h(x)$		$\hat{y}$	
	-	h_3	min	0 (-)	TN
	-	h_1		0 (-)	TN
…	…	…		…	…
	-	h_6	t	0 (-)	TN
	+	h_m		1 (+)	TP
	+	h_2		1 (+)	TP
	+	h_5		1 (+)	TP
…	…	…		…	…
	+	h_4	max	1 (+)	TP

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

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



(ROC: receiver operating characteristic)

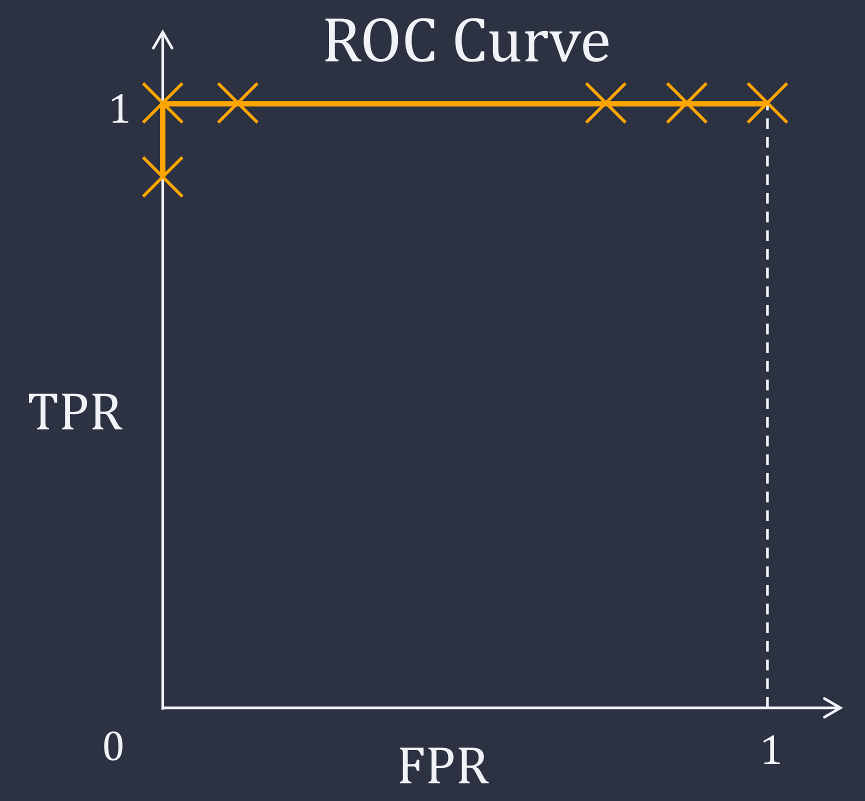
The best (idealistic) $h(X)$

All positive examples receive higher scores than all negative examples.

x	y	$h(x)$		$\hat{y}$	
	-	h_3	min	0 (-)	TN
	-	h_1		0 (-)	TN
…	…	…		…	…
	-	h_6		0 (-)	TN
	+	h_m	t	0 (-)	FN
	+	h_2		1 (+)	TP
	+	h_5		1 (+)	TP
…	…	…		…	…
	+	h_4	max	1 (+)	TP

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

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



(ROC: receiver operating characteristic)

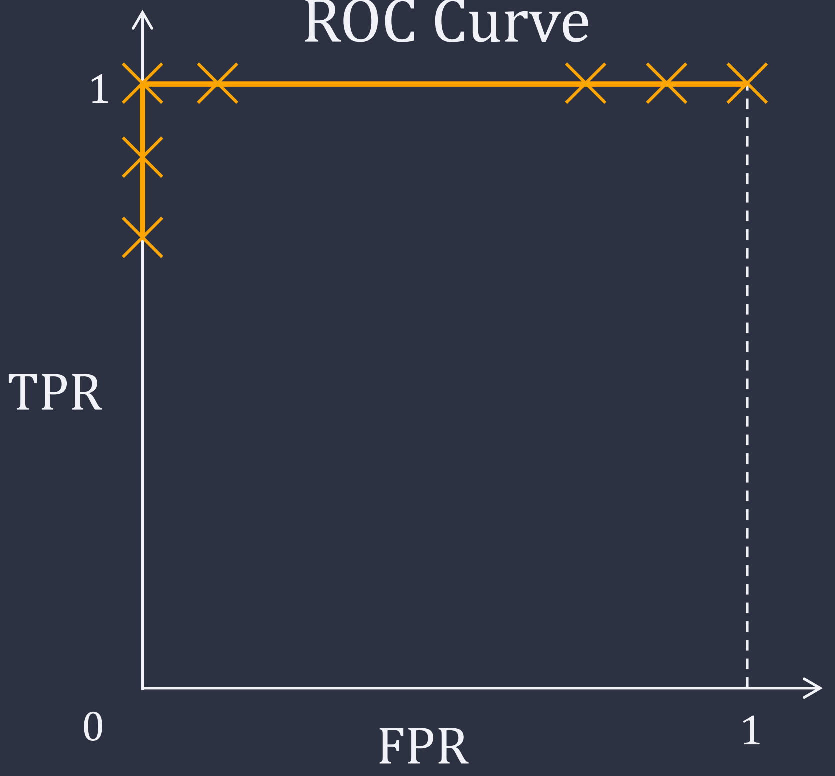
The best (idealistic) $h(X)$

All positive examples receive higher scores than all negative examples.

x	y	$h(x)$		$\hat{y}$	
	-	h_3	min	0 (-)	TN
	-	h_1		0 (-)	TN
…	…	…		…	…
	-	h_6		0 (-)	TN
	+	h_m		0 (-)	FN
	+	h_2	t	0 (-)	FN
	+	h_5		1 (+)	TP
…	…	…		…	…
	+	h_4	max	1 (+)	TP

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

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



(ROC: receiver operating characteristic)

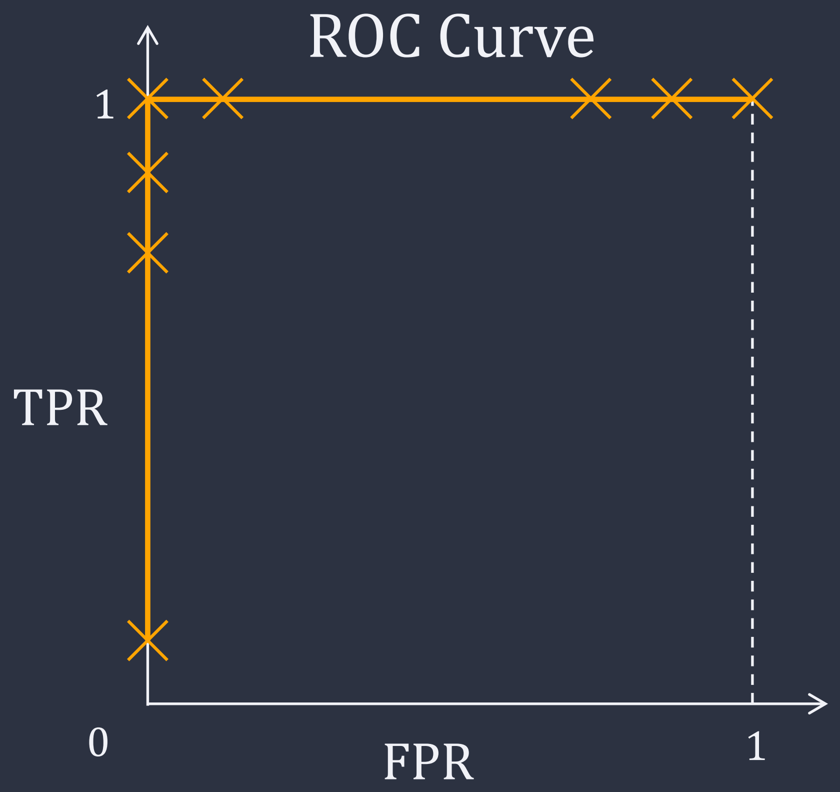
The best (idealistic) $h(X)$

All positive examples receive higher scores than all negative examples.

x	y	$h(x)$	$\hat{y}$	
	-	h_3 min	0 (-)	TN
	-	h_1	0 (-)	TN
…	…	…	…	…
	-	h_6	0 (-)	TN
	+	h_m	0 (-)	FN
	+	h_2	0 (-)	FN
	+	h_5	0 (-)	FN
…	…	… t	…	…
	+	h_4 max	1 (+)	TP

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

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



(ROC: receiver operating characteristic)

The best (idealistic) $h(X)$

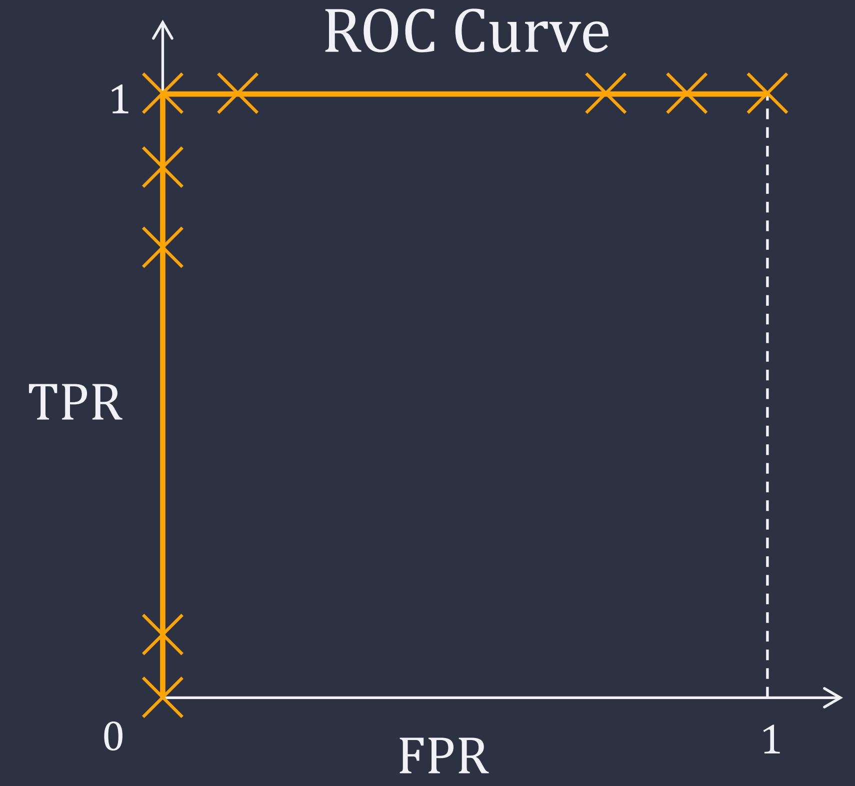
All positive examples receive higher scores than all negative examples.

x	y	$h(x)$		$\hat{y}$	
	-	h_3	min	0 (-)	TN
	-	h_1		0 (-)	TN
…	…	…		…	…
	-	h_6		0 (-)	TN
	+	h_m		0 (-)	FN
	+	h_2		0 (-)	FN
	+	h_5		0 (-)	FN
…	…	…		…	…
	+	h_4	max	0 (-)	FN

t

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

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



(ROC: receiver operating characteristic)

The best (idealistic) $h(X)$

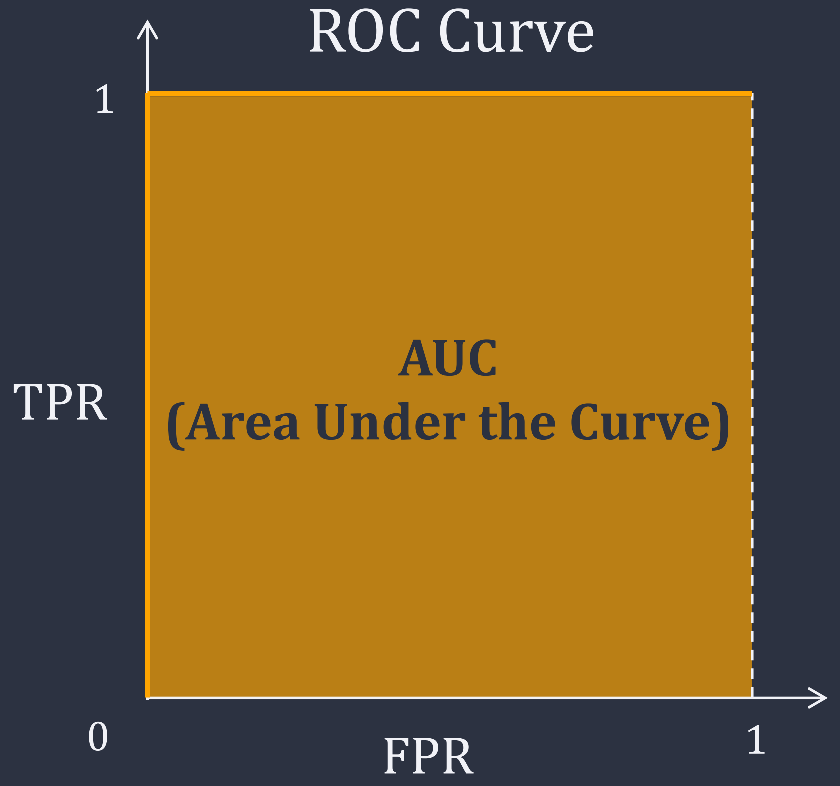
All positive examples receive higher scores than all negative examples.

x	y	$h(x)$	$\hat{y}$	
	-	h_3 min	0 (-)	TN
	-	h_1	0 (-)	TN
…	…	…	…	…
	-	h_6	0 (-)	TN
	+	h_m	0 (-)	FN
	+	h_2	0 (-)	FN
	+	h_5	0 (-)	FN
…	…	…	…	…
	+	h_4 max	0 (-)	FN

t

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

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



(ROC: receiver operating characteristic)

The best (idealistic) $h(X)$

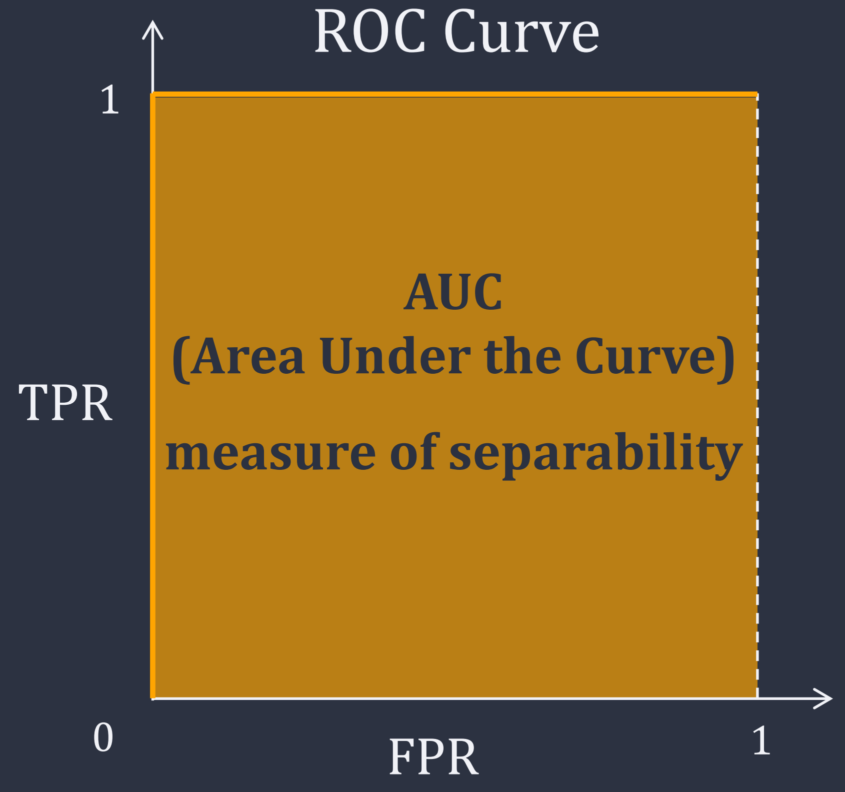
All positive examples receive higher scores than all negative examples.

