

Computational Analysis of Big Data

Week 4

Machine Learning 1

Canonical example

Canonical example



Canonical example



[]

Dog



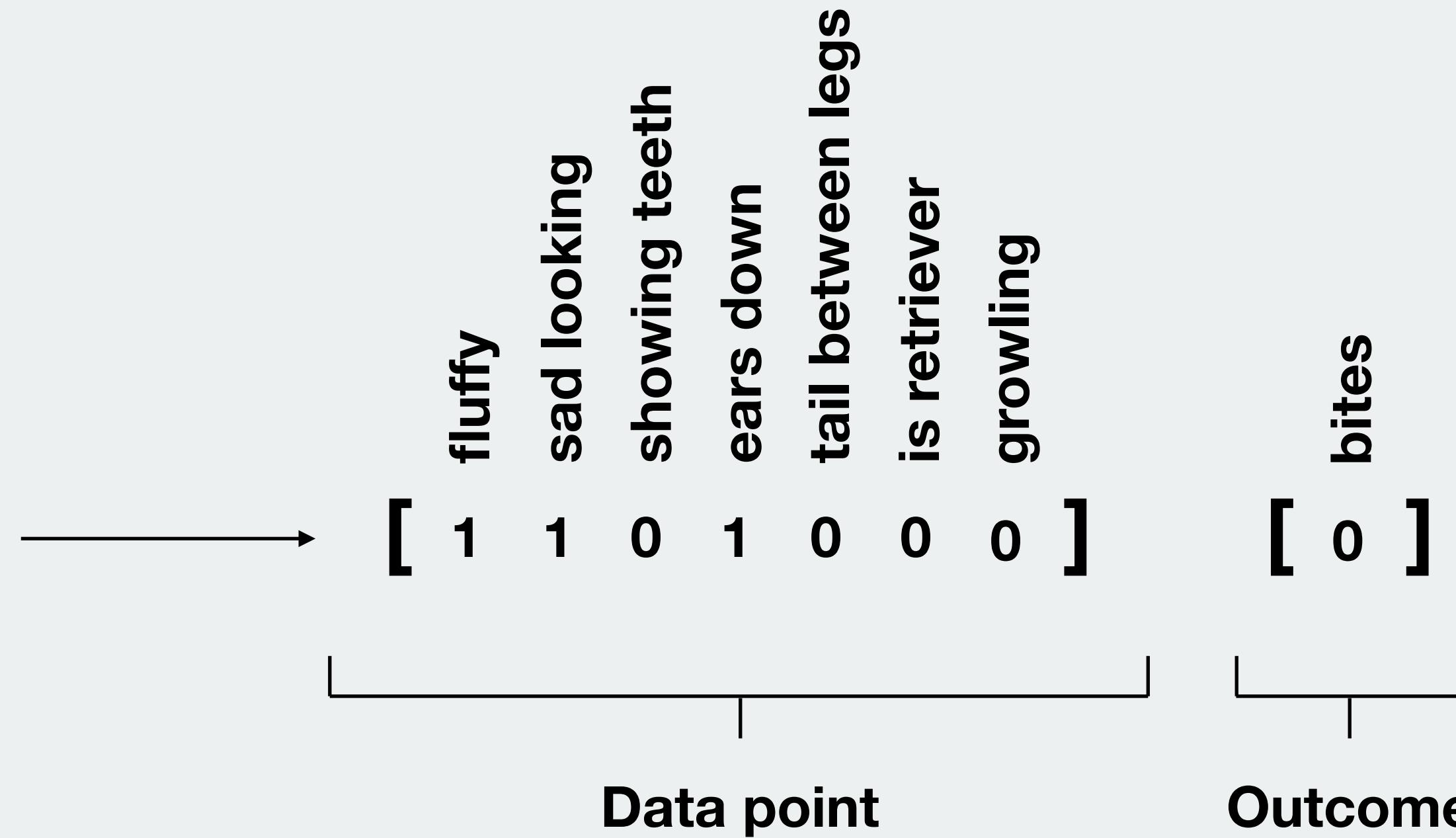
[1 1 0 1 0 0 0]

[]

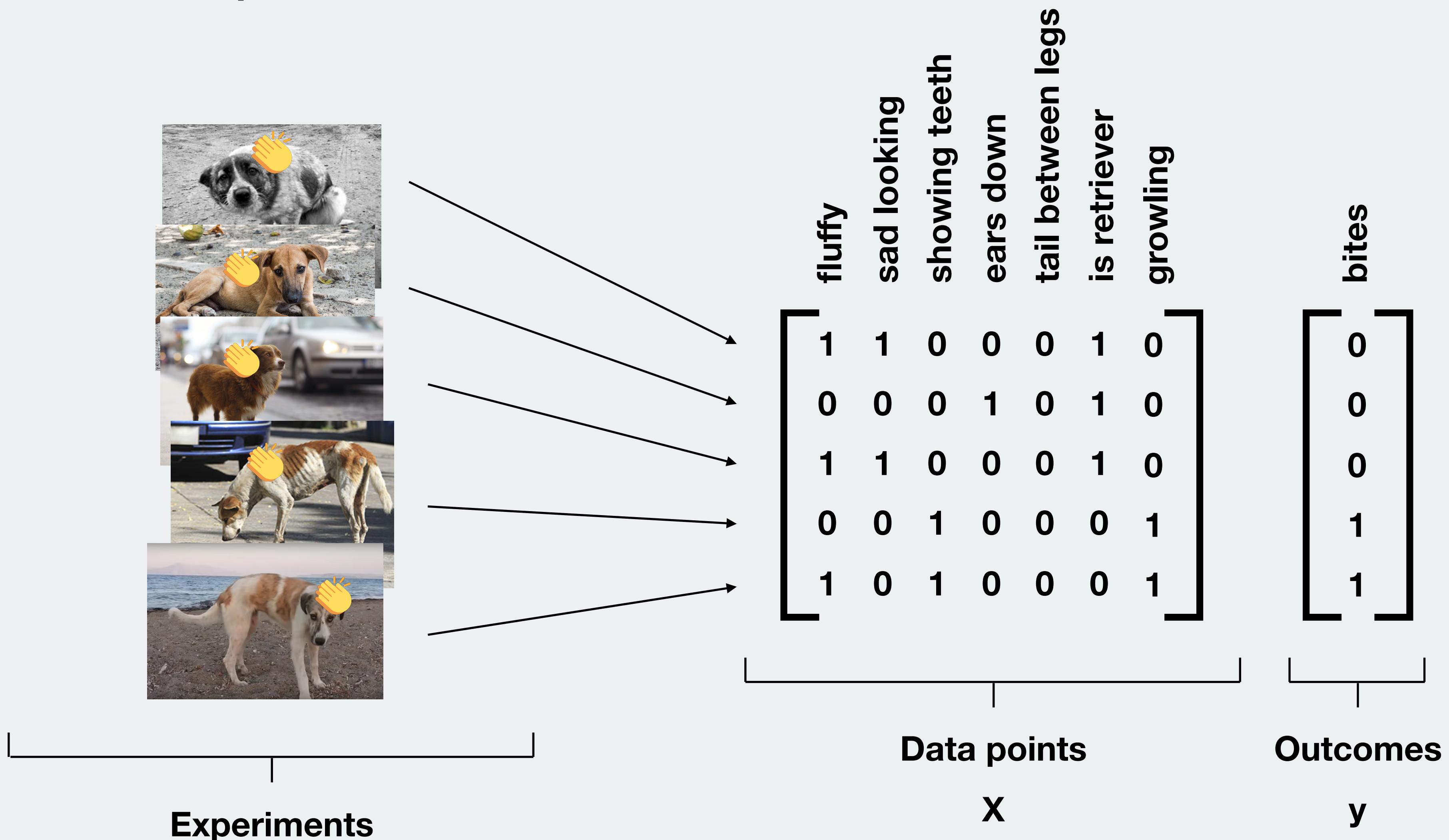
Data point

fluffy
sad looking
showing teeth
ears down
tail between legs
is retriever
growling

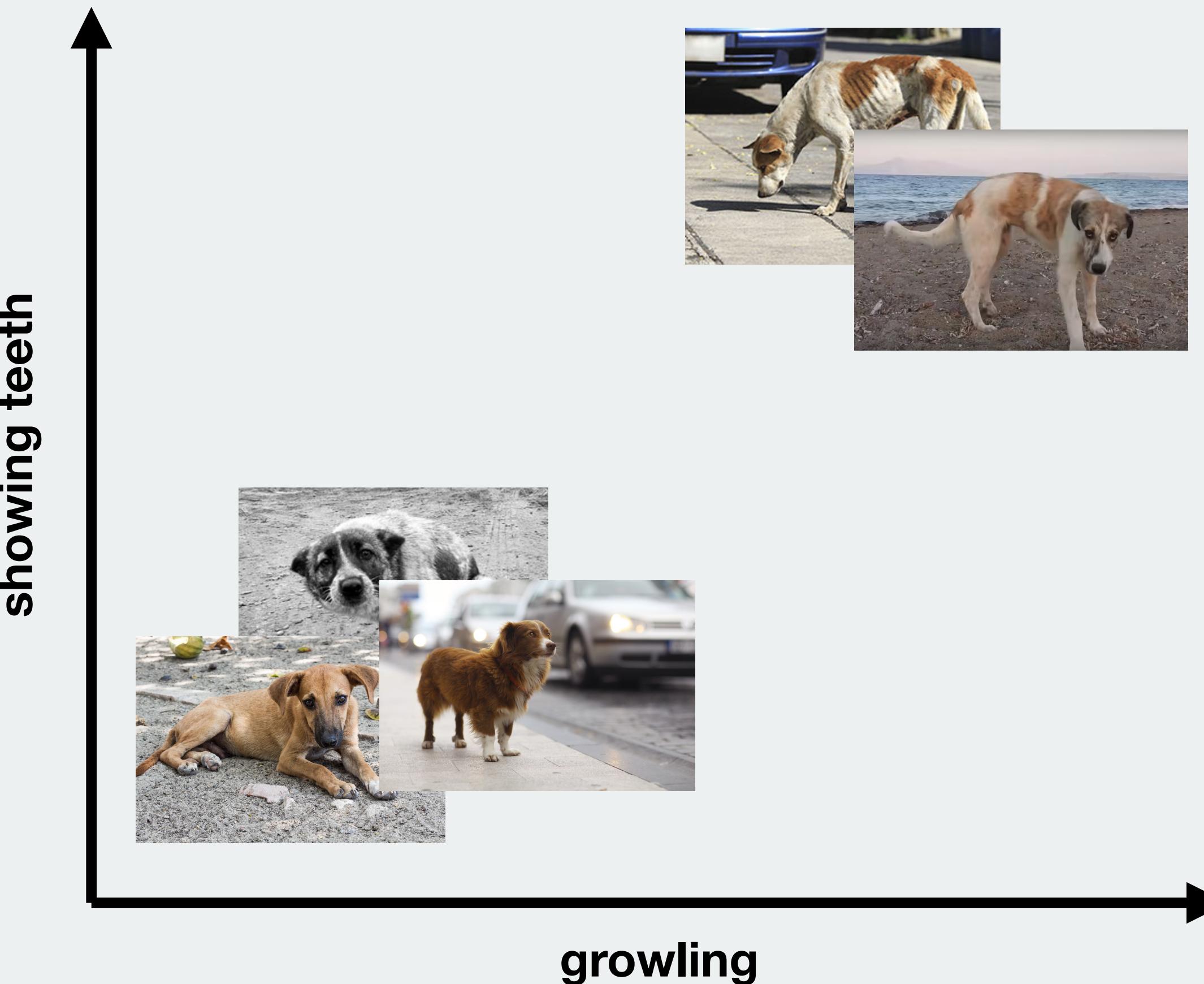
Canonical example



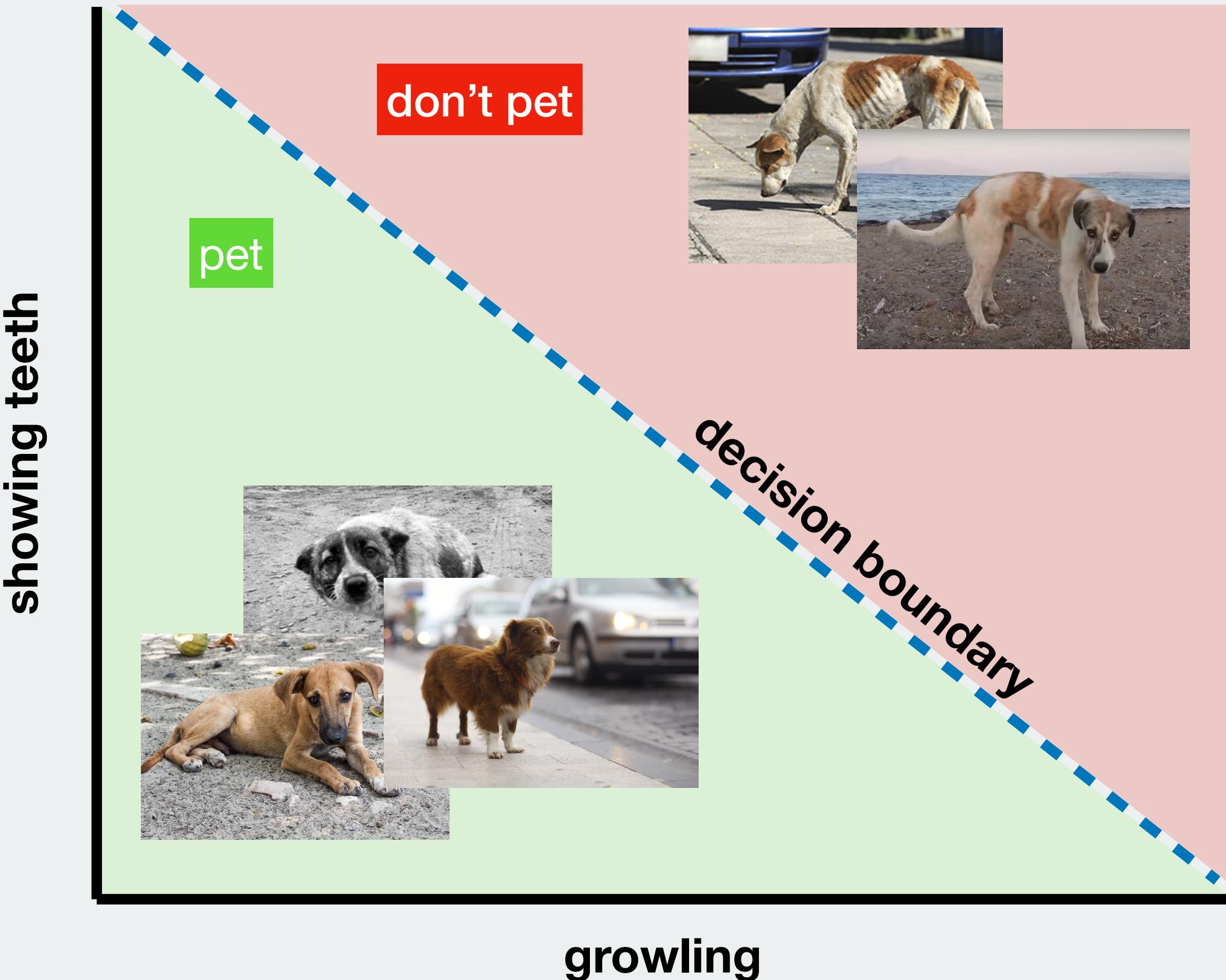
Canonical example



Canonical example



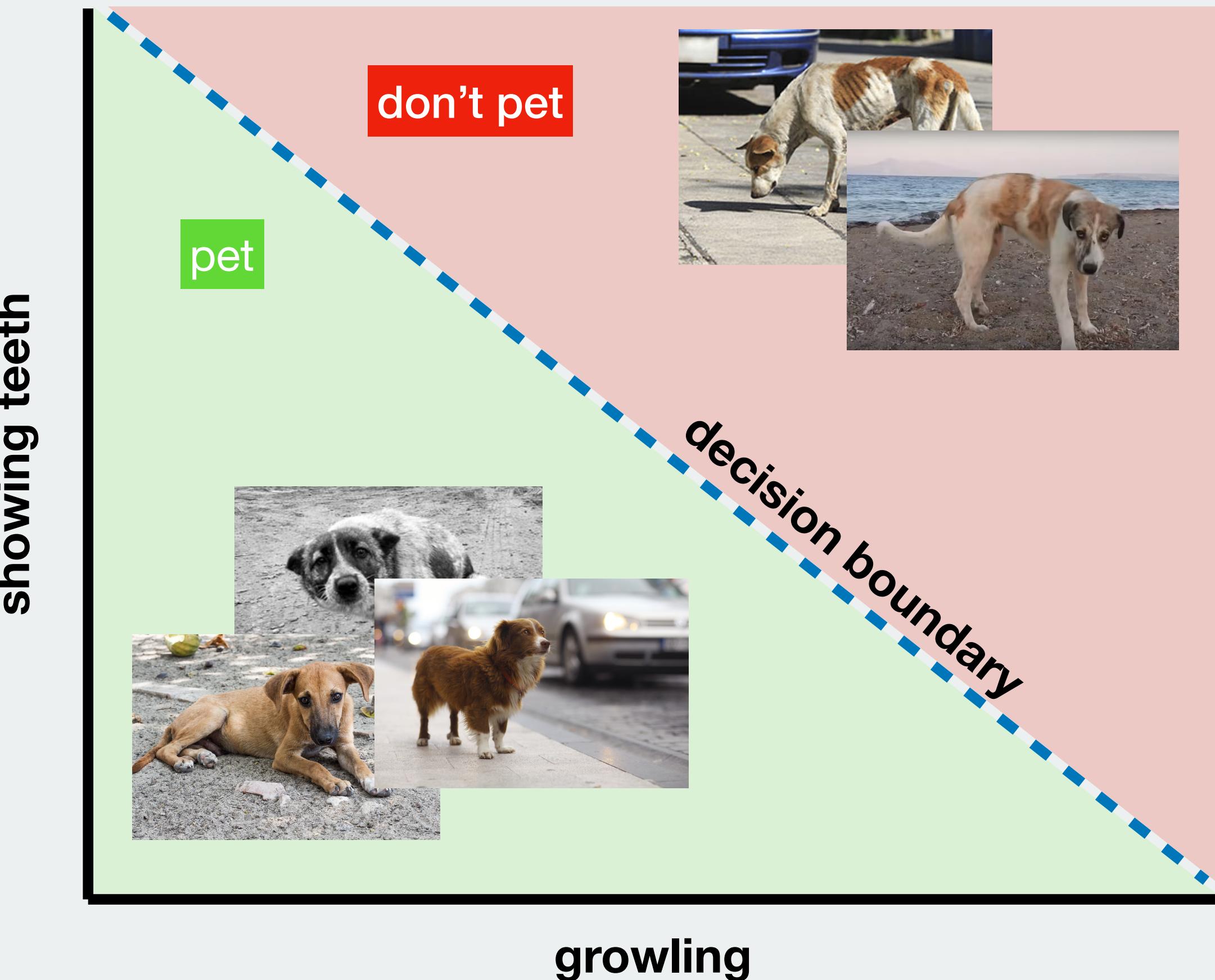
Canonical example



Canonical example

Supervised Machine Learning

When the input data has outcome labels



Types of machine learning

Types of machine learning

Supervised

When you **have** outcomes



Unsupervised

When you **don't have** outcomes

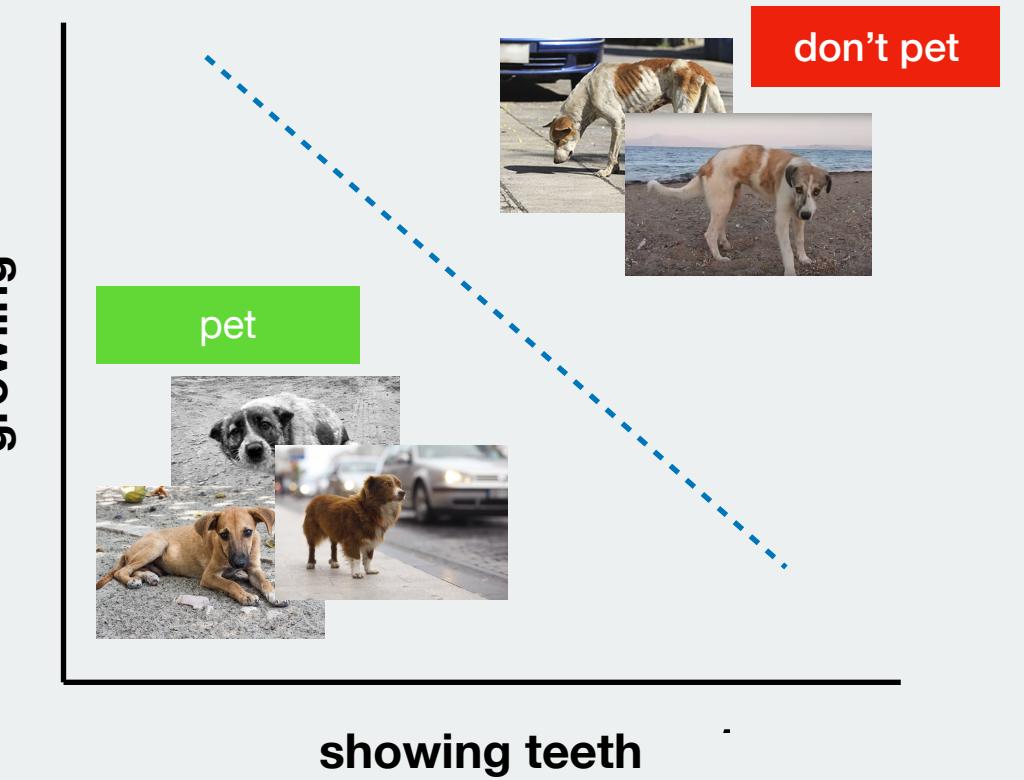


Types of machine learning

Supervised

When you **have** outcomes 