x	y	$h(x)$		$\hat{y}$	
	-	h_3	min	0 (-)	TN
	-	h_1		0 (-)	TN
…	…	…		…	…
	-	h_6		0 (-)	TN
	+	h_m		0 (-)	FN
	+	h_2		0 (-)	FN
	+	h_5		0 (-)	FN
…	…	…		…	…
	+	h_4	max	0 (-)	FN

t

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

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



(ROC: receiver operating characteristic)

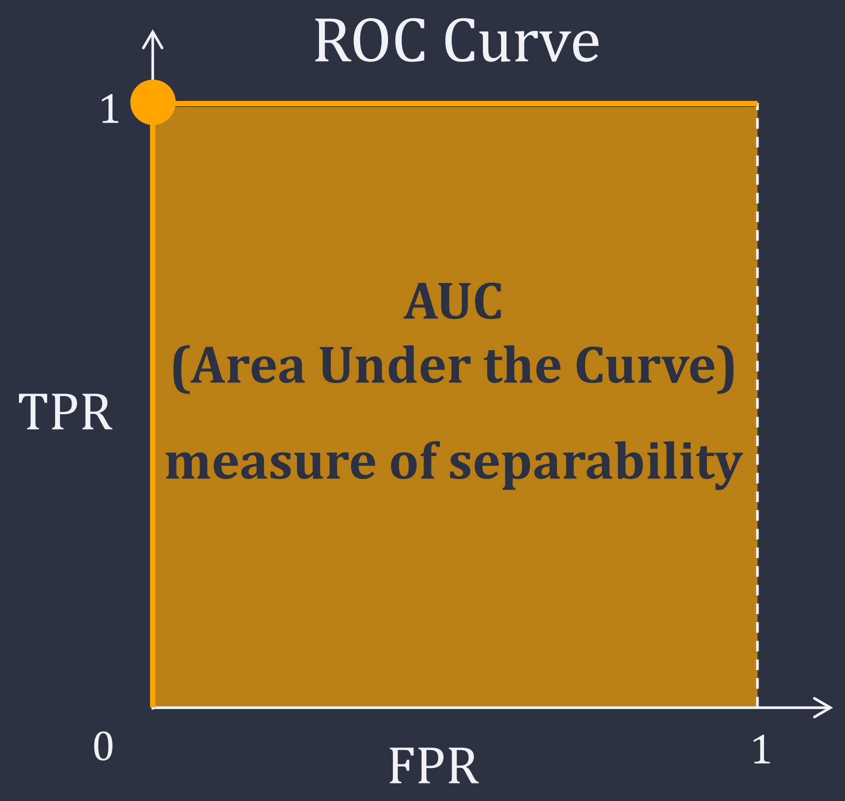
The best (idealistic) $h(X)$

All positive examples receive higher scores than all negative examples.

x	y	$h(x)$		$\hat{y}$	
	-	h_3	min	0 (-)	TN
	-	h_1		0 (-)	TN
…	…	…		…	…
	-	h_6	t	0 (-)	TN
	+	h_m		0 (-)	FN
	+	h_2		0 (-)	FN
	+	h_5		0 (-)	FN
…	…	…		…	…
	+	h_4	max	0 (-)	FN

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

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



(ROC: receiver operating characteristic)

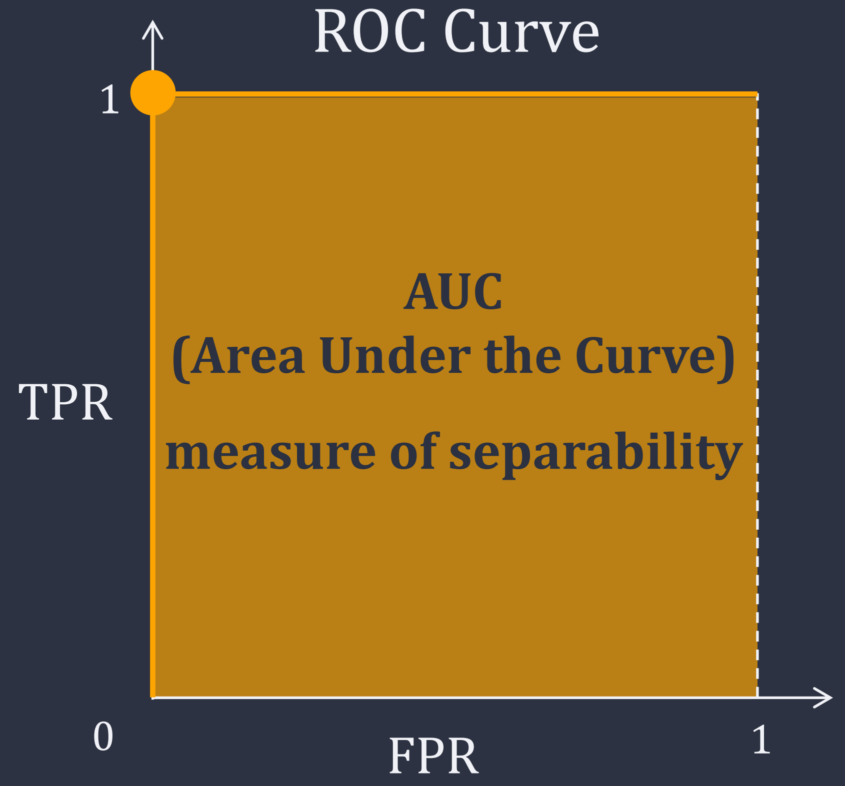
The best (idealistic) $h(X)$

All positive examples receive higher scores than all negative examples.

x	y	$h(x)$		$\hat{y}$	
	-	h_3	min	0 (-)	TN
	-	h_1		0 (-)	TN
…	…	…		…	…
	-	h_6	t	0 (-)	TN
	+	h_m		0 (-)	FN
	+	h_2		0 (-)	FN
	+	h_5		0 (-)	FN
…	…	…		…	…
	+	h_4	max	0 (-)	FN

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

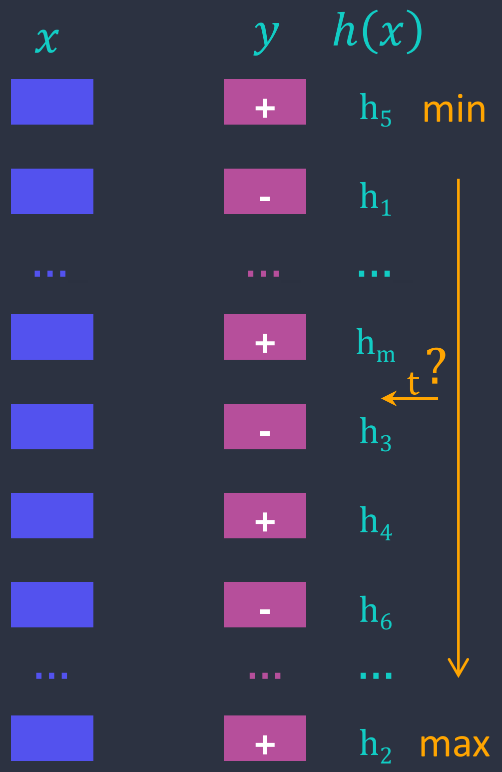
$$FPR = \frac{FP}{FP + TN}$$



(ROC: receiver operating characteristic)

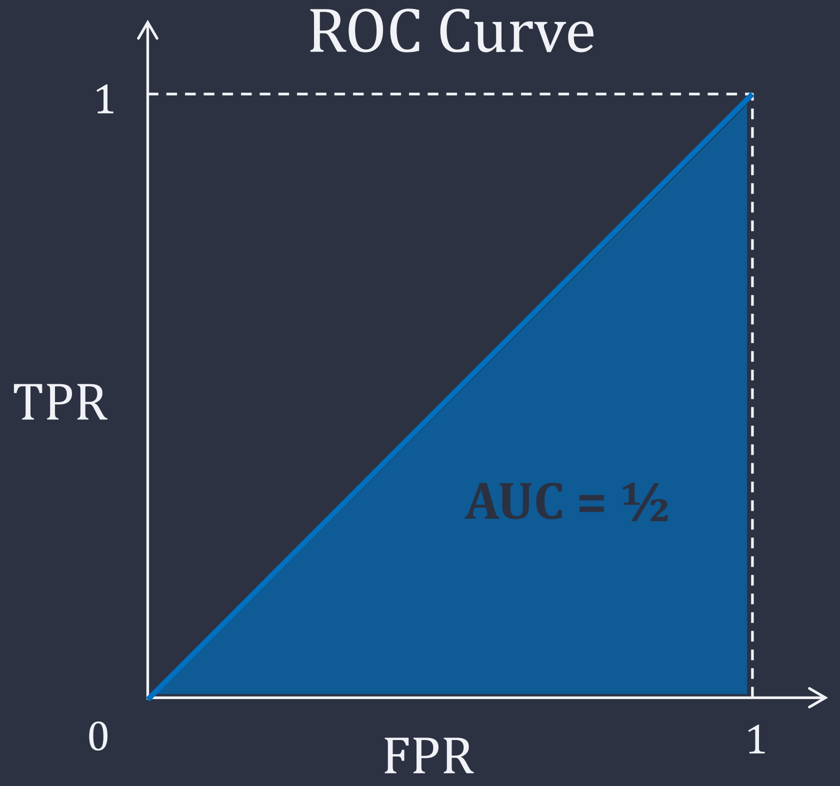
The worst $h(X)$

Mixture of positives and negatives in random way.



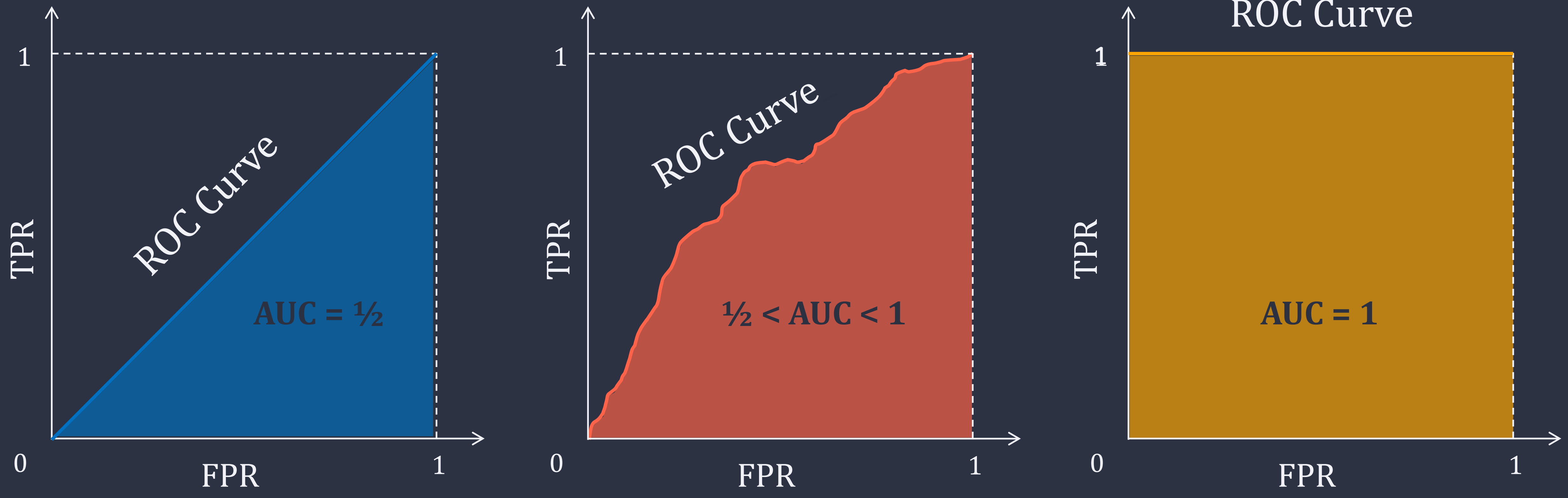
$$TPR = \frac{TP}{TP + FN}$$

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



(ROC: receiver operating characteristic)

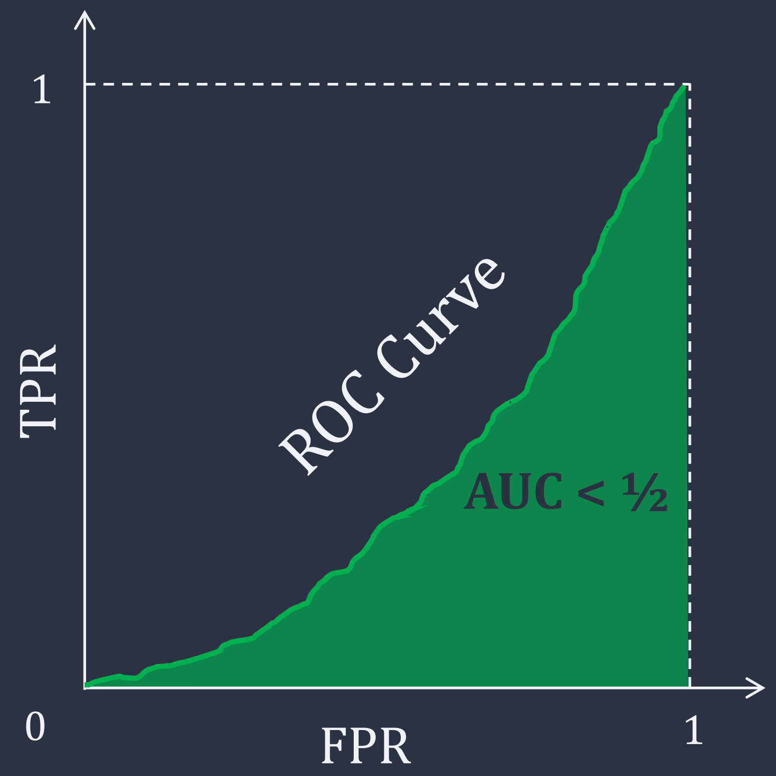
The worst $h(X)$ | The 'normal' $h(X)$ | The 'idealistic' $h(X)$



Random classifications | With some misclassifications | 100% correct classifications

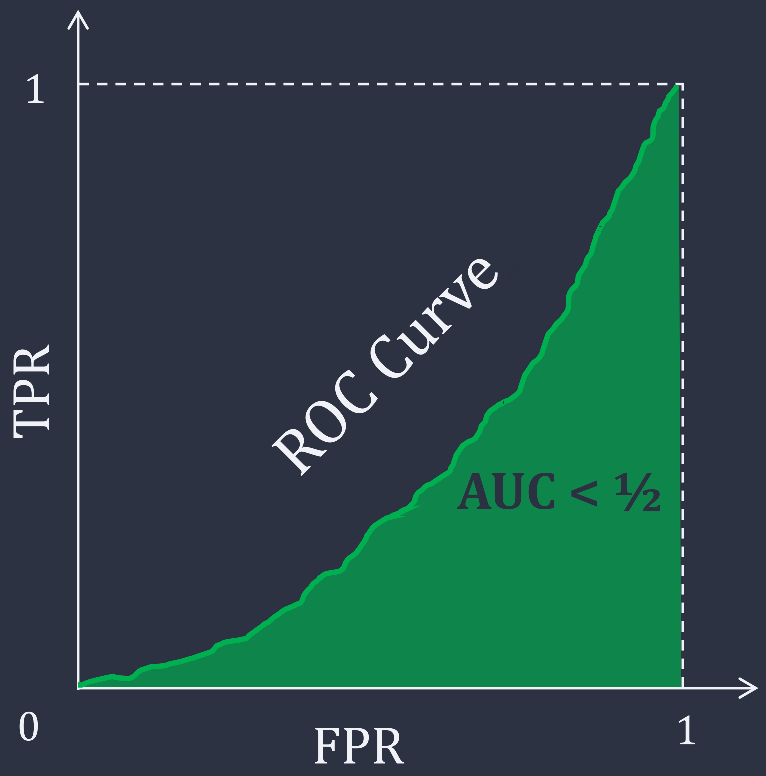
The higher the AUC value, the better the classifier performs.