- Classification



Unsupervised

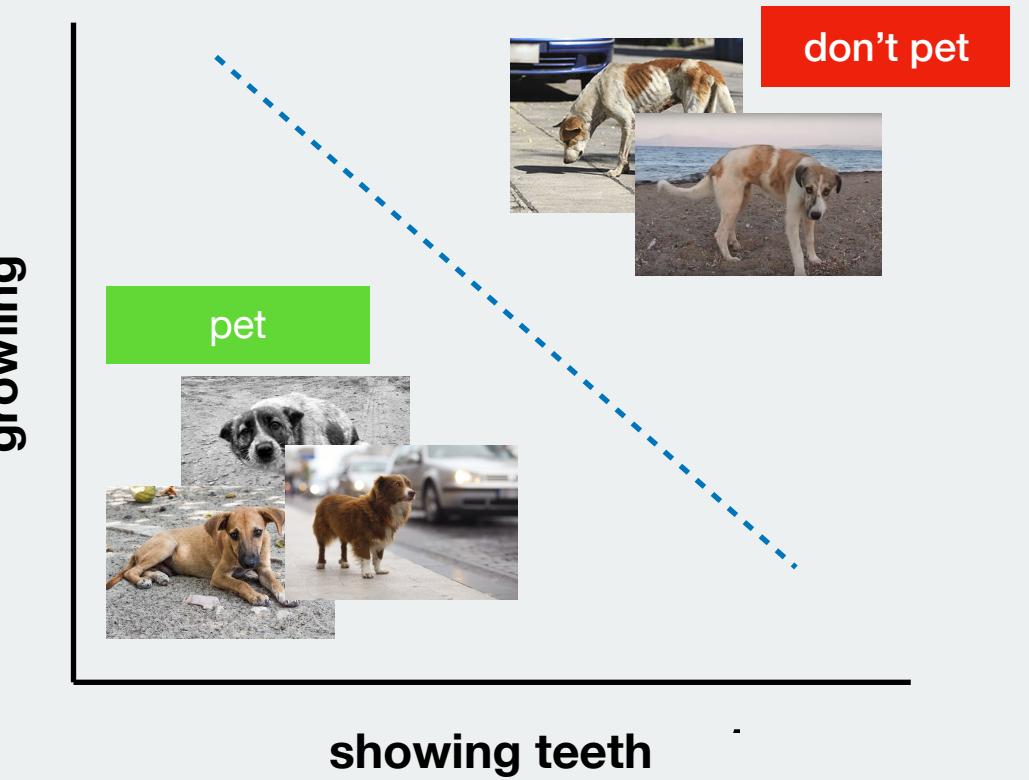
When you **don't have** outcomes 

Types of machine learning

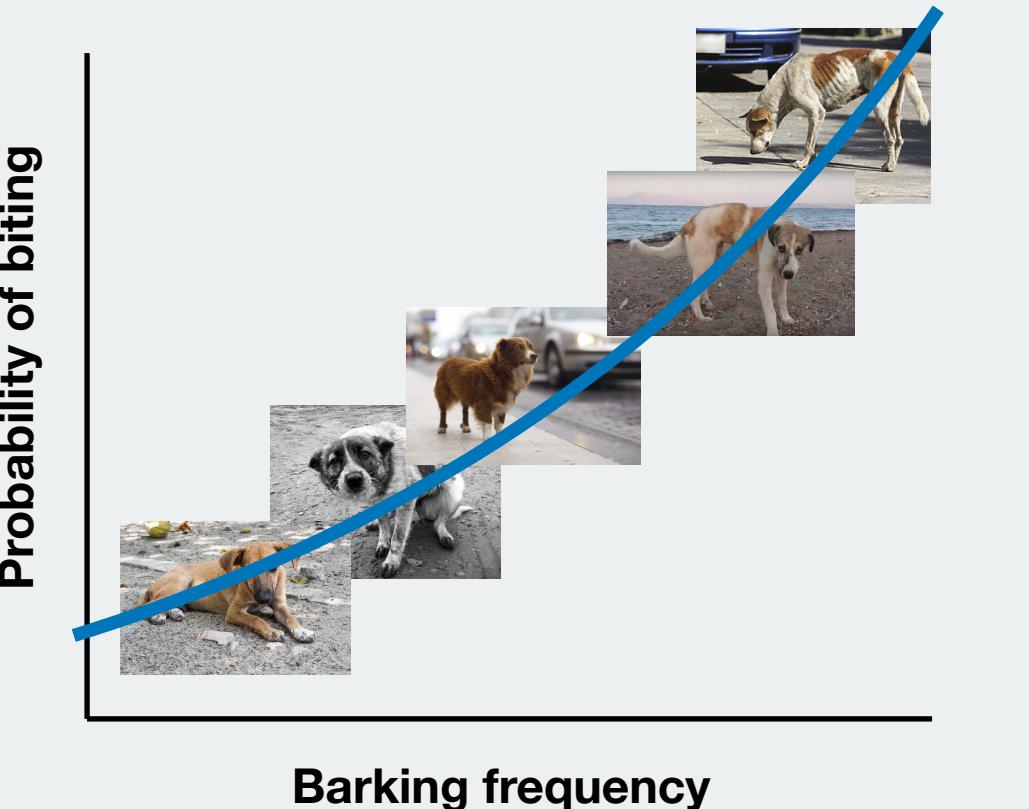
Supervised

When you **have** outcomes 

- Classification



- Regression



Unsupervised

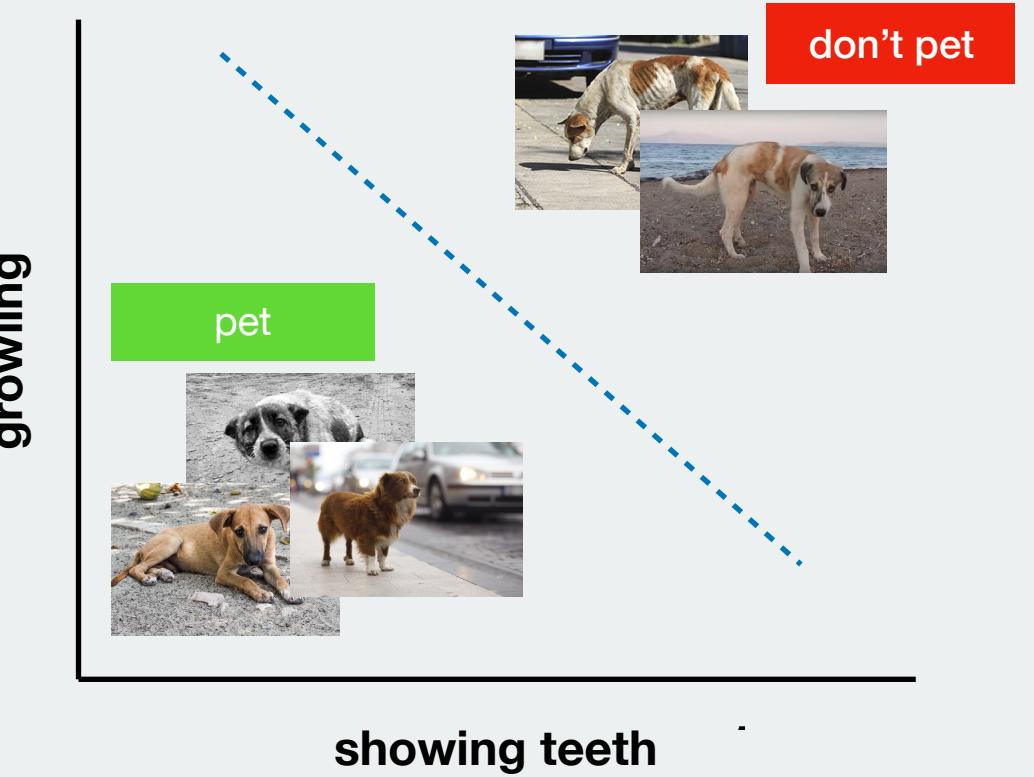
When you **don't have** outcomes 

Types of machine learning

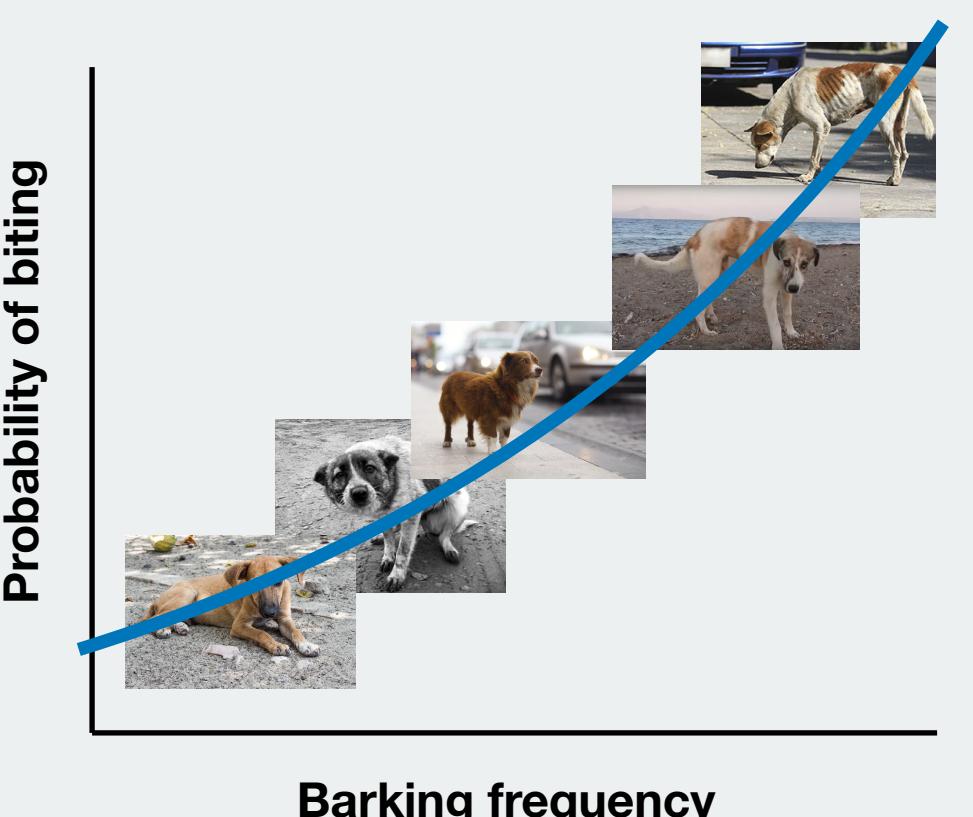
Supervised

When you **have** outcomes 

- Classification



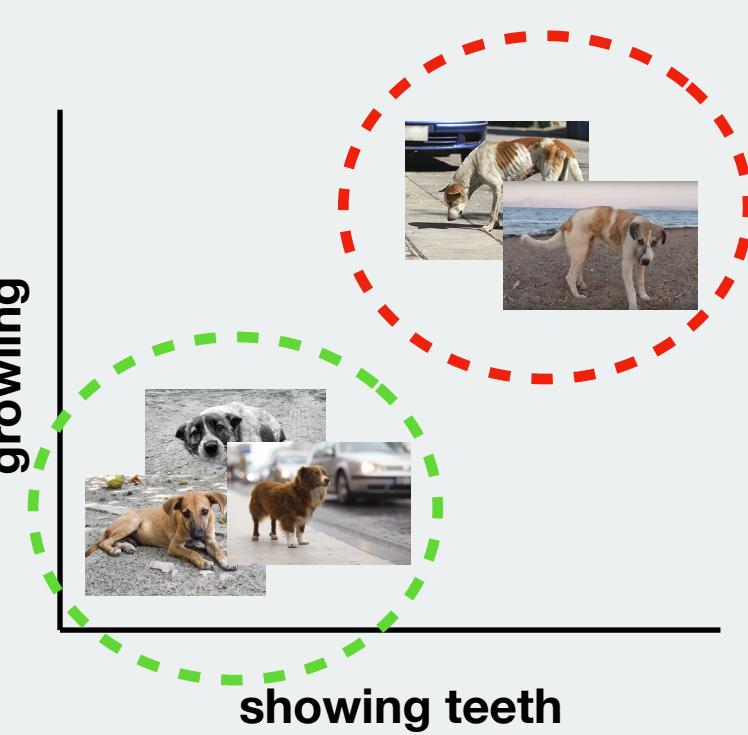
- Regression



Unsupervised