?



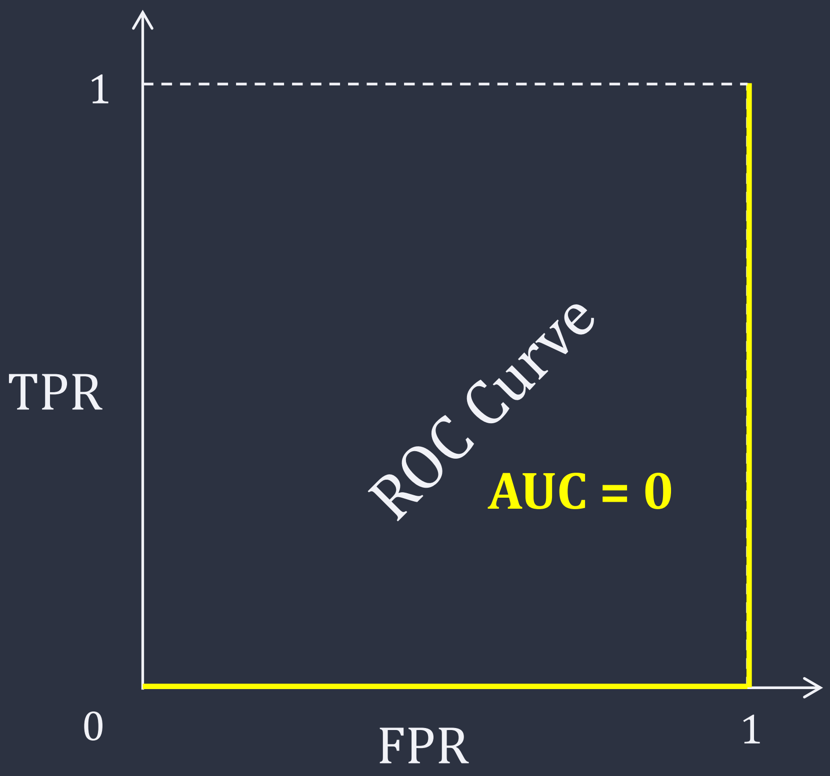
Reciprocating – predicting negatives as positives and positives as negatives.

?



Reciprocating – predicting negatives as positives and positives as negatives.

?



Making perfectly opposite predictions.

sklearn.metrics

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics

metrics.accuracy_score(y_true, y_pred, *[, …])
metrics.auc(x, y)
metrics.f1_score(y_true, y_pred, *[, …])
metrics.precision_recall_curve(y_true, …)
metrics.precision_recall_fscore_support(…)
metrics.precision_score(y_true, y_pred, *[, …])
metrics.recall_score(y_true, y_pred, *[, …])
metrics.roc_auc_score(y_true, y_score, *[, …])
metrics.roc_curve(y_true, y_score, *[, …])
…

Summary

Summary

1. Accuracy
2. Confusion Matrix
3. Sensitivity (Recall) & Specificity (Selectivity)
4. Positive Predictive Value (Precision) & Negative Predictive Value
5. ROC & AUC

COMP2261 ARTIFICIAL INTELLIGENCE / MACHINE LEARNING

Cross-Validation & Hyperparameter Tuning & Sub-module Wrap-up

Dr Yang Long

Durham University

Lecture Overview

1. Cross-Validation
2. Hyperparameter Tuning
3. Sub-module Wrap-up

1. Cross-Validation

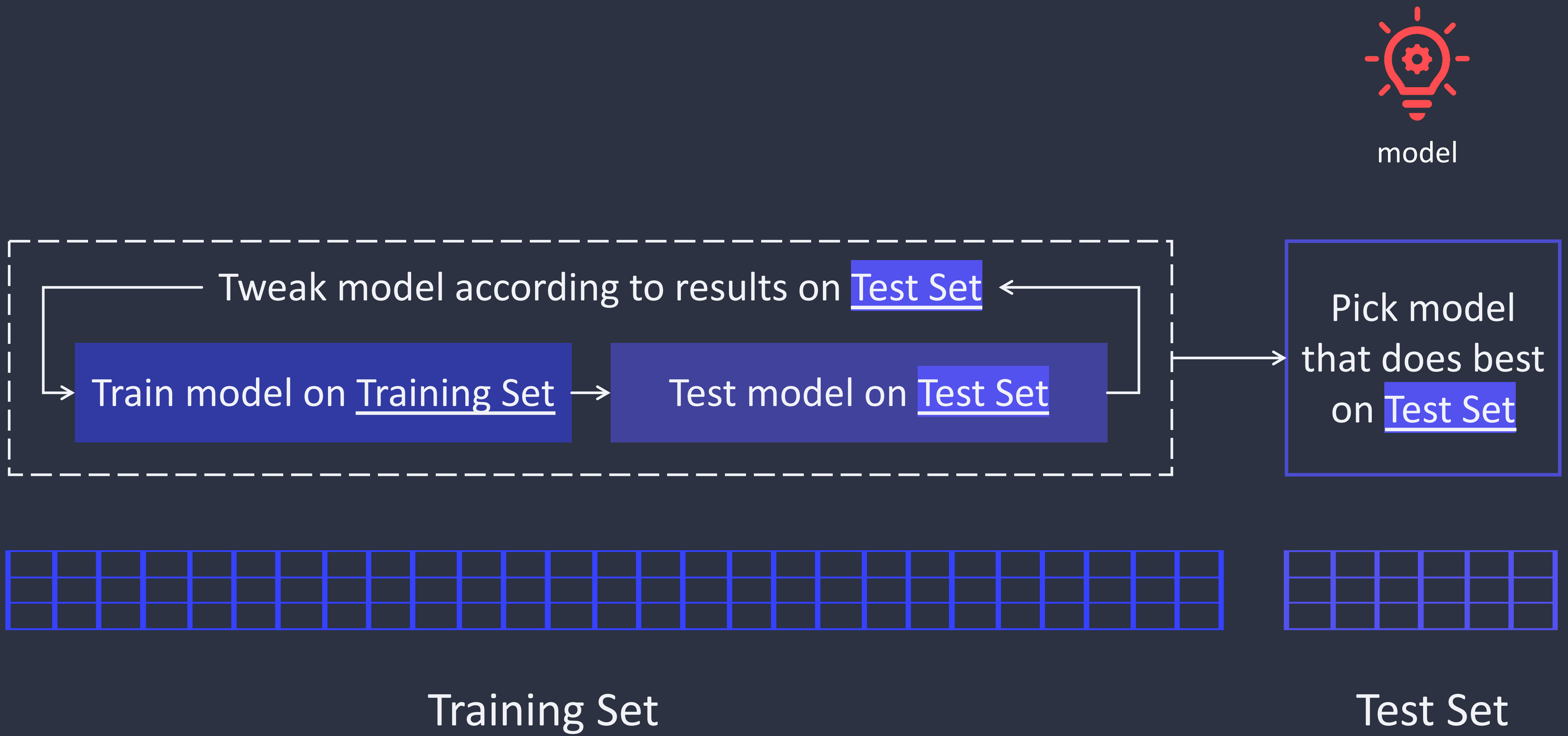
In a machine learning project…

Data

Evaluation metrics

Divide data for training and testing…

Data


model
Tweak model according to results on Test Set
Train model on Training Set
Test model on Test Set
Pick model that does best on Test Set
Training Set
Test Set

Having the testing results on the same evaluation metrics for these trained models
-> choose the one with the best performance.

final model 1
+ testing result 1

final model 2
+ testing result 2

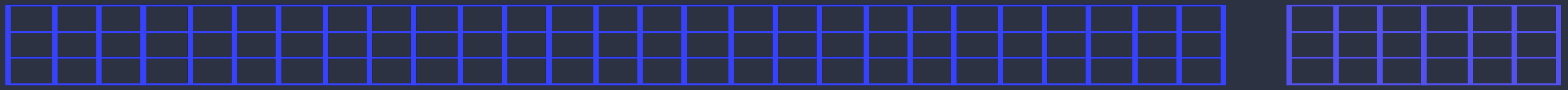
…

final model m
+ testing result m

Tweak model according to results on Test Set

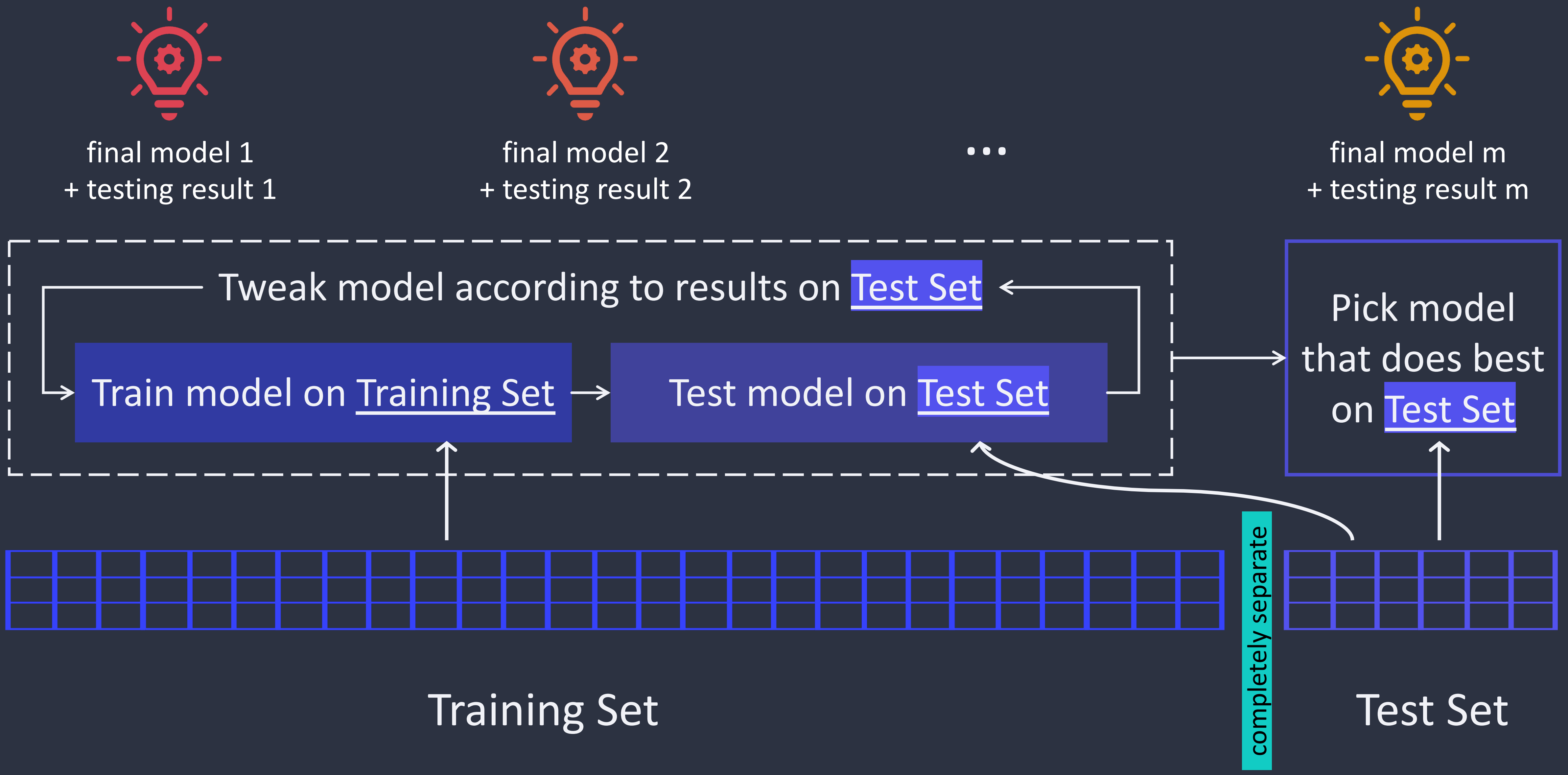
Train model on Training Set → Test model on Test Set