When you **don't have** outcomes 

- Clustering

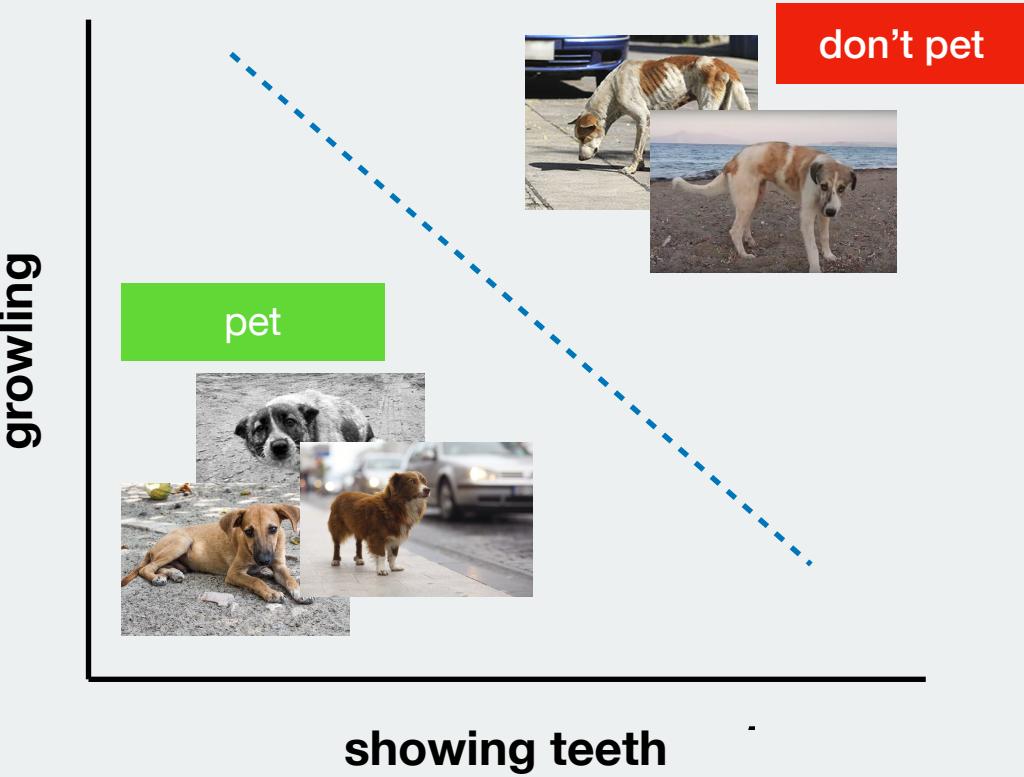


Types of machine learning

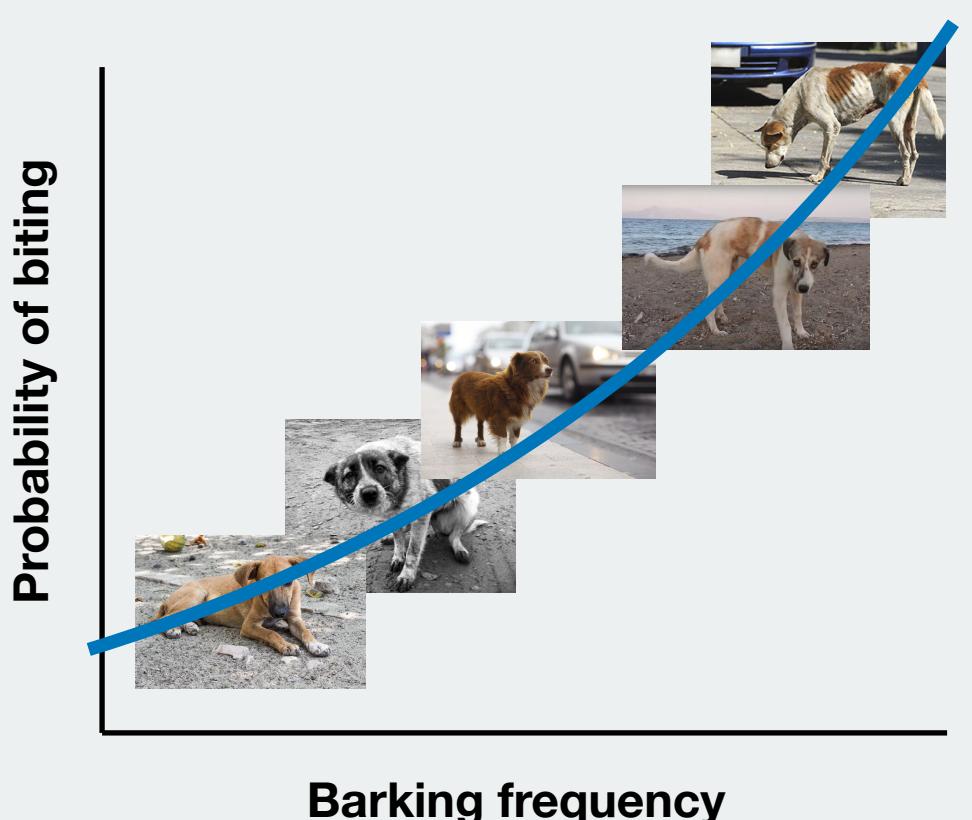
Supervised

When you **have** outcomes 

- Classification



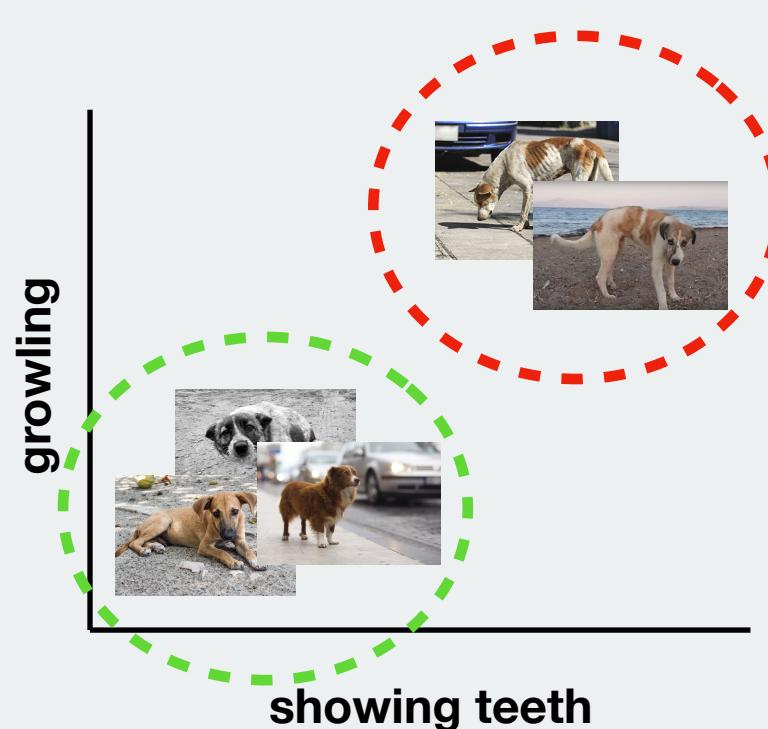
- Regression



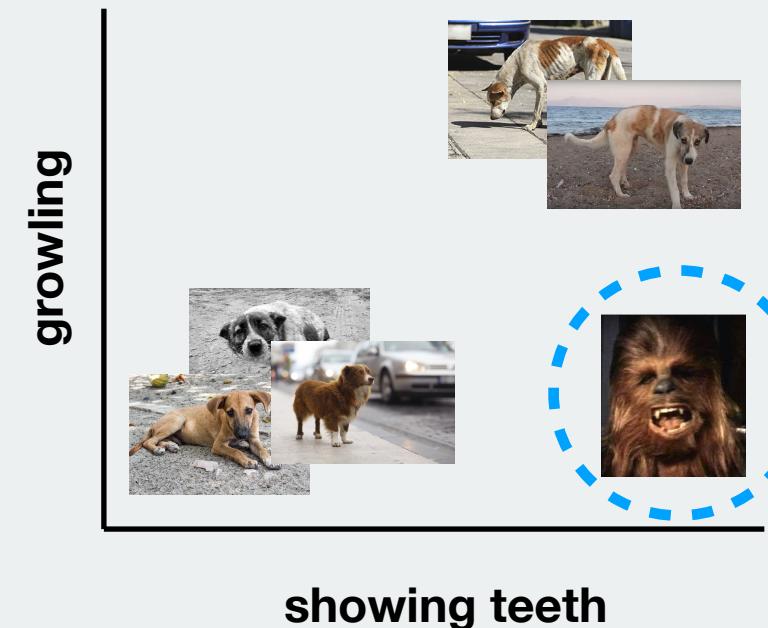
Unsupervised

When you **don't have** outcomes 

- Clustering



- Outlier detection

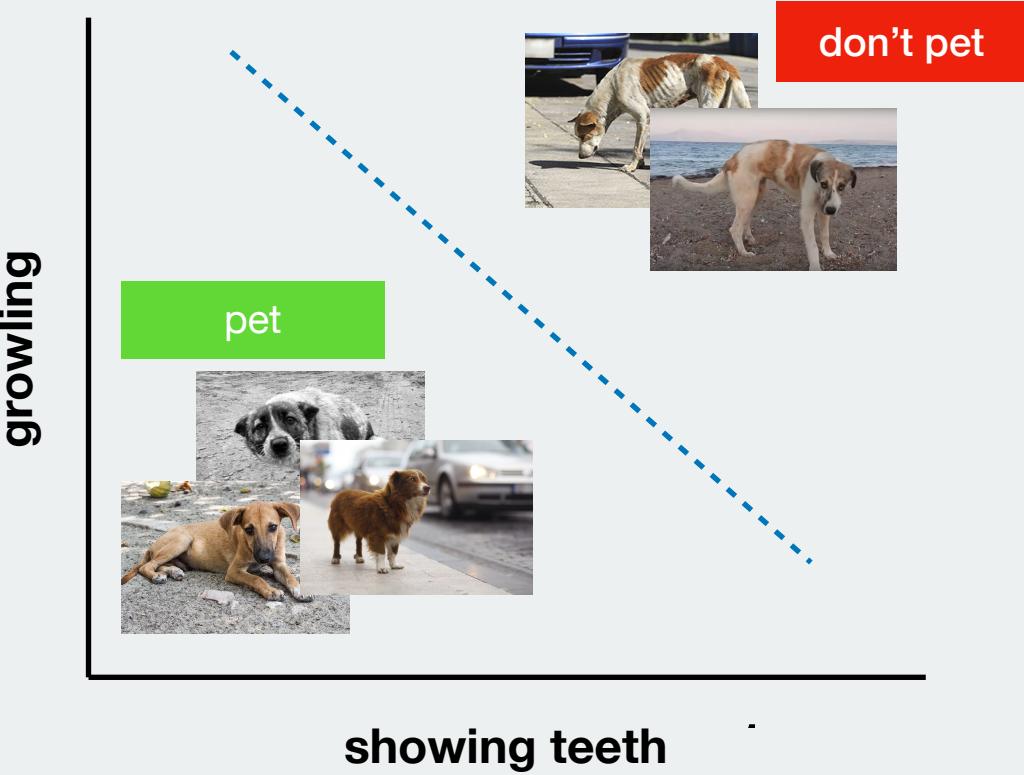


Types of machine learning

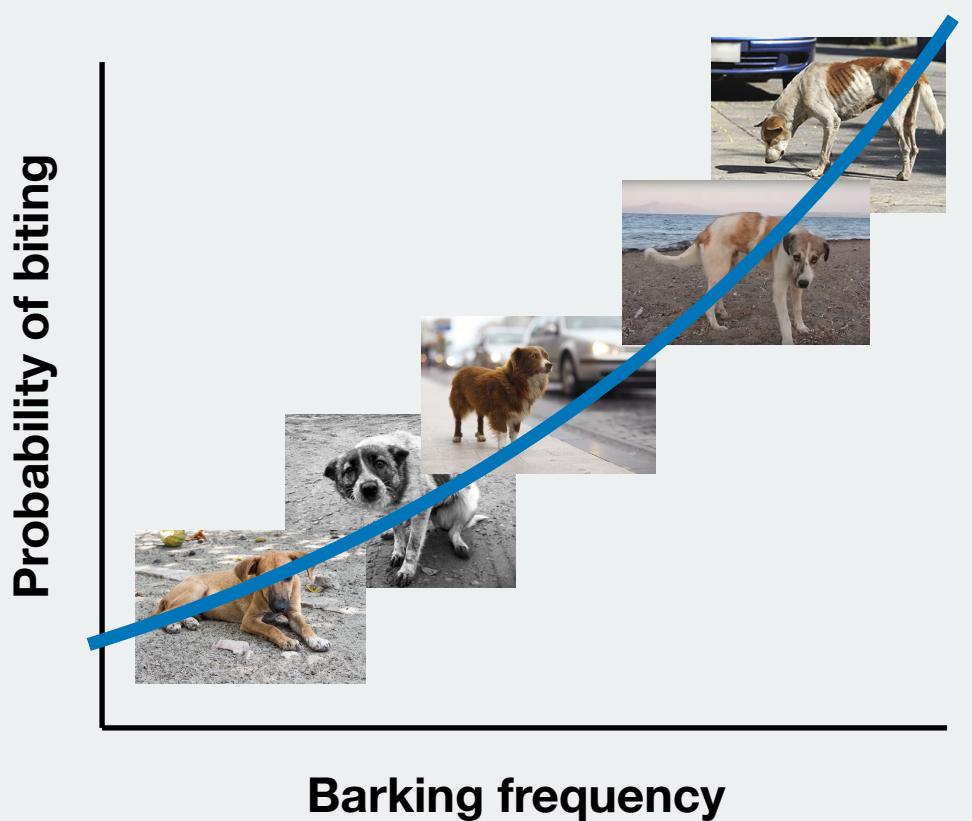
Supervised

When you **have** outcomes 