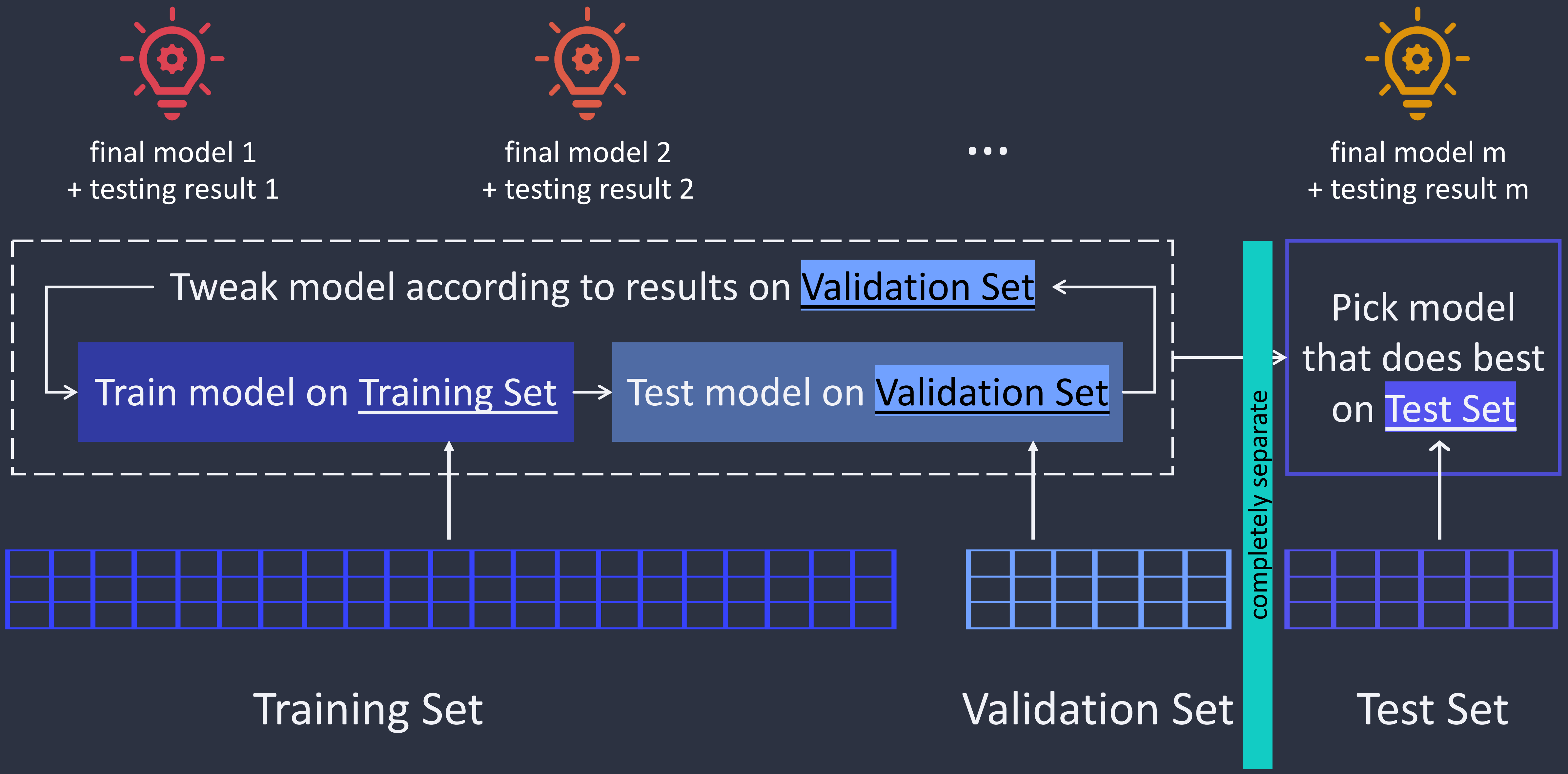
Pick model that does best on Test Set



Training Set

Test Set


final model 1
+ testing result 1
final model 2
+ testing result 2
...
final model m
+ testing result m
Tweak model according to results on Test Set
Train model on Training Set
Test model on Test Set
Pick model that does best on Test Set
completely separate
Training Set
Test Set
Durham University

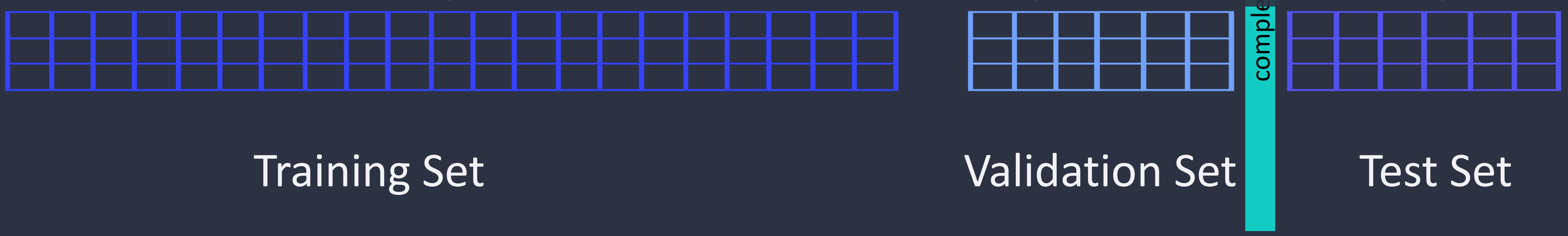


Durham University

However…

- The size of the Training Set is reduced, which the models are trained on.
- Not sure if the dataset is split in the best way.

  (results may depend on a particular random choice of data split.)

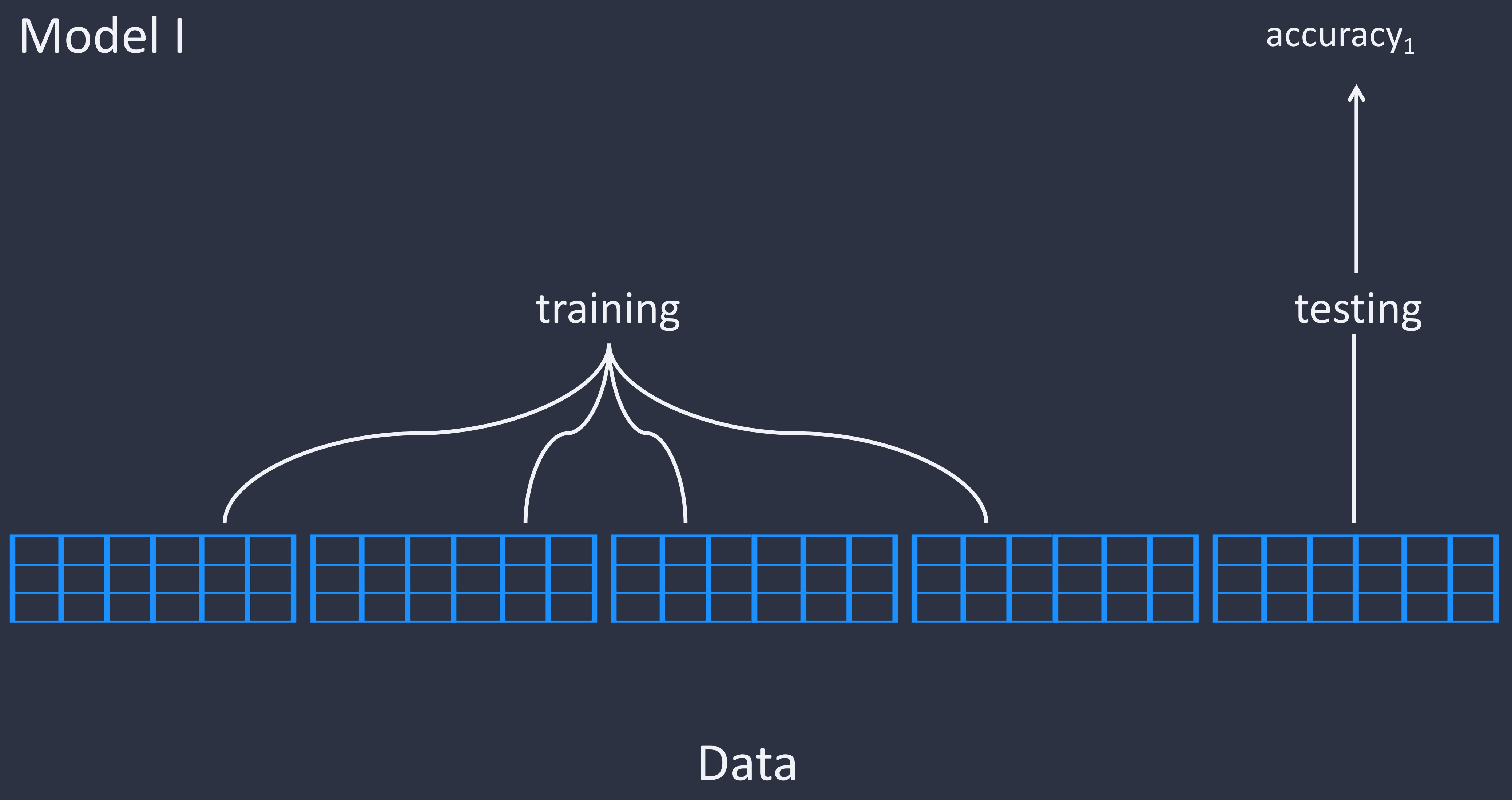


Cross Validation

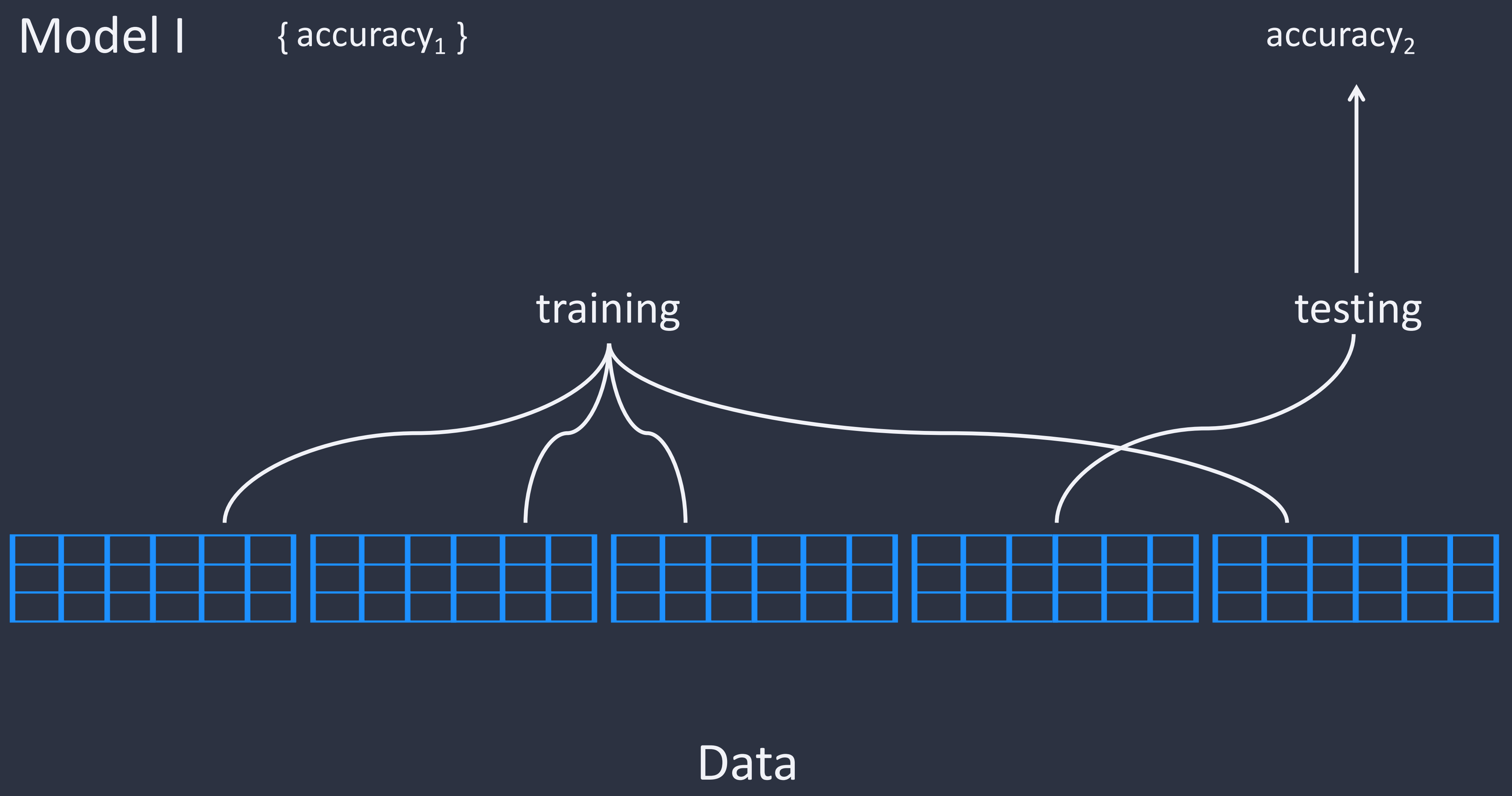
Cross Validation (rotation estimation / out-of-sample testing)

- Reduce the change of overfitting.
- Assess how well a model performs on previously unseen data.
- Is a resampling procedure to test models on a limited data sample.