- Classification



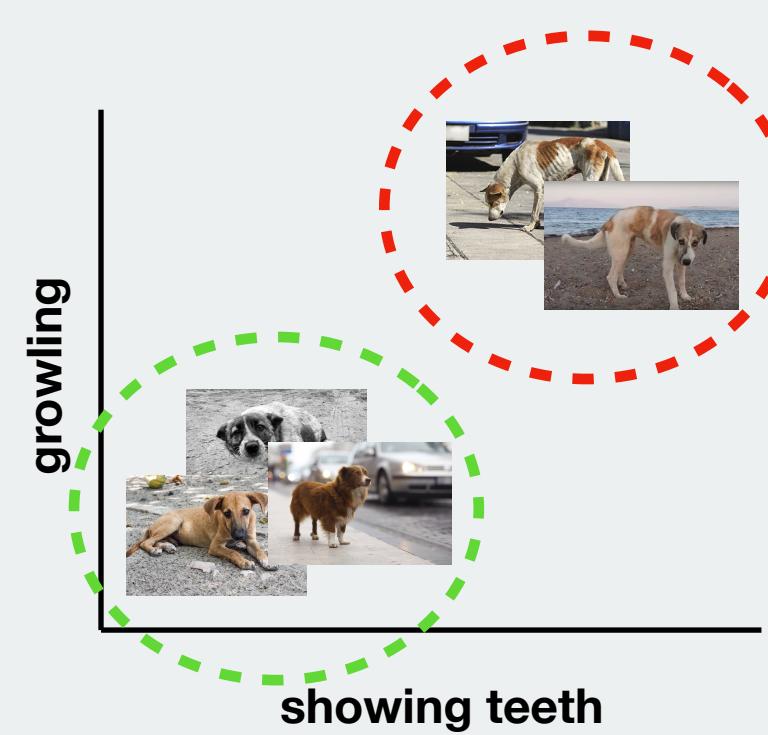
- Regression



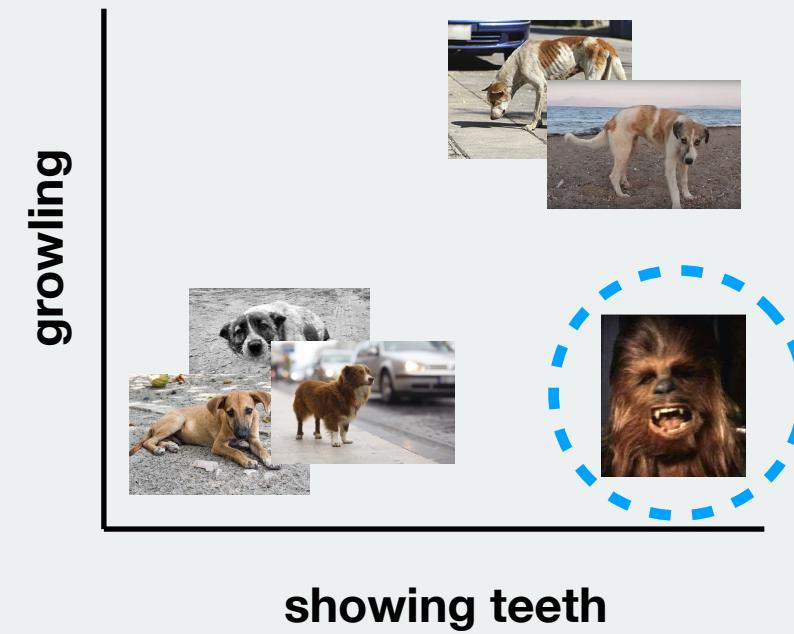
Unsupervised

When you **don't have** outcomes 

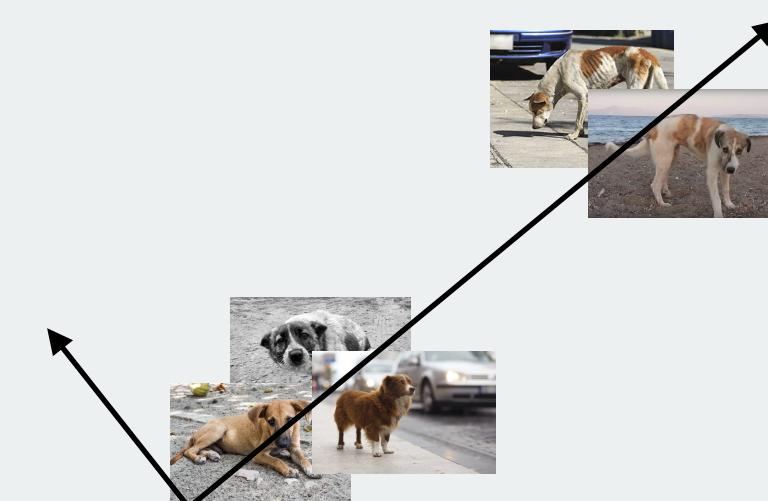
- Clustering



- Outlier detection



- Latent variable analysis

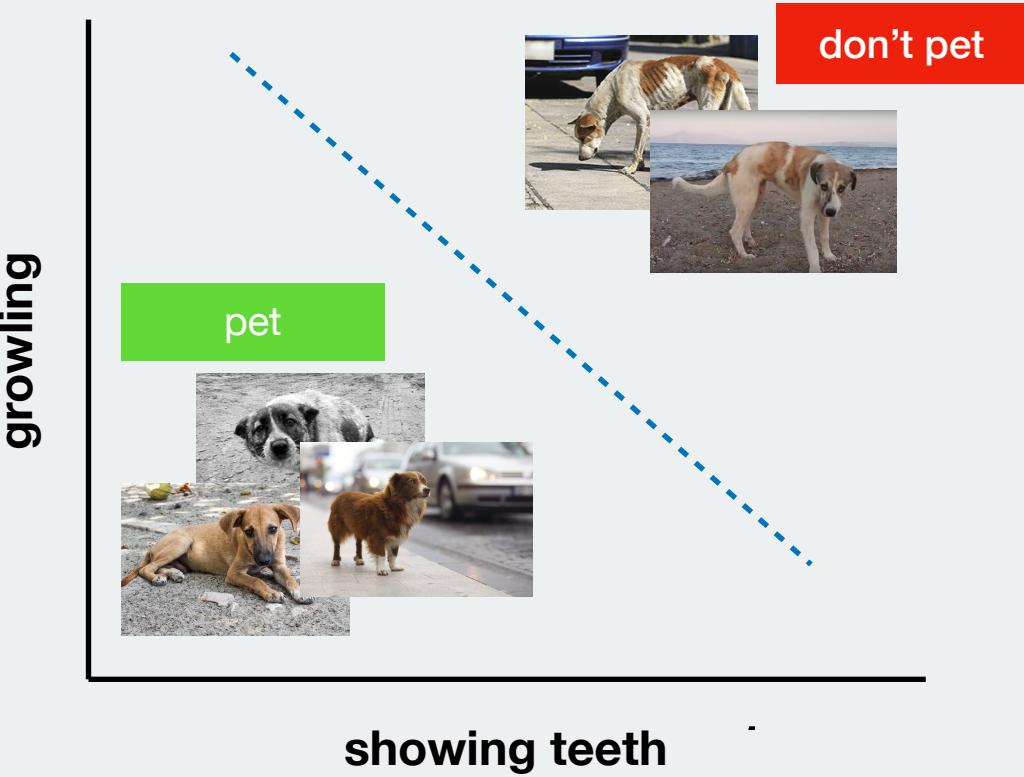


Types of machine learning

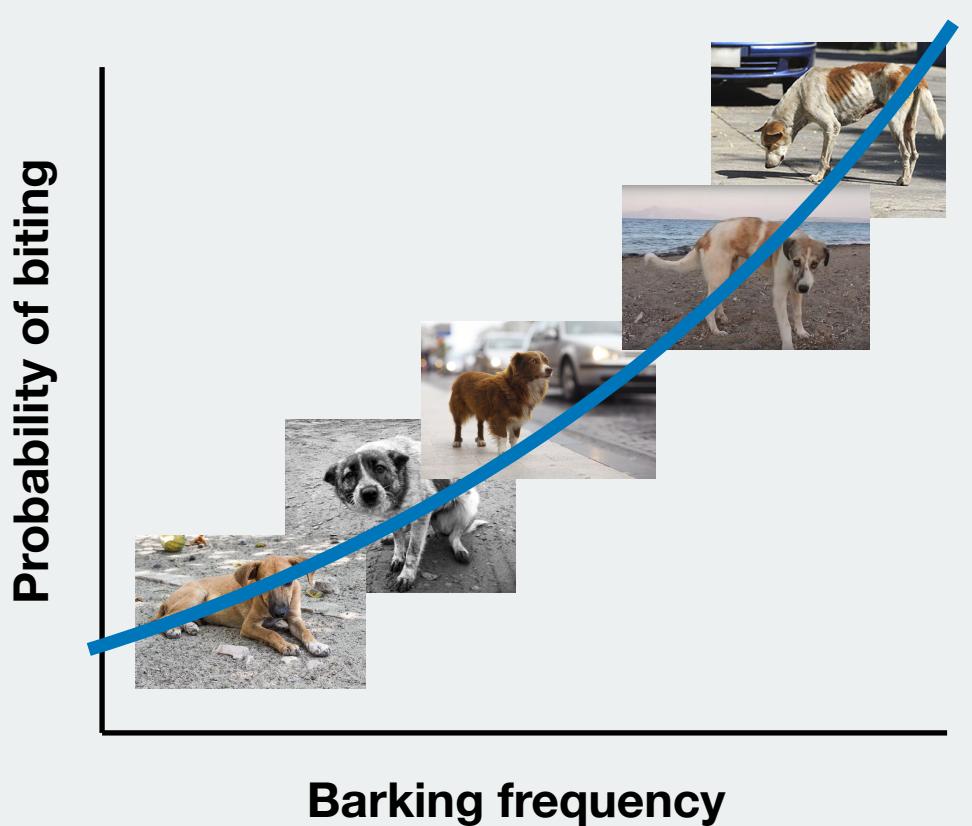
Supervised

When you **have** outcomes 

- Classification



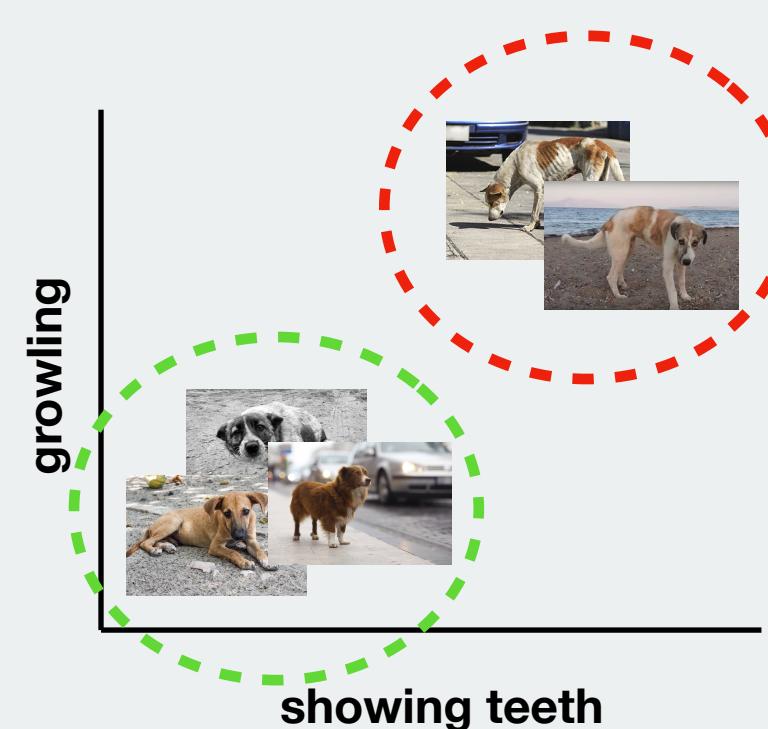
- Regression



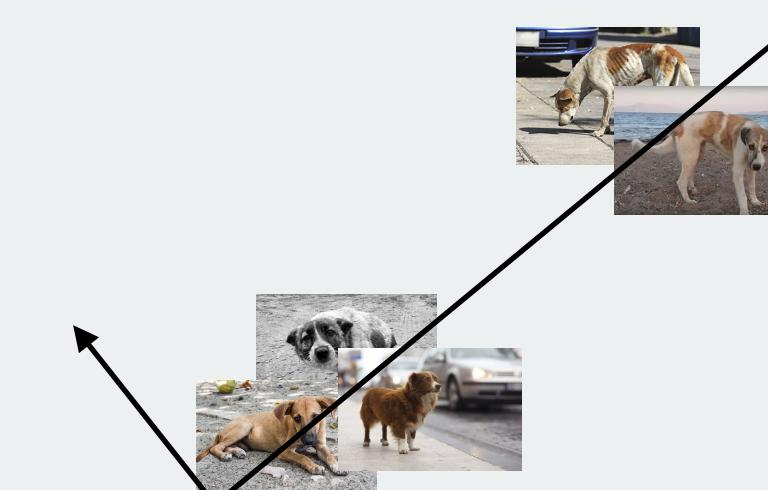