Data



Model I
accuracy$_1$
training
testing
Data

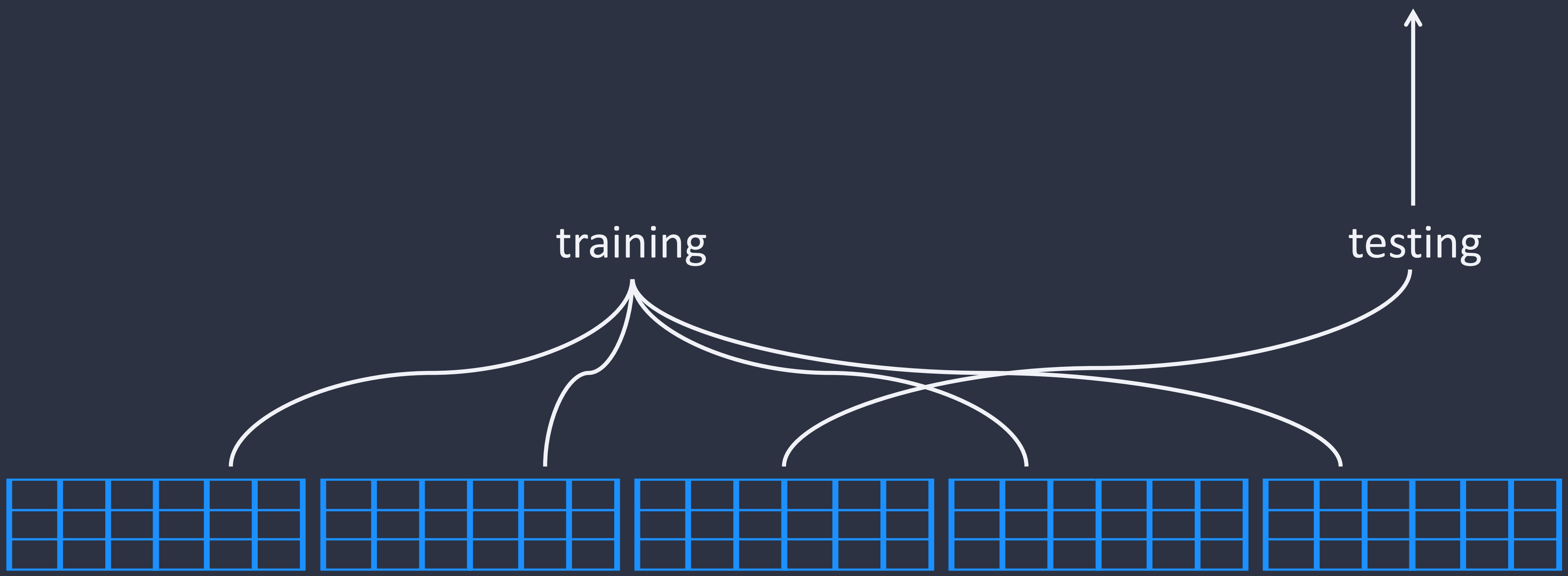


Model I
{ accuracy$_1$ }
accuracy$_2$
training
testing
Data

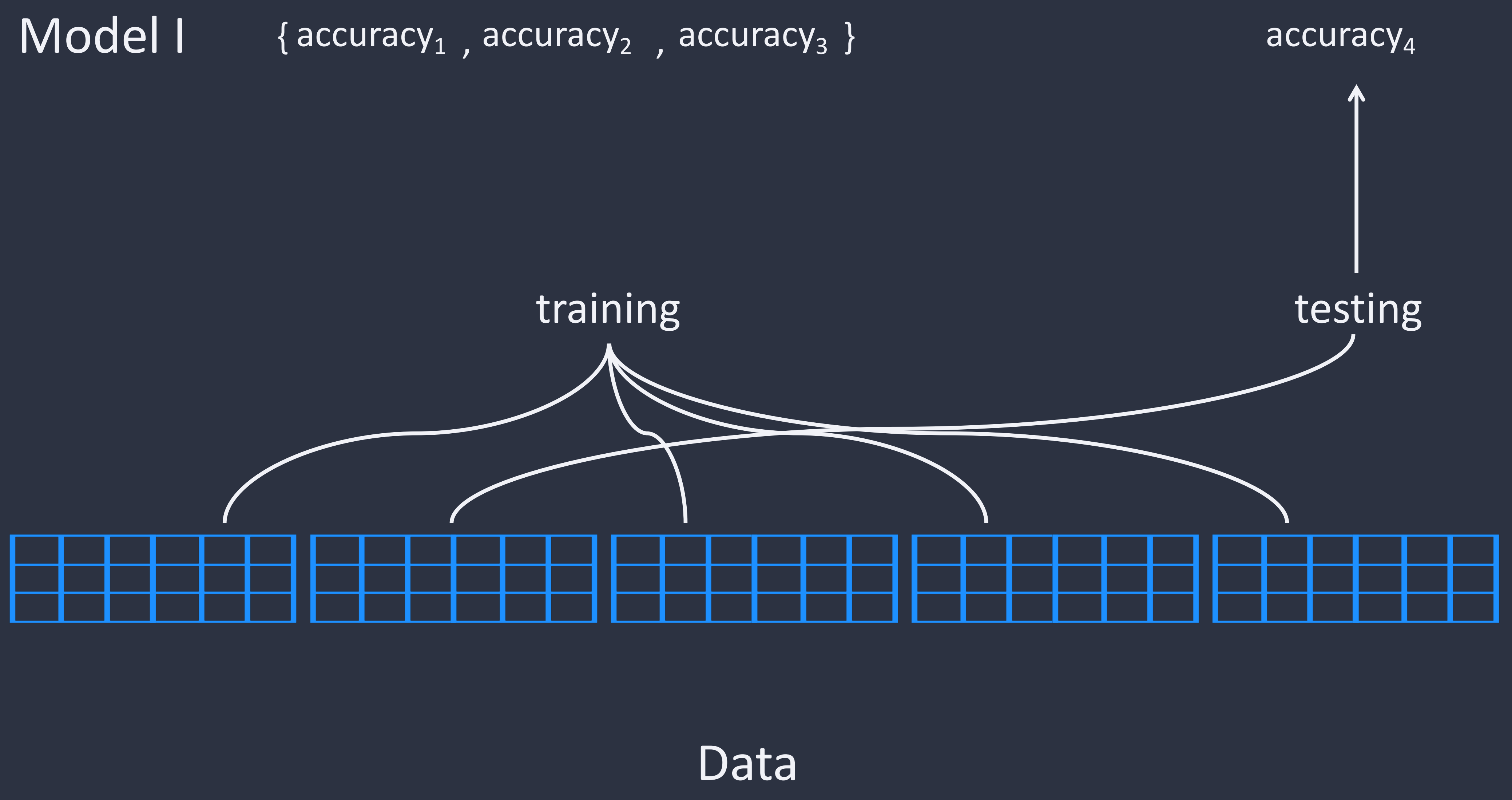
Model I

$\{ \text{accuracy}_1 , \text{accuracy}_2 \}$

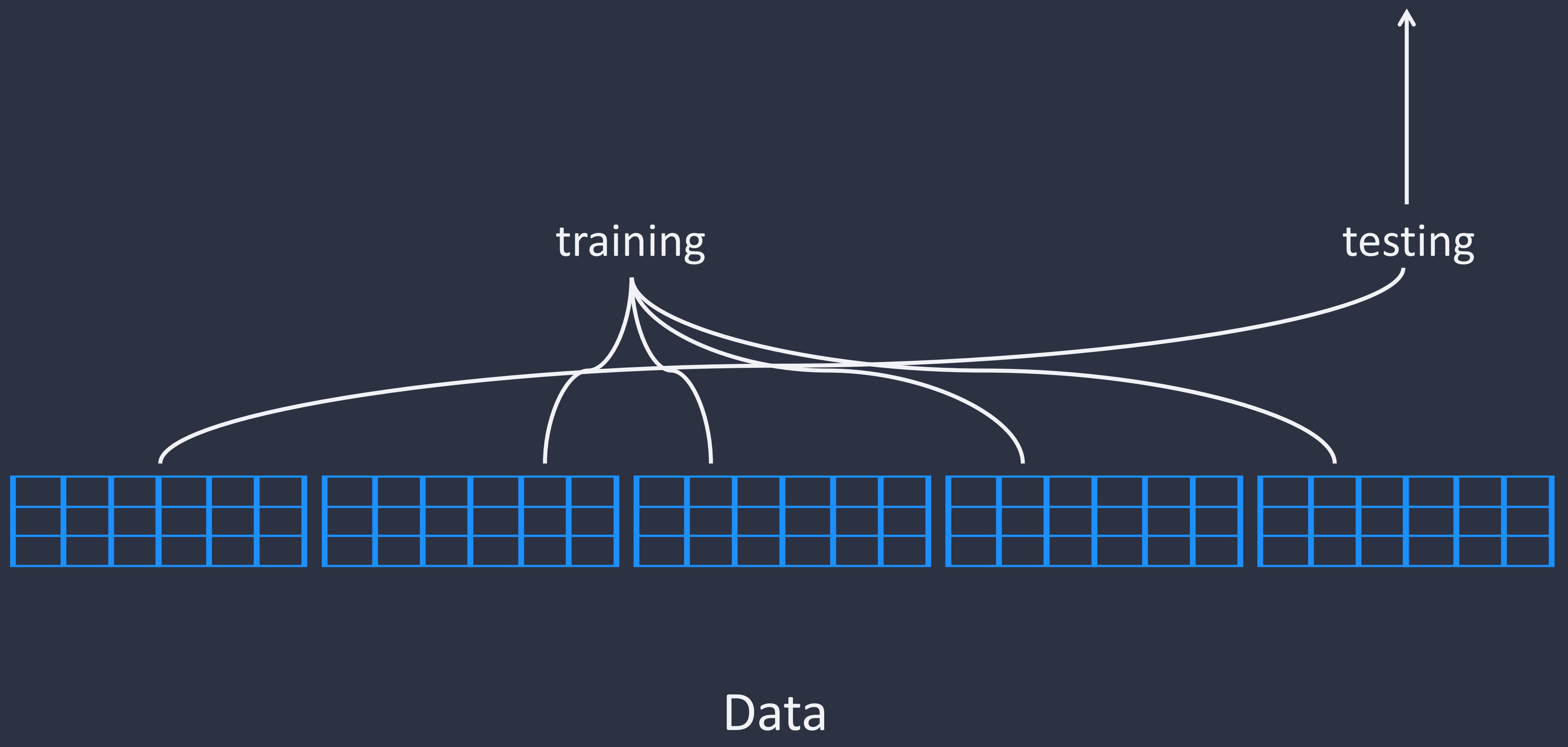
accuracy_3



Data


Model I
{ accuracy$_1$, accuracy$_2$, accuracy$_3$ }
accuracy$_4$
training
testing
Data

Model I $\{ accuracy_1, accuracy_2, accuracy_3, accuracy_4 \}$ $accuracy_5$



Summarise the result for the trained model

$$\text{Accuracy}_{\text{Model I}} = \frac{1}{5}\left(\text{accuracy}_1 + \text{accuracy}_2 + \text{accuracy}_3 + \text{accuracy}_4 + \text{accuracy}_5 \right)$$

$$\text{Accuracy}_{\text{Model II}} = \frac{1}{5}\left(\text{accuracy}_1 + \text{accuracy}_2 + \text{accuracy}_3 + \text{accuracy}_4 + \text{accuracy}_5 \right)$$

$$\text{Accuracy}_{\text{Model III}} = \frac{1}{5}\left(\text{accuracy}_1 + \text{accuracy}_2 + \text{accuracy}_3 + \text{accuracy}_4 + \text{accuracy}_5 \right)$$

$$\text{Accuracy}_{\text{Model IV}} = \frac{1}{5}\left(\text{accuracy}_1 + \text{accuracy}_2 + \text{accuracy}_3 + \text{accuracy}_4 + \text{accuracy}_5 \right)$$

$$\text{Accuracy}_{\text{Model V}} = \frac{1}{5}\left(\text{accuracy}_1 + \text{accuracy}_2 + \text{accuracy}_3 + \text{accuracy}_4 + \text{accuracy}_5 \right)$$

Compare the result for each trained model

$\text{Accuracy}_{\text{Model II}}$

$\text{Accuracy}_{\text{Model I}}$

$\text{Accuracy}_{\text{Model III}}$

Which is the best?

$\text{Accuracy}_{\text{Model IV}}$

$\text{Accuracy}_{\text{Model V}}$

5-Fold Cross-Validation

fold 1 fold 2 fold 3 fold 4 fold 5

Data

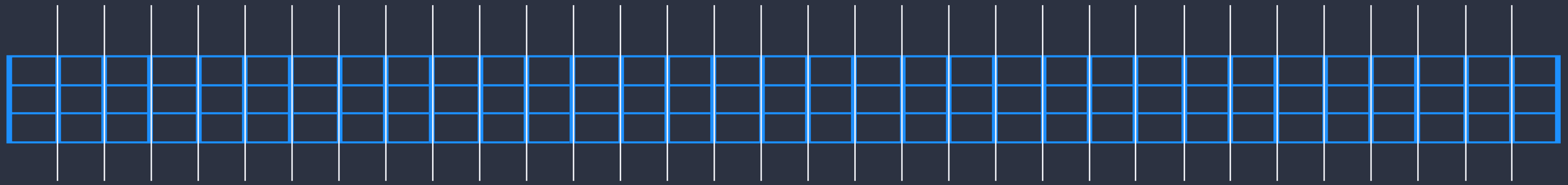
K-Fold Cross-Validation

Data

- k is arbitrary & may depend on size of dataset and how many models to train and compare.
- Larger k means less pessimistic bias (towards overestimating the true expected errors).

Reading about K: https://stats.stackexchange.com/questions/27730/choice-of-k-in-k-fold-cross-validation

Leave-One-Out Cross-Validation (LOOCV)



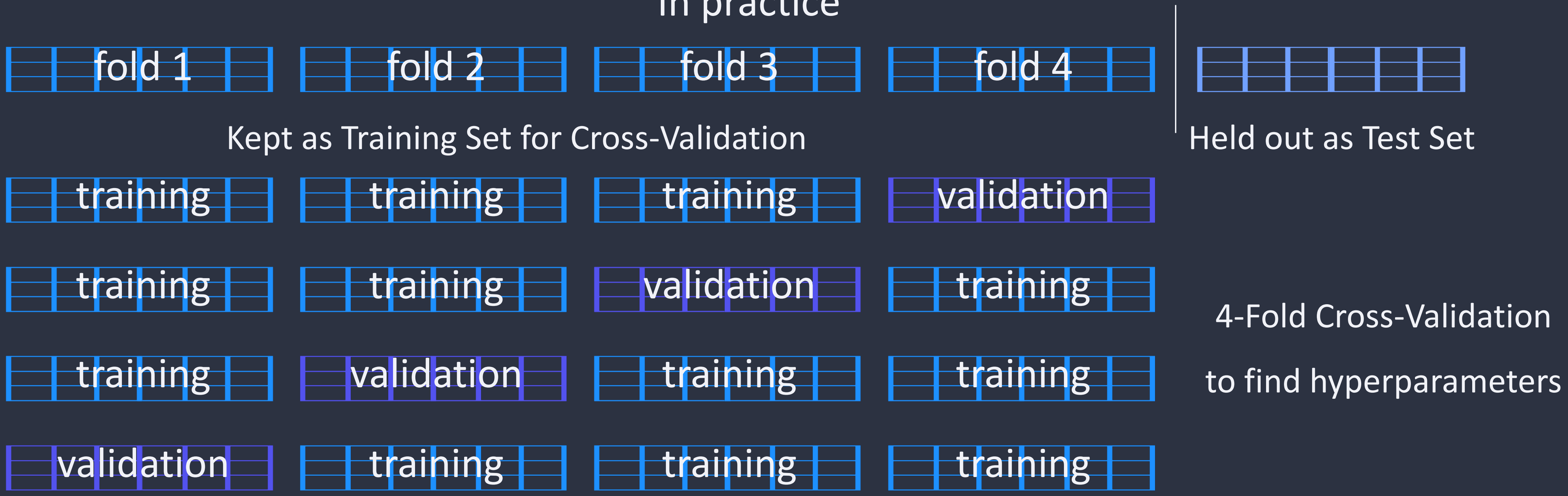
Data

- The largest possible k is equal to the number of instances in the original dataset.
- One instance in each fold – and each instance is tested individually.
- In practice, it's common to split data into 10 fold (k=10).

In practice

- Before cross validation for training, dataset randomly shuffled.
- Then, a subset is held out as Test Set, which will not be used in training at all.
- Test Set will be used only for the final evaluation of trained models.
- The rest dataset is kept as Training Set and to be used during cross validation.

In practice



1. Split Training Set into k smaller folds, of approximately equal size.
2. Now, run training k times, each time:
   - Train a model using k-1 of the folds
   - Validate the resulting model using the remaining 1 fold
3. Calculate the average of the performance measured in the loop, as the K-Fold CV result.

fold 1	fold 2	fold 3	fold 4	
Kept as Training Set for Cross-Validation				Held out as Test Set
training	training	training	validation	
training	training	validation	training	4-Fold Cross-Validation
training	validation	training	training	to find hyperparameters
validation	training	training	training	

Computationally expensive.

Re-uses data, which is very efficient, especially in case of small size of datasets.

Note, each instance is assigned into one fold and it stays in that fold for the whole k-Fold Cross-Validation procedure. This means that each instance has only one chance to be held out in the training Set and has k-1 times being used to train the model.