Unsupervised

When you **don't have** outcomes 

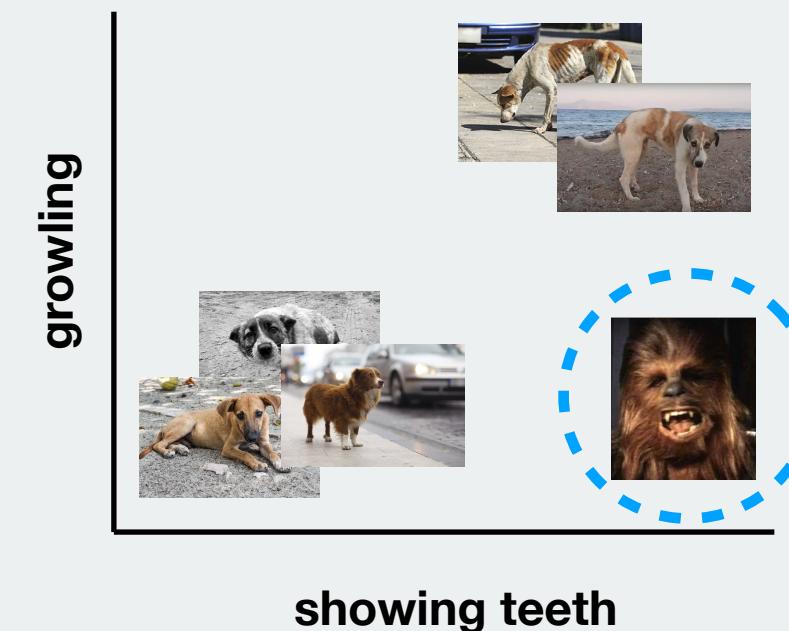
- Clustering



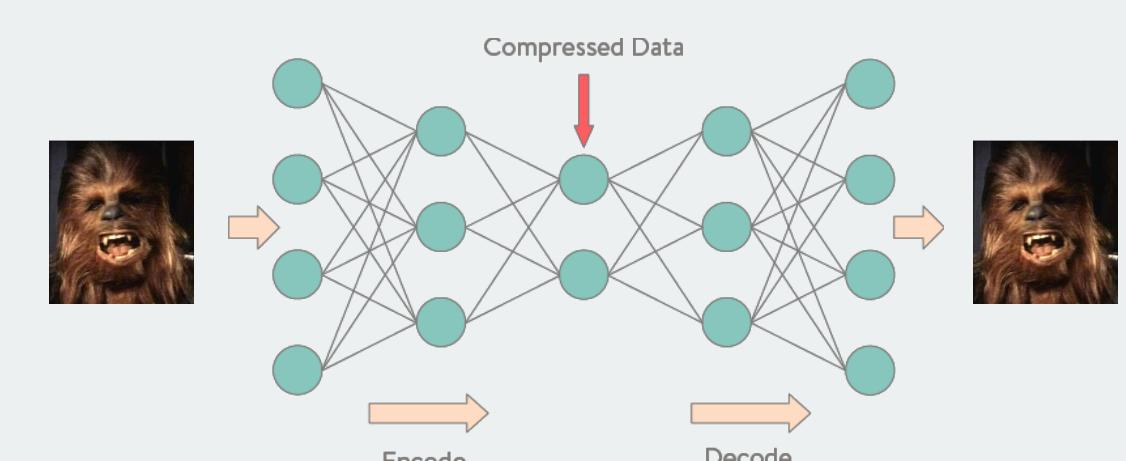
- Latent variable analysis



- Outlier detection

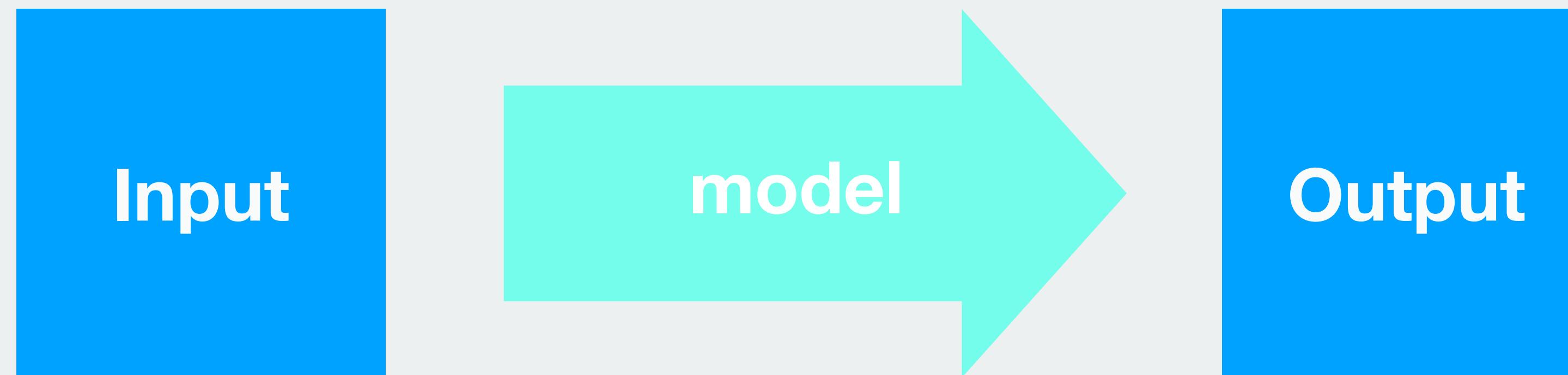


- Auto encoding



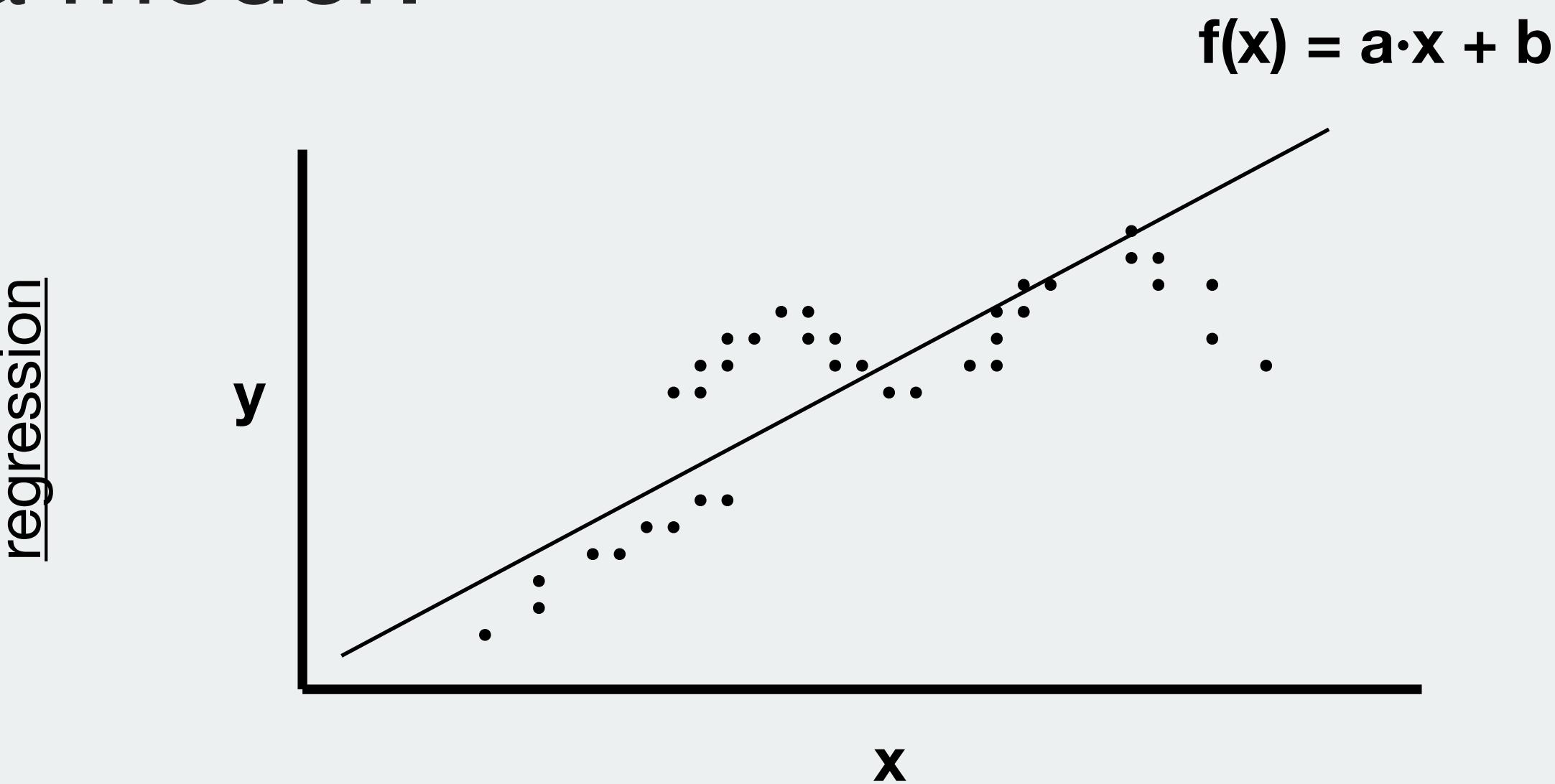
In a nutshell

You want the **model** that
best fits your **data**

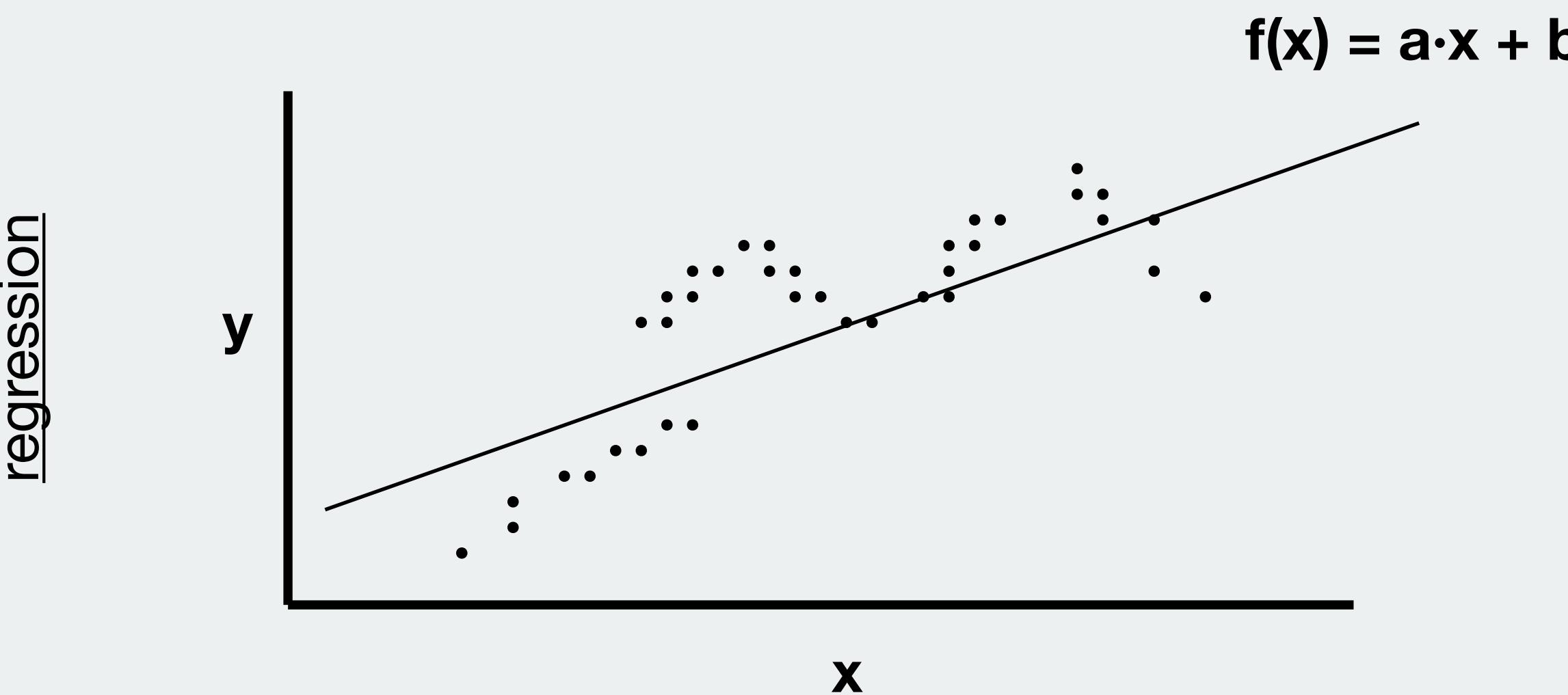


What is a model?

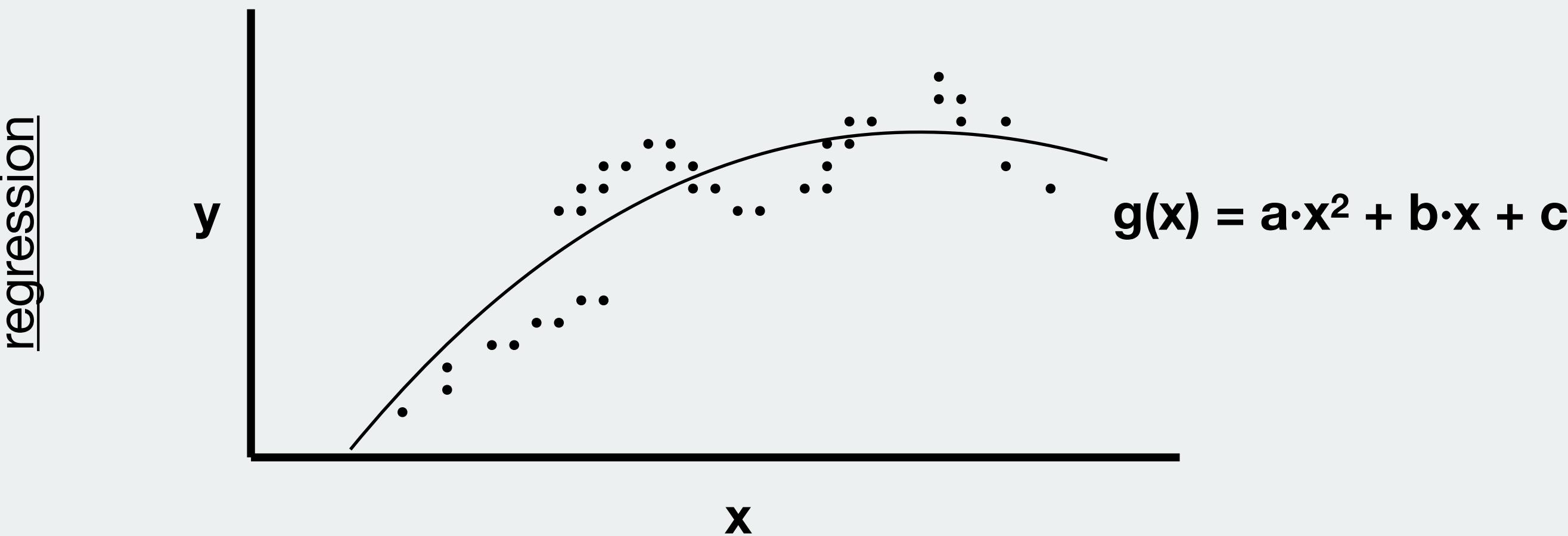
What is a model?



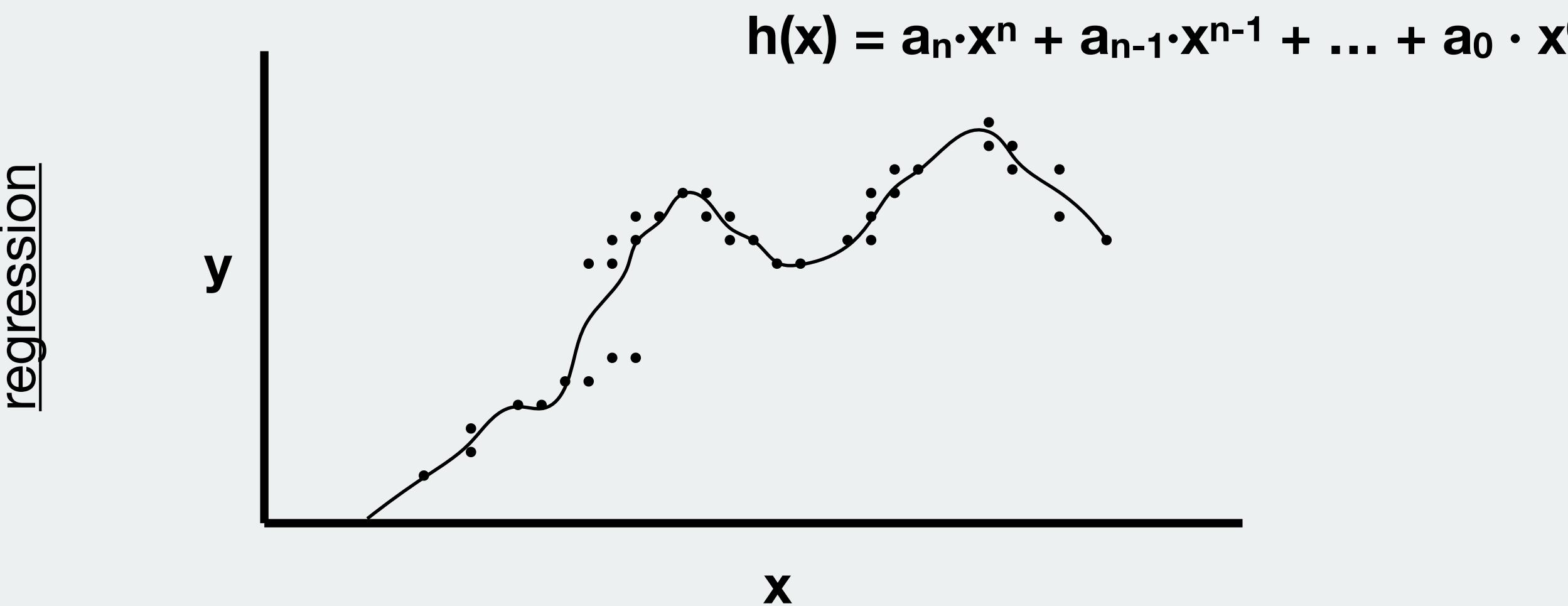
What is a model?