1. Split Training Set into k smaller folds, of approximately equal size.
2. Now, run training k times, each time:
   - Train a model using k-1 of the folds
   - Validate the resulting model using the remaining 1 fold
3. Calculate the average of the performance measured in the loop, as the K-Fold CV result.

2. Hyperparameter Tuning

What is a Hyperparameter ?

Hyperparameter

A Hyperparameter is a configuration of a learning algorithm and whose value is manually set.

external configuration setting option tunning parameter

- Used in the process of helping estimate model parameters.
- Chosen manually (rules of thumb, copy from other tasks; search by trial & error).
- To be tuned for a given predictive modelling task.

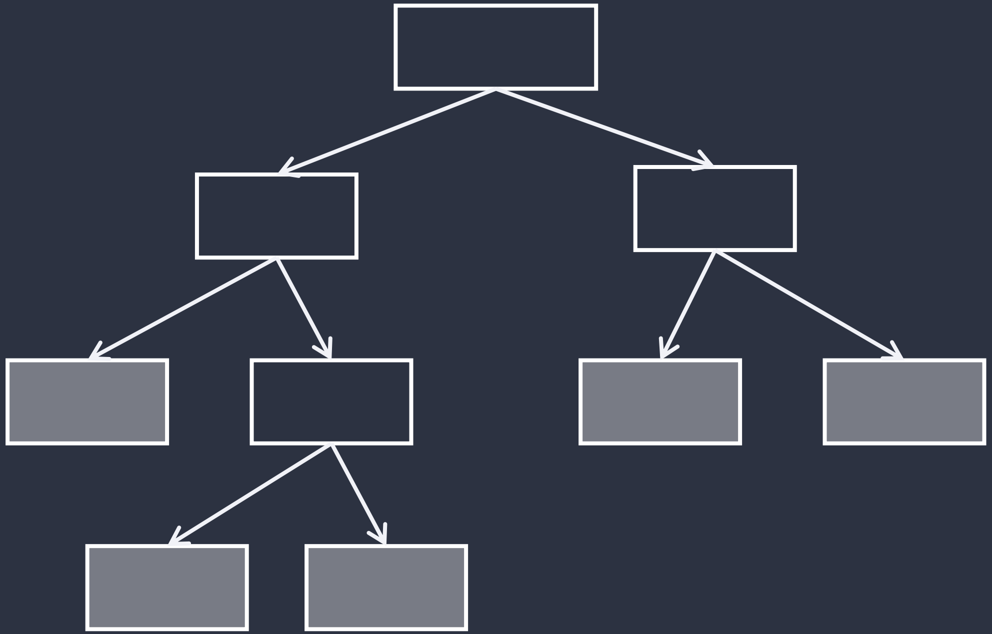
A Parameter is an internal feature of a model and whose value is estimated from data.

internal feature part of model learned from data automatically

- Used by the model when making predictions.
- Estimated or learned from data automatically.
- Saved as part of the trained/learned model.

Hyperparameter

EXAMPLE. Decision Tree



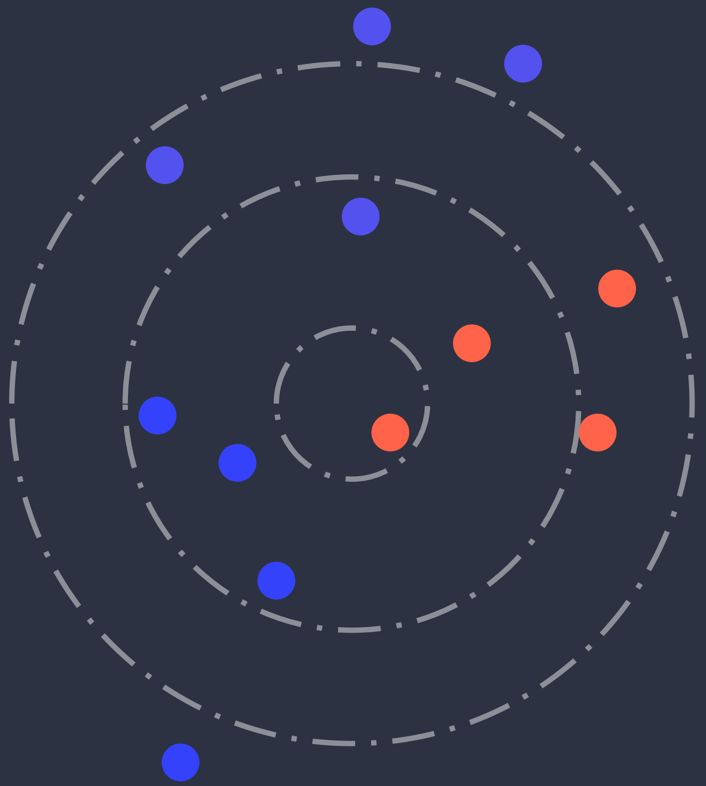
```
class sklearn.tree.DecisionTreeClassifier(*,
criterion='gini', splitter='best', max_depth=None,
min_samples_split=2, min_samples_leaf=1,
min_weight_fraction_leaf=0.0, max_features=None,
random_state=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
class_weight=None, ccp_alpha=0.0)
```

```
criterion: {“gini”, “entropy”}, default=”gini”
splitter: {“best”, “random”}, default=”best”
max_depth: int, default=None
min_sample_split: int or float, default=2
…
```

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Hyperparameter

EXAMPLE. k Nearest Neighbours



```
class sklearn.neighbors.KNeighborsClassifier(
n_neighbors=5, *, weights='uniform',
algorithm='auto', leaf_size=30, p=2,
metric='minkowski', metric_params=None, n_jobs=None,
**kwargs)
```

```
n_neighbors: int, default=5

weights: {'uniform', 'distance'} or callable,
default='uniform'

algorithm: {'auto', 'ball_tree', 'kd_tree',
'brute'}, default='auto'
…
```

https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

Hyperparameter

Why is it important?

Control Model Capacity

- How flexible the model is.
- How many degrees of freedom it has in fitting data.

- To prevent overfitting.

Control Training Process

- Learning rate.
- Convergence threshold.

Hyperparameter Tuning Mechanism

Hyperparameter Tuning Mechanism

Hyperparameter tuning is an optimisation task

- like model training, an optimisation task to seek best model parameters.

Model training

The proposed set of model parameters can be formalised in math formulas (cost function.)

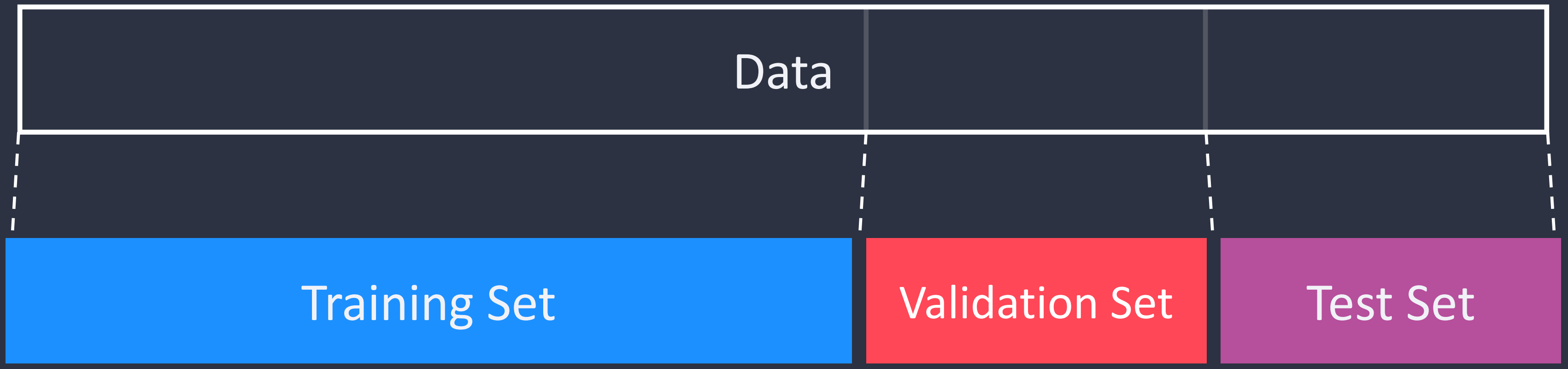
Hyperparameter tuning

No closed-form formula for hyperparameters, as it depends on the outcome of a black box (model training process)

Hyperparameter Tuning Mechanism

3-Way Holdout Method

1 3-Way Split



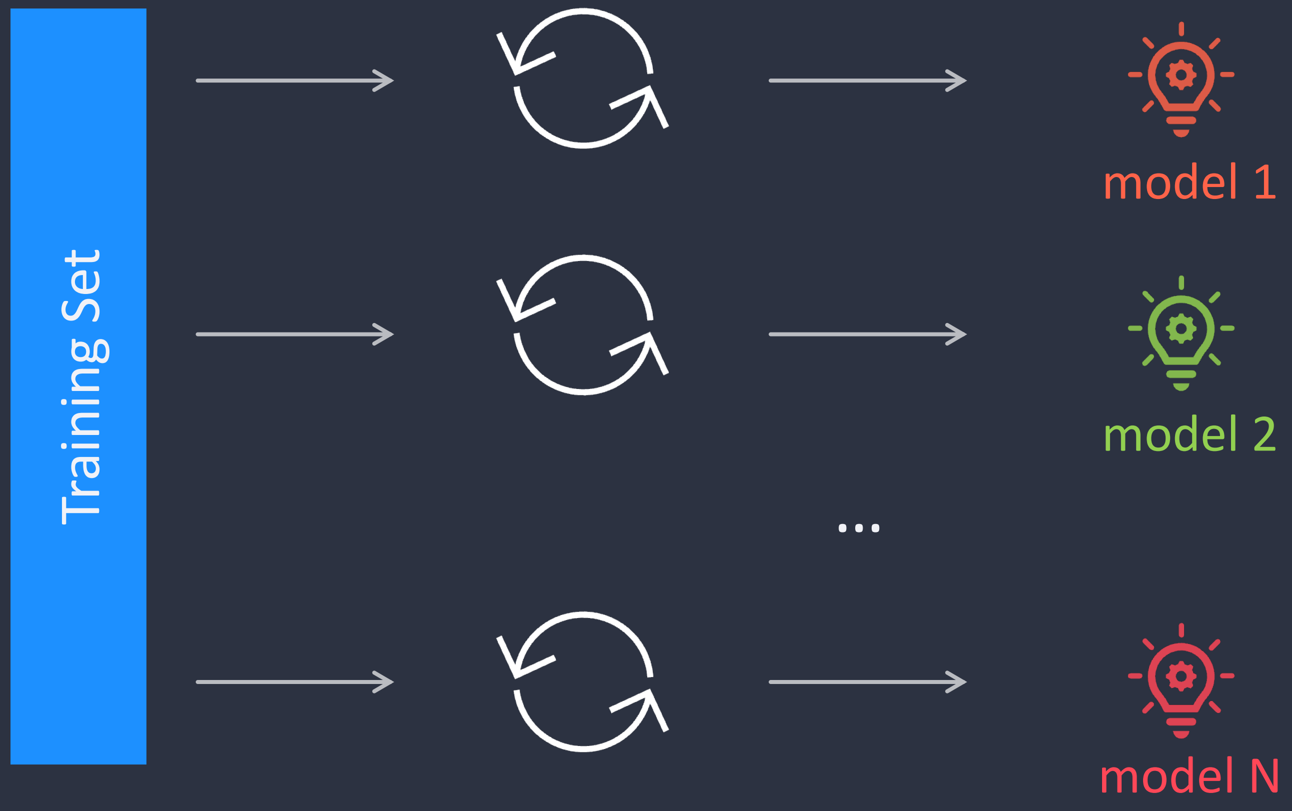
Hyperparameter Tuning Mechanism

e.g. kNN

n_neighbors	metric
3	“euclidean”
5	“manhattan”
9	“minkowski”