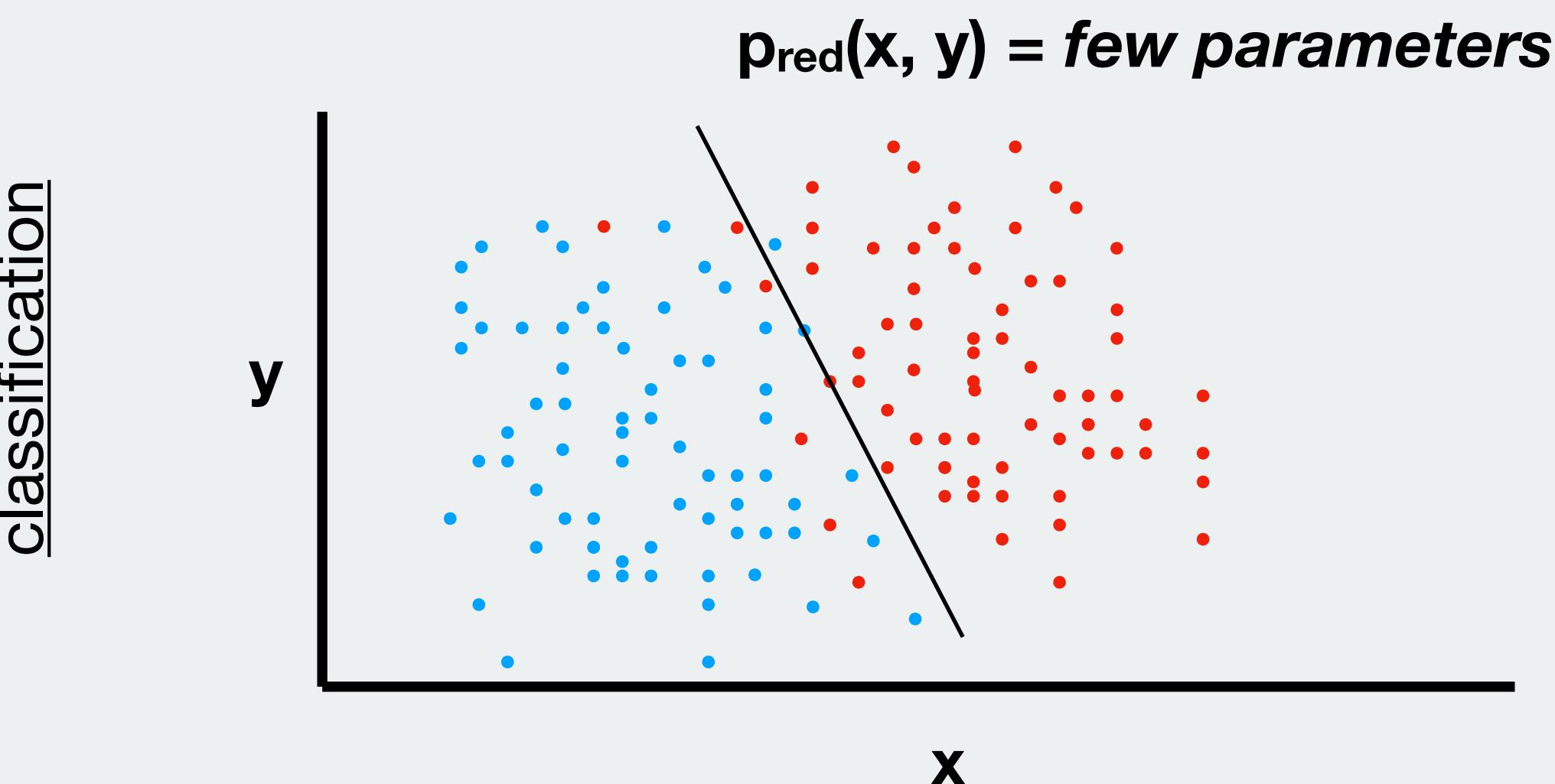
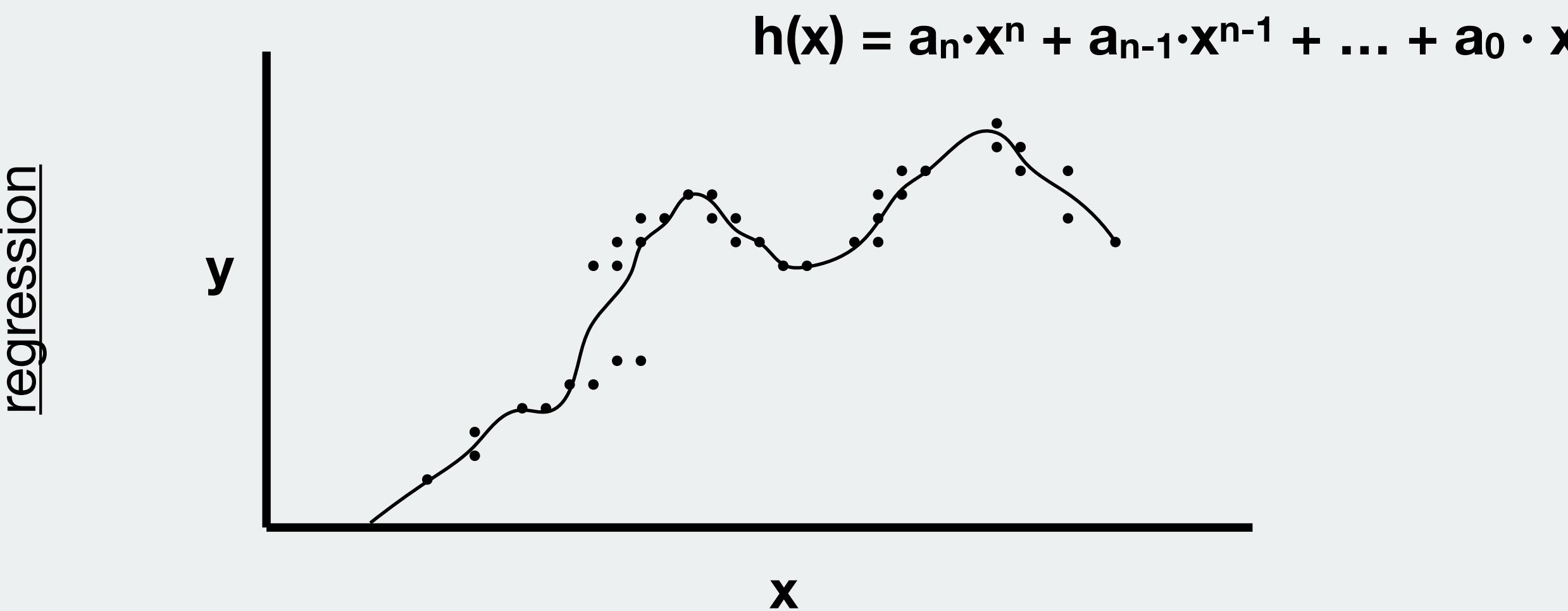
What is a model?



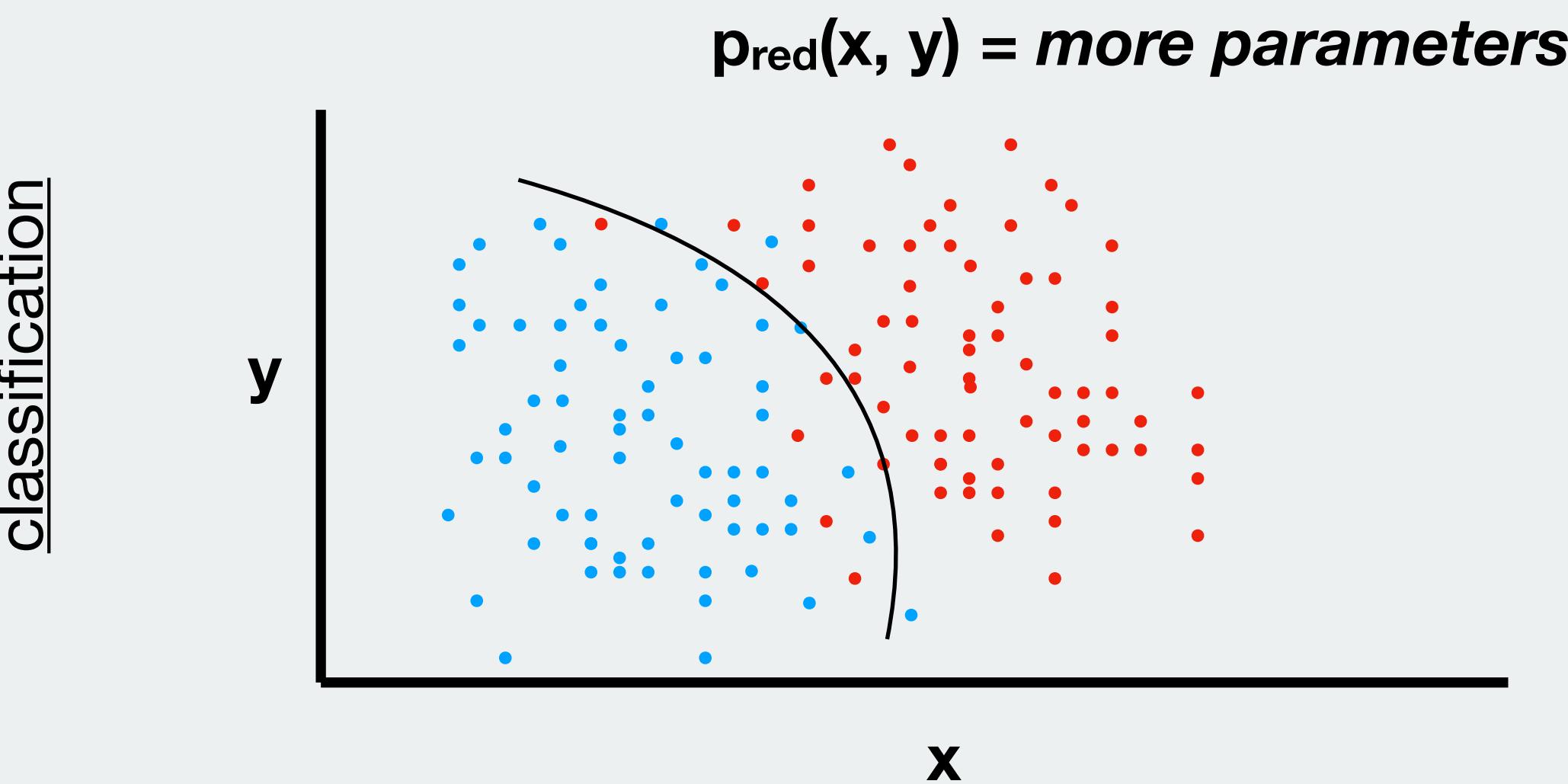
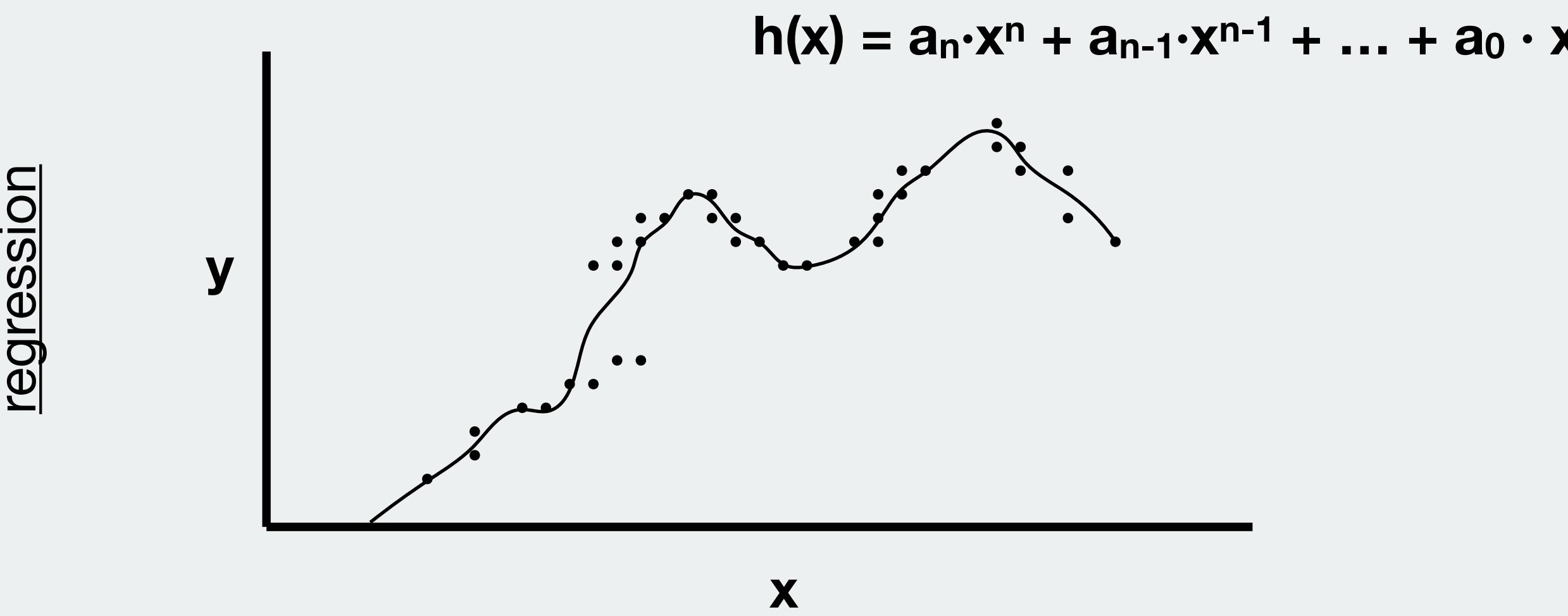
What is a model?



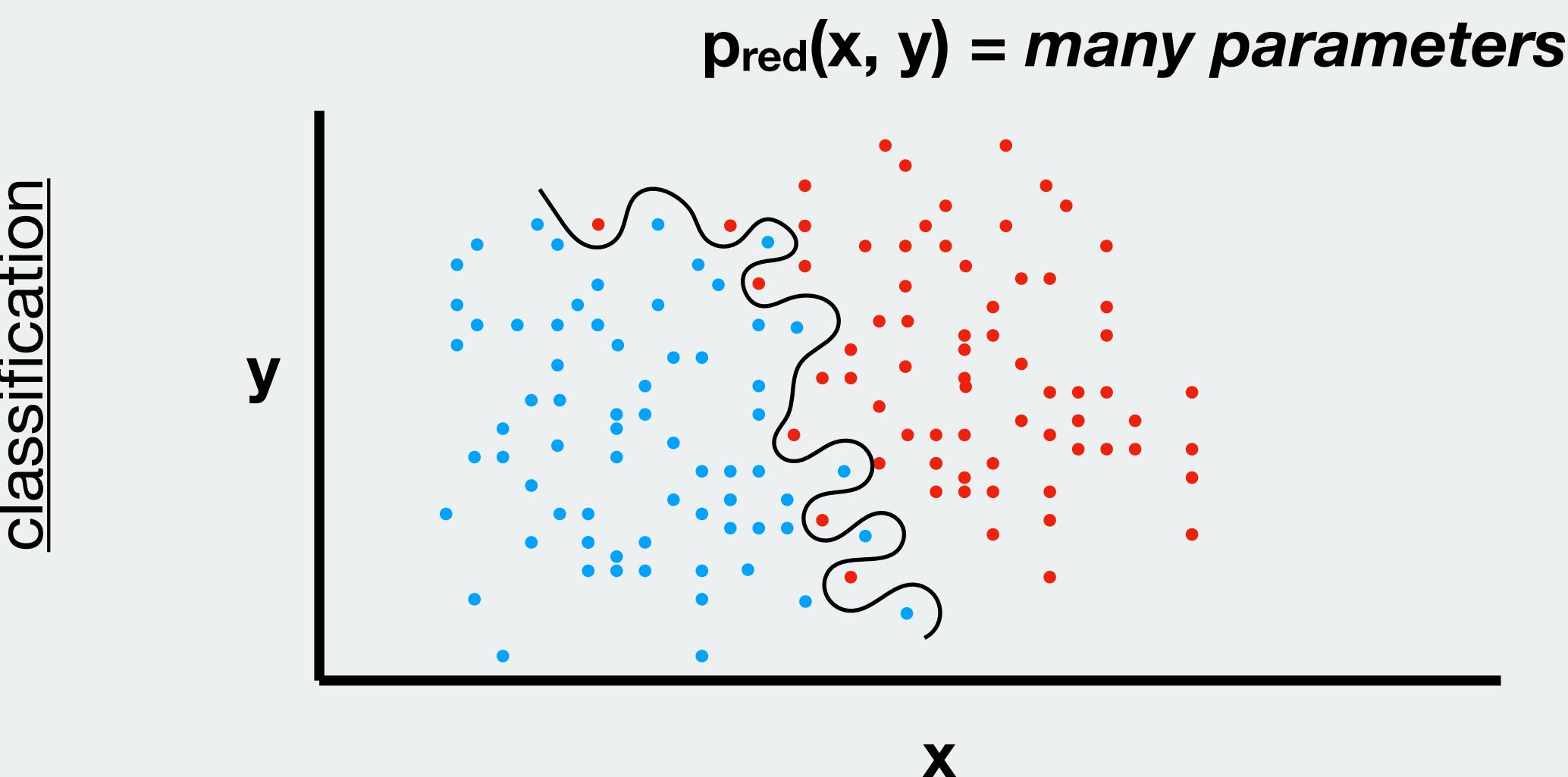
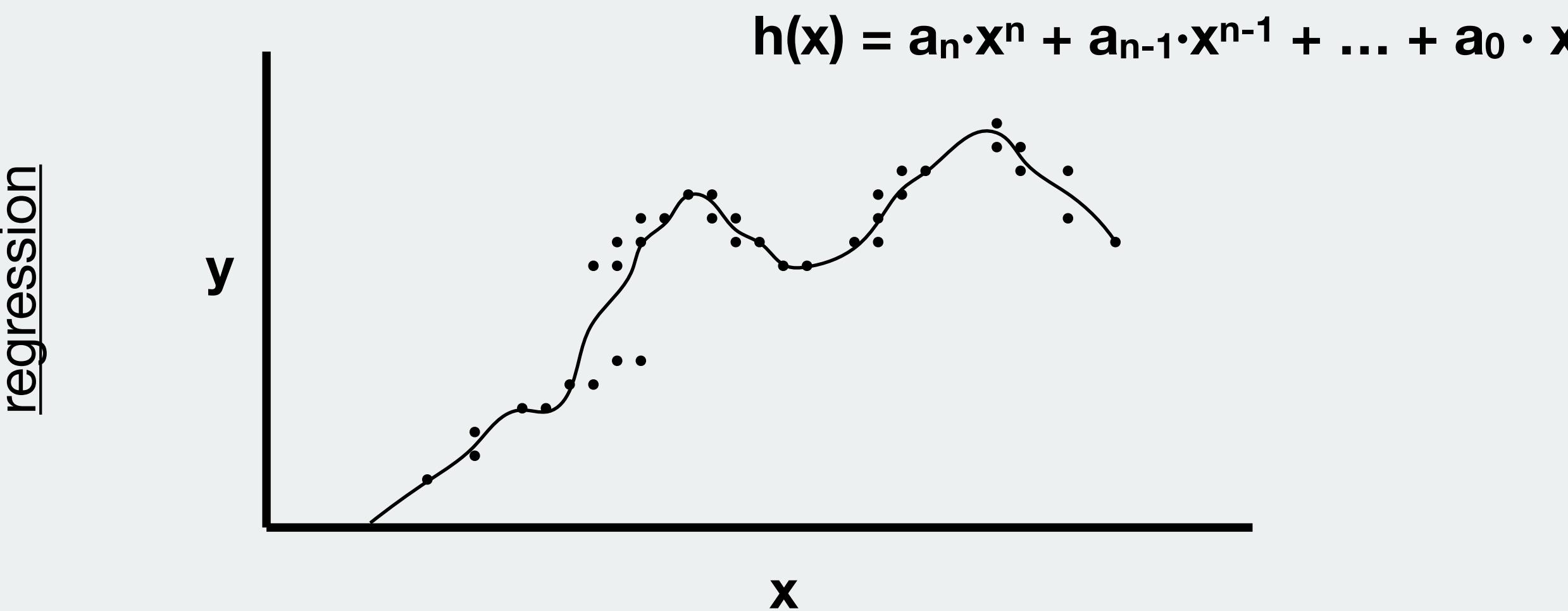
What is a model?