3-Way Holdout Method

2 Different sets of hyperparameters to fit different models

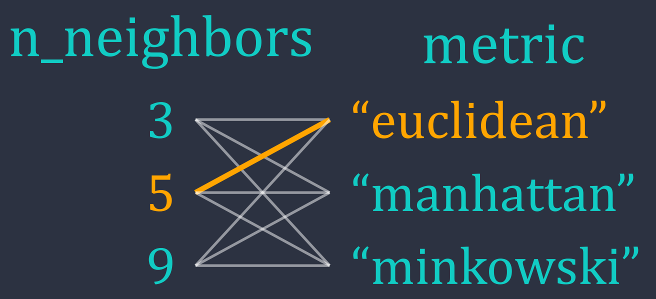


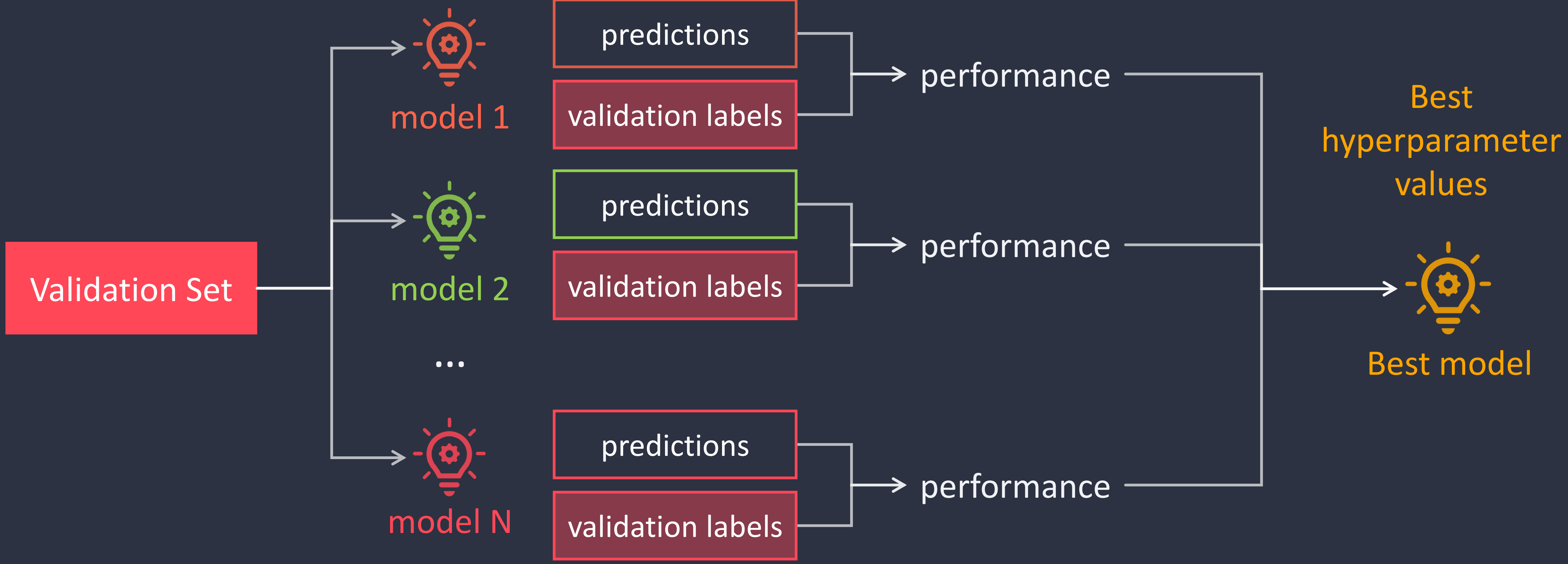
Hyperparameter Tuning Mechanism

3-Way Holdout Method

③ Using validation set to compare model performance

e.g. kNN





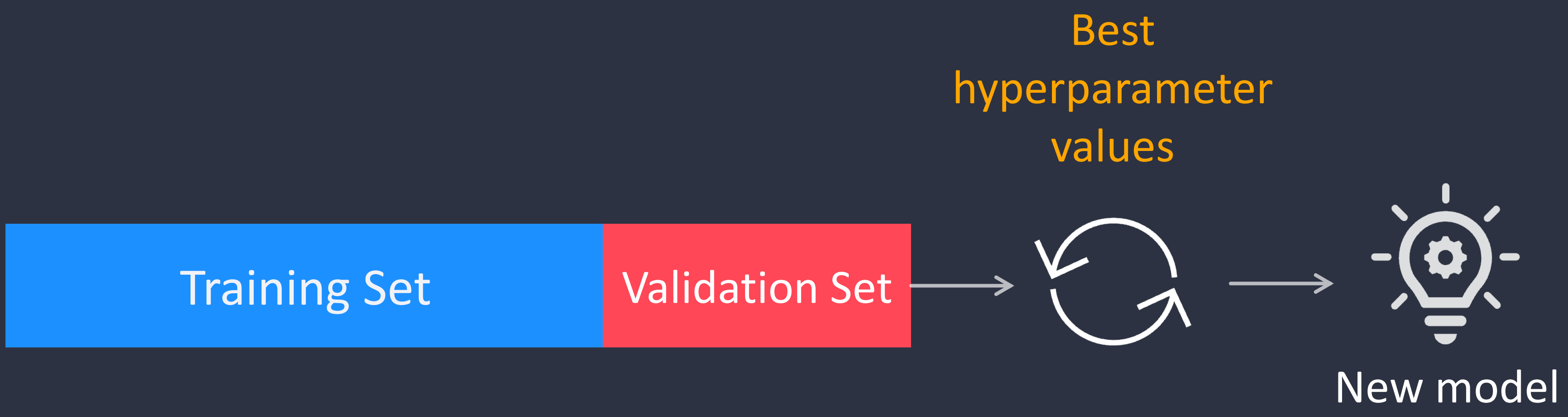
Hyperparameter Tuning Mechanism

e.g. kNN

n_neighbors	metric
3	“euclidean”
5	“manhattan”
9	“minkowski”

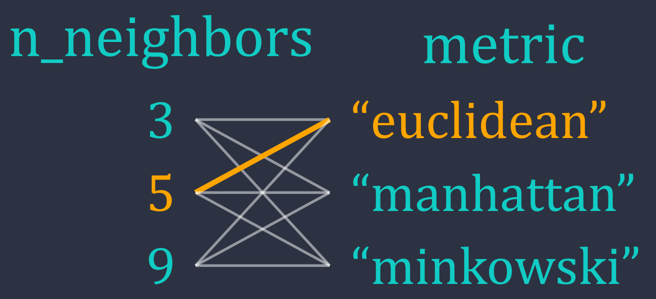
3-Way Holdout Method

4 Fit a new model with the best hyperparameter values on the combined training and validation set



Hyperparameter Tuning Mechanism

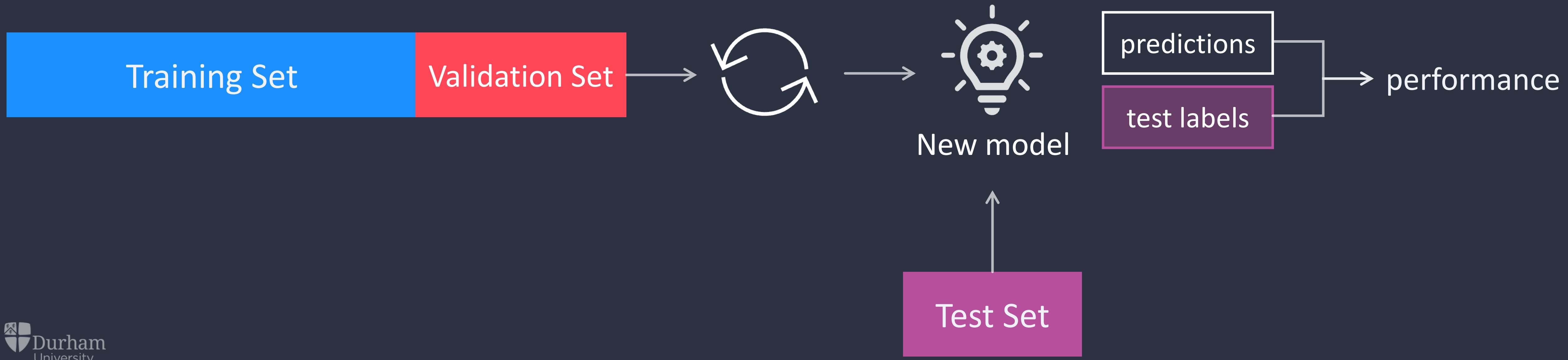
e.g. kNN



3-Way Holdout Method

4 Fit a new model with the best hyperparameter values on the combined training and validation set

5 Evaluate the model on test set

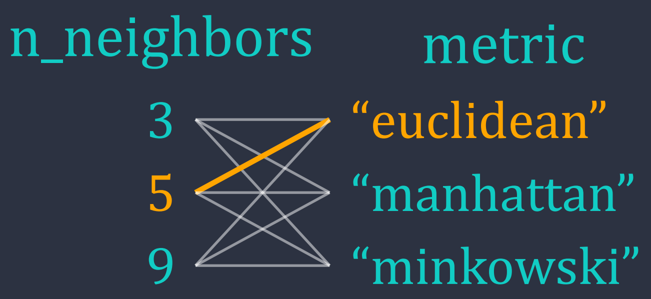


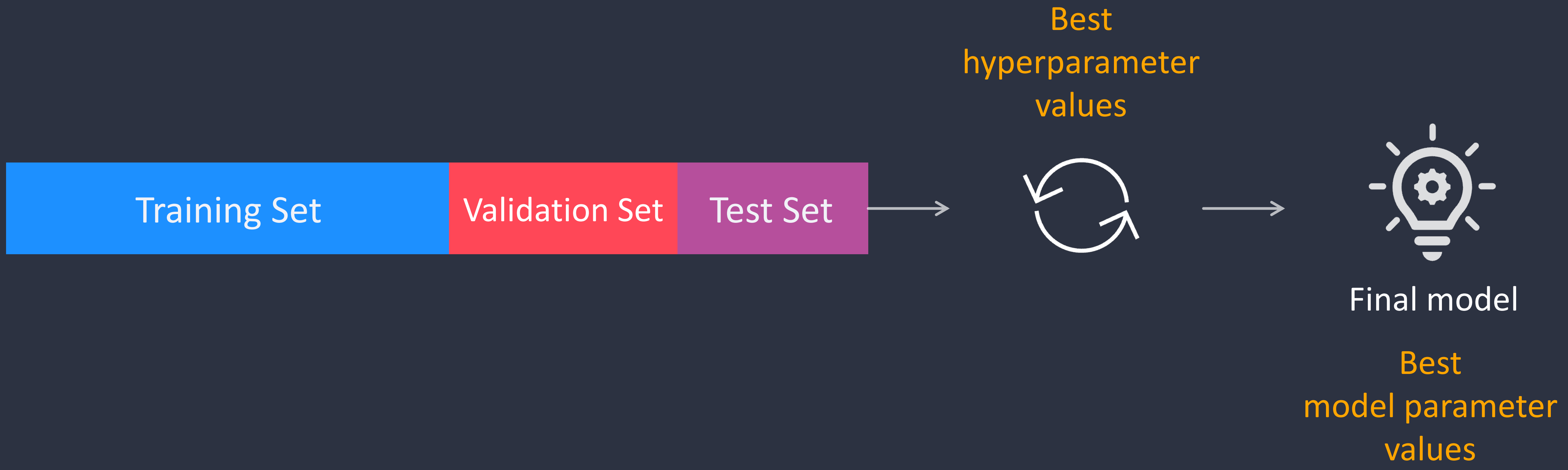
Hyperparameter Tuning Mechanism

3-Way Holdout Method

6 Fit the final model on the whole data set

e.g. kNN





e.g. kNN

n_neighbors	metric	algorithm
3	"euclidean"	'ball_tree'
5	"manhattan"	'kd_tree'
9	"minkowski"	'brute'
2	"chebyshev"	'auto'
4	"wminkowski"	
6	"seuclidean"	
7	"mahalanobis"	
8		
…		

Hyperparameter Tuning Algorithms

Hyperparameter Tuning Algorithm

To search for a set of hyperparameters that results in the best performance of a <u>model</u> on a <u>dataset</u>.

- Grid Search (Exhaustive Search)

  To try each combination of hyperparameters.

  Good for spot-checking combinations known to form well.

- Random Search

  To try random combination of hyperparameters.

  Good for exploring combinations difficult to have guessed intuitively.

Hyperparameter Tuning Algorithm

- Grid Search GridSearchCV
- Random Search RandomizedSearchCV

```
...
# define model
model = svm.SVC()
# define search space
space = dict()
...
# define search
search = GridSearchCV(model, space)
```

```
...
# define evaluation
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
search = GridSearchCV(..., cv=cv)
```

RepeatedKFold for regression tasks

```
...
# define search
search = GridSearchCV(..., scoring='accuracy')
```

'accuracy' for classification
'neg_mean_absolute_error' for regression

```
...
# define search
search = GridSearchCV(..., n_jobs=5)
...
# execute search
result = search.fit(X,y)
...
# summarise result
Print(result.best_score_)
Print(result.best_params_)
```

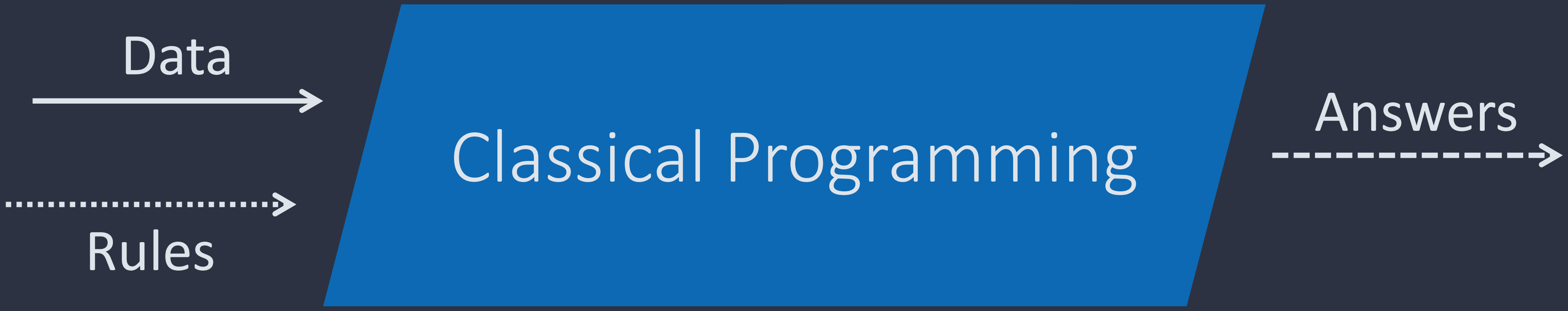
Hyperparameter Tuning Algorithm

- Grid Search GridSearchCV
- Random Search RandomizedSearchCV

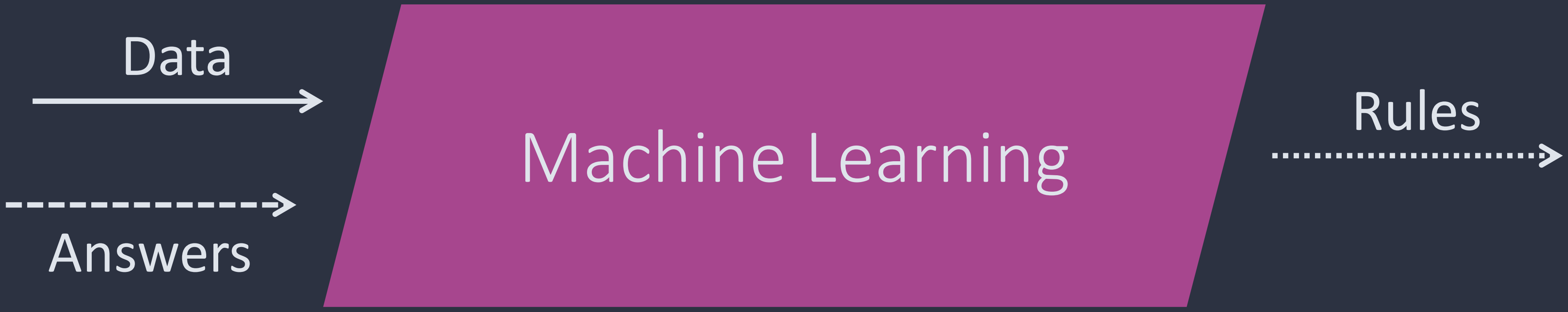
```
from sklearn import svm, datasets
From sklearn.model_selection import GridSearchCV

# load data
iris =datasets.load_iris()

#define model, hyperparameters to tune for, and the search space
hyperparameters = {‘kernel’: {‘linear’, ‘rbf’}, ‘C’: {1,5}}
svc = svm.SVC()

clf = GridSearchCV(svc, hyperparameters, cv=5)
clf = fit(iris.data, iris.target)

clf.cv_results_
clf.best_params_            {‘C’:1, ‘kernel’: ‘linear’}
```

3. Sub-module Wrap-up

What's Machine Learning



The programmer learns the rules from observing the data



The machine learns the rules from observing the data

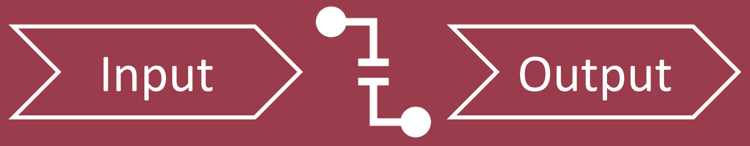
Machine Learning : a Definition

“A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.”

-- Tom Mitchell, 1997

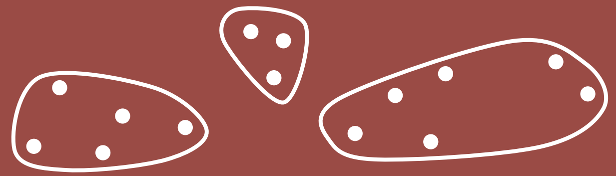
Three types of Machine Learning

Supervised Learning



- To learn the mapping (rules) between inputs and outputs
- Labelled data is provided of past input & output pairs during the learning process to train the model how it should behave for previously unseen data.

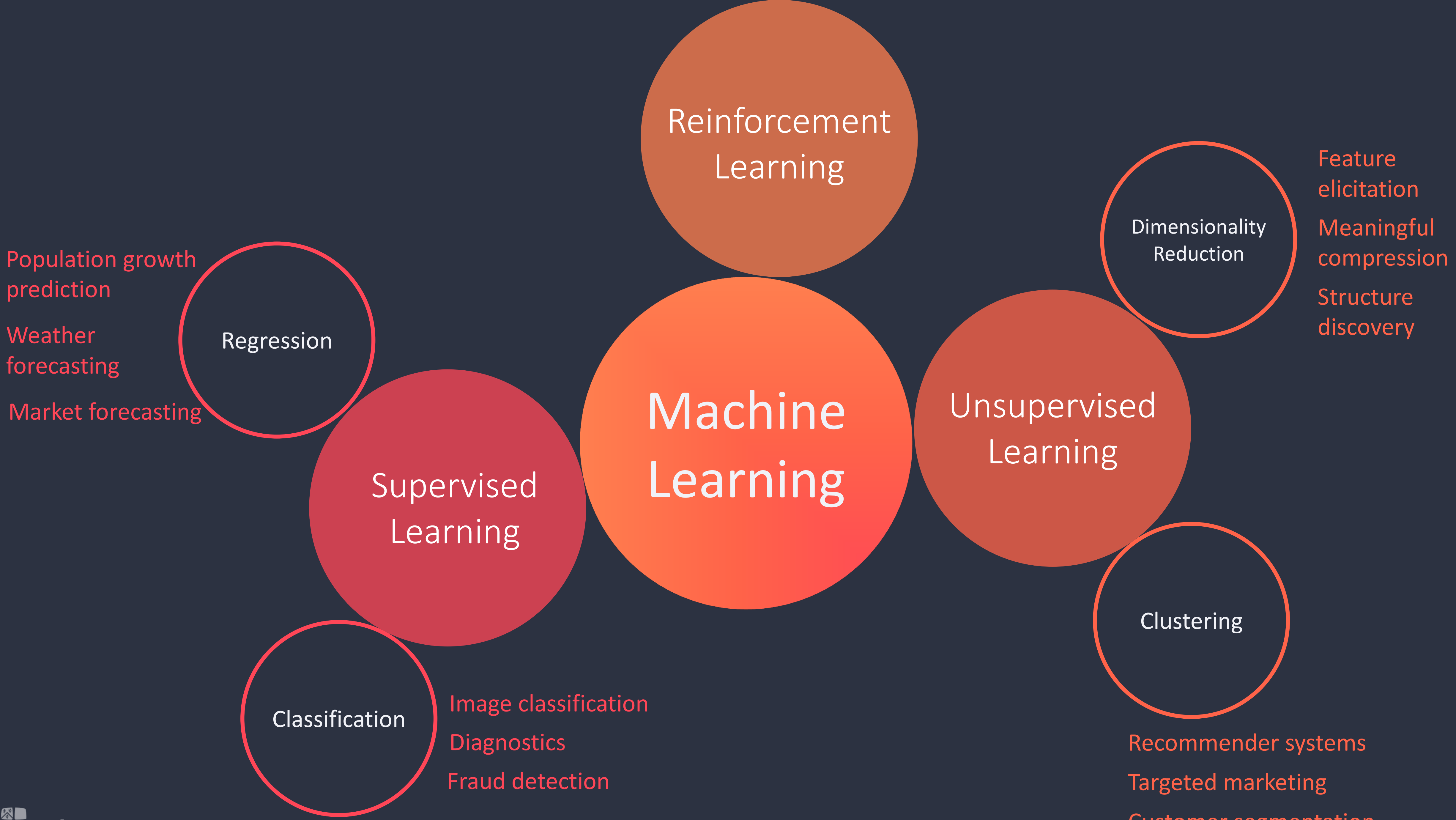
Unsupervised Learning



- To learn hidden pattern (rules) from a set of inputs (no output).
- Unlabelled data is provided of past input (not a input & output pair) during the learning process.
- Instances in the same group are more similar to each other than to those in other groups.

Reinforcement Learning

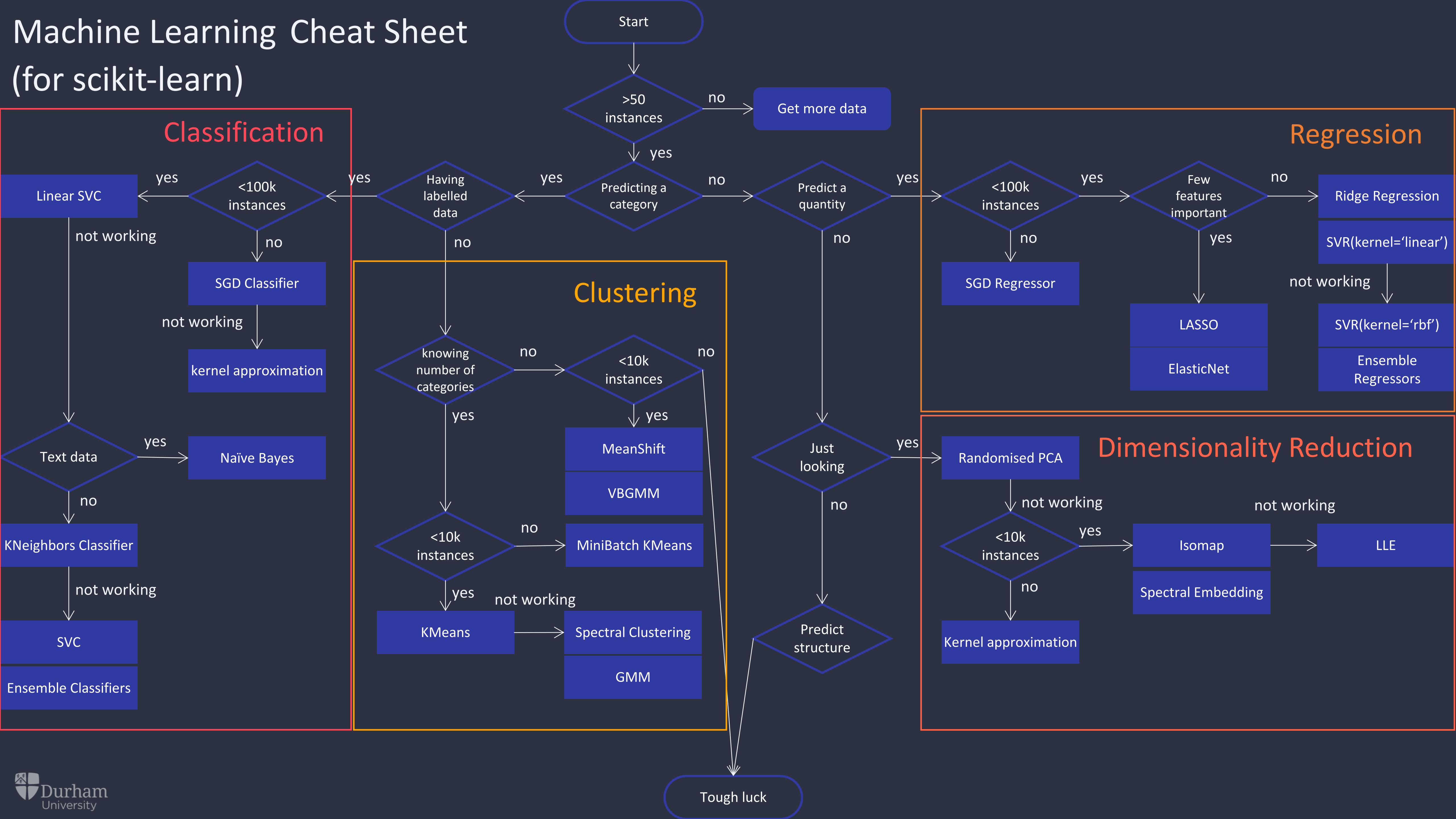
- Occasional positive & negative feedback to reinforce behaviours.
- Good behaviours are rewarded with treat → more common. Bad ones are punished → less common.
- Balancing between exploration (of uncharted territory) and exploitation (of current knowledge)



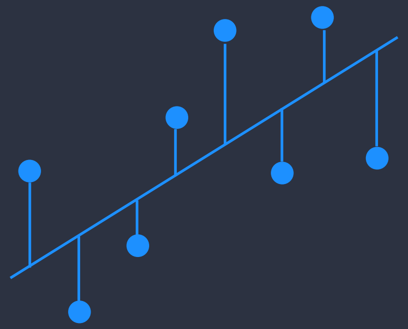
Durham University

Machine Learning Workflow

1. Problem Framing
2. Data Preparation
3. Algorithm Selection
4. Model Training
5. Model Testing
6. Hyperparameter Tuning
7. Inference / Prediction

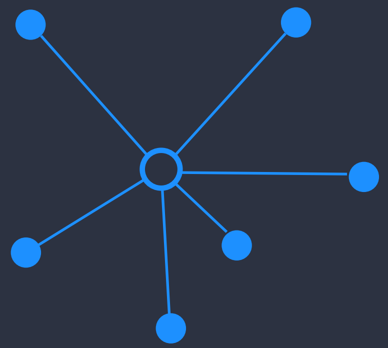

Machine Learning Cheat Sheet
(for scikit-learn)
Start
>50 instances
no
Get more data
yes
Predicting a category
yes
Having labelled data
yes
Classification
<100k instances
yes
Linear SVC
not working
no
SGD Classifier
not working
kernel approximation
Text data
yes
Naïve Bayes
no
KNeighbors Classifier
not working
SVC
Ensemble Classifiers
no
Clustering
knowing number of categories
no
<10k instances
no
yes
MeanShift
VBGMM
yes
<10k instances
no
MiniBatch KMeans
yes
not working
KMeans
Spectral Clustering
GMM
Tough luck
no
Predict a quantity
yes
Regression
<100k instances
yes
no
SGD Regressor
Few features important
no
Ridge Regression
SVR(kernel='linear')
not working
SVR(kernel='rbf')
Ensemble Regressors
yes
LASSO
ElasticNet
no
Just looking
yes
Dimensionality Reduction
Randomised PCA
not working
<10k instances
yes
Isomap
Spectral Embedding
not working
LLE
no
Kernel approximation
no
Predict structure

Machine Learning Algorithms

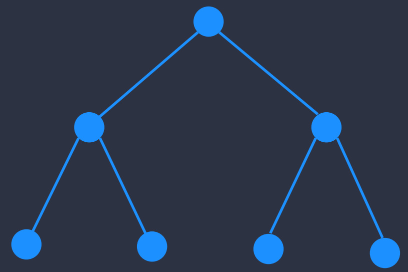


Regression

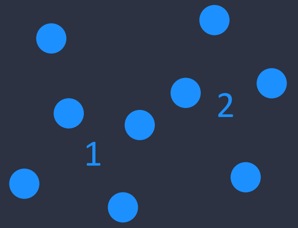
Regularisation



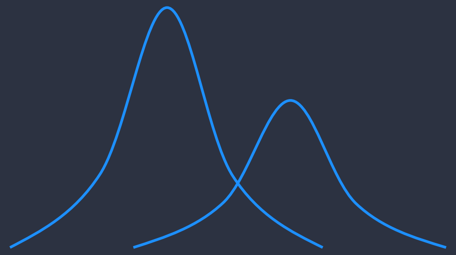
Instance-based



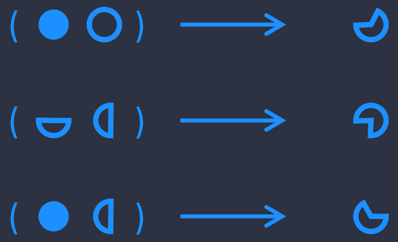
Tree-based



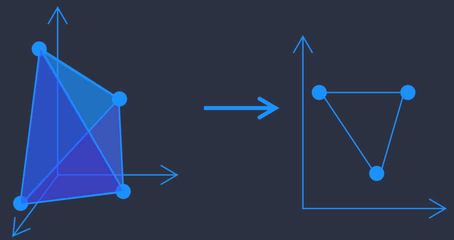
Clustering



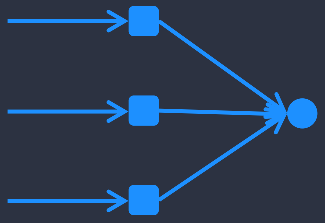
Bayesian



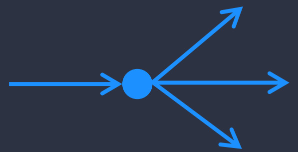
Associated Rule Learning



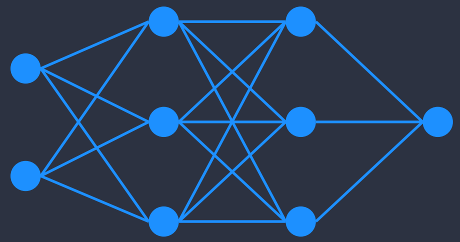
Dimensionality Reduction



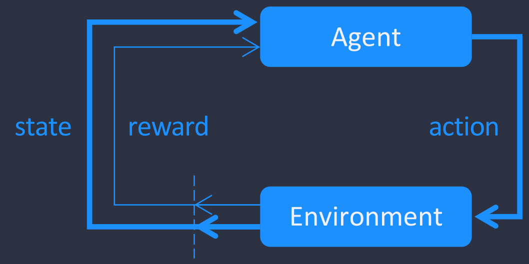
Ensemble



Neural Network



Deep Learning



Reinforcement Learning

Decomposition of Machine Learning Learning algorithms

Representation

Evaluation — Optimisation

Pedro Domingos, a CS professor at the University of Washington

https://homes.cs.washington.edu/~pedrod/papers/cacm12.pdf

Content

Philosophy behind machine learning

Fundamental concepts in machine learning

Machine learning workflow

Defining machine learning tasks

Data preparation for machine learning

Model selection and evaluation

Implement machine learning algorithms using Python and scikit-learn

Interpreting results

Learning Outcomes

Understand key principles of ML for use in managing dataset and building models

Understand differences between supervised learning and unsupervised learning

Understand the math behind ML models and algorithms

Be able to select and implement appropriate learning algorithms for real-life problems

Be able to train, optimise, evaluate, and compare ML models

Be able to scientifically report the result of machine learning projects

What’s next?

Machine Learning

Advancement	COMP2261 Artificial Intelligence – Bias in AI
	COMP3547 Deep Learning and Reinforcement Learning
Applications	COMP3517 Computational Modelling in the Humanities and Social Sciences
	COMP3527 Computer Vision
	COMP3647 Human-AI Interaction Design
	COMP4157 Learning Analytics
	COMP4167 Natural Language Processing
	…

Good luck!