What is a model?



What is a model?



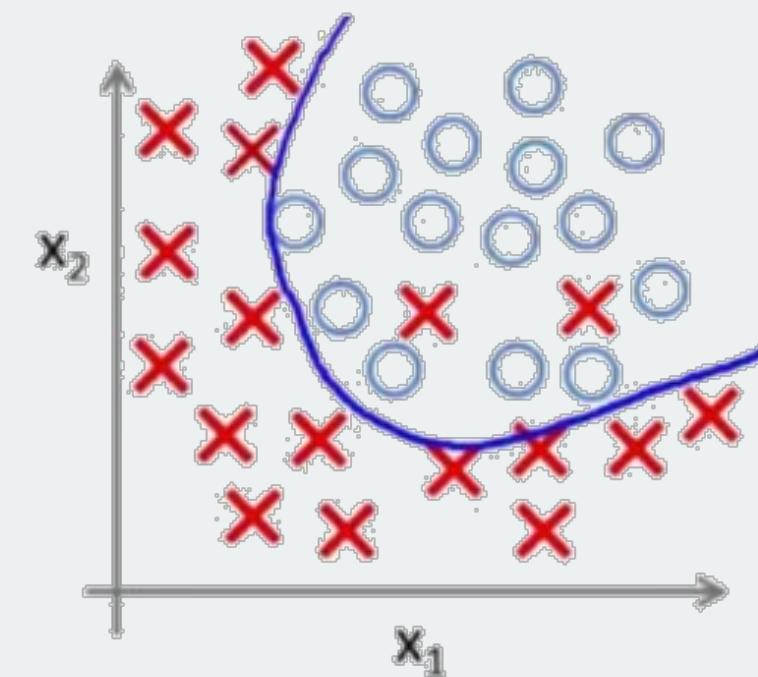
What is a model?



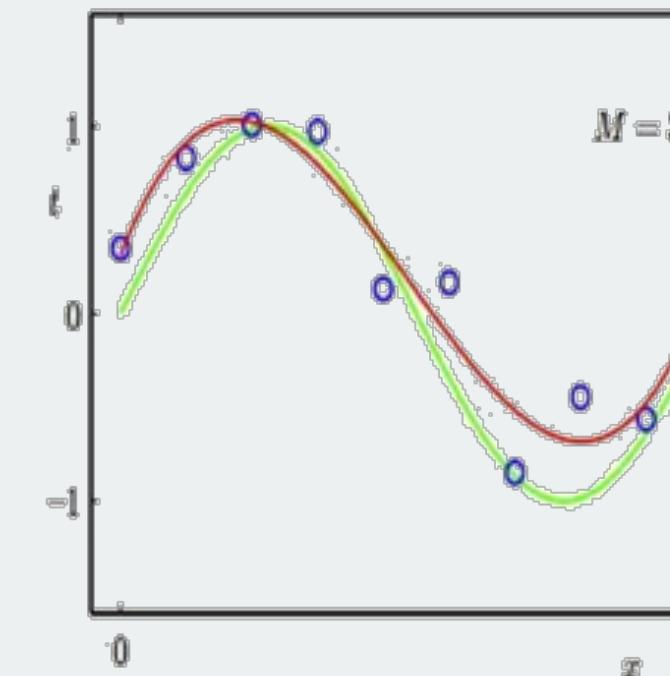
Underfitting and overfitting

Underfitting and overfitting

Classification



Regression

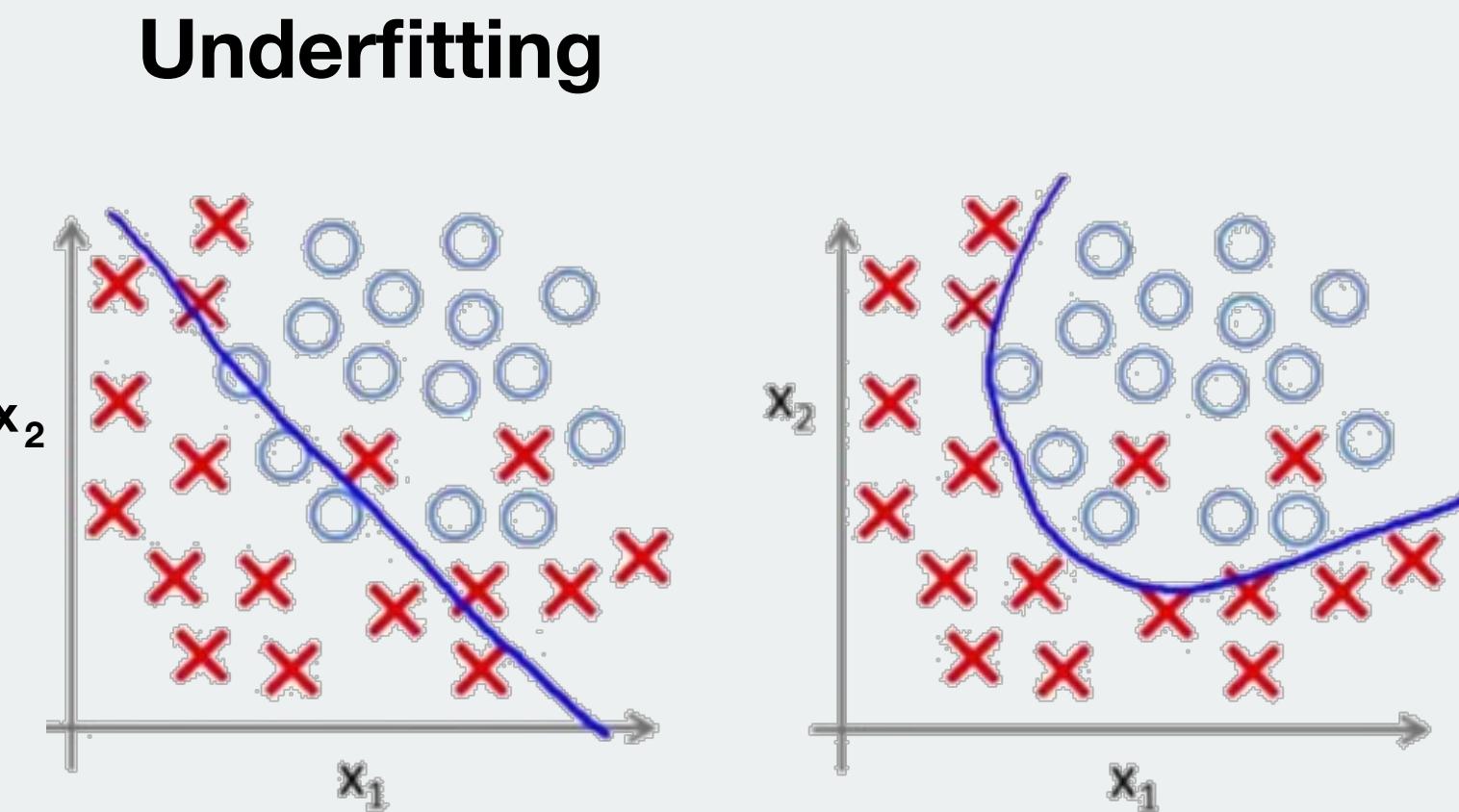


Small train error

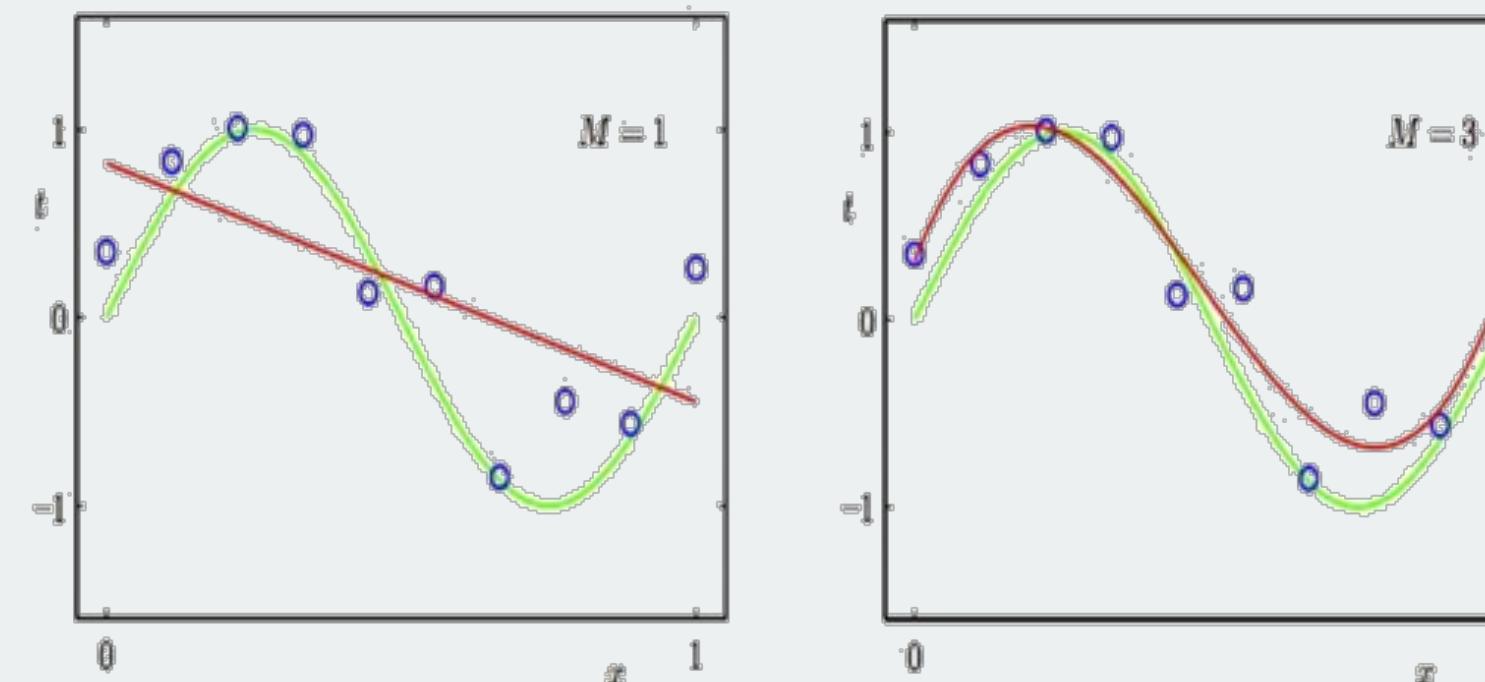
Small test error

Underfitting and overfitting

Classification



Regression



High train error

High test error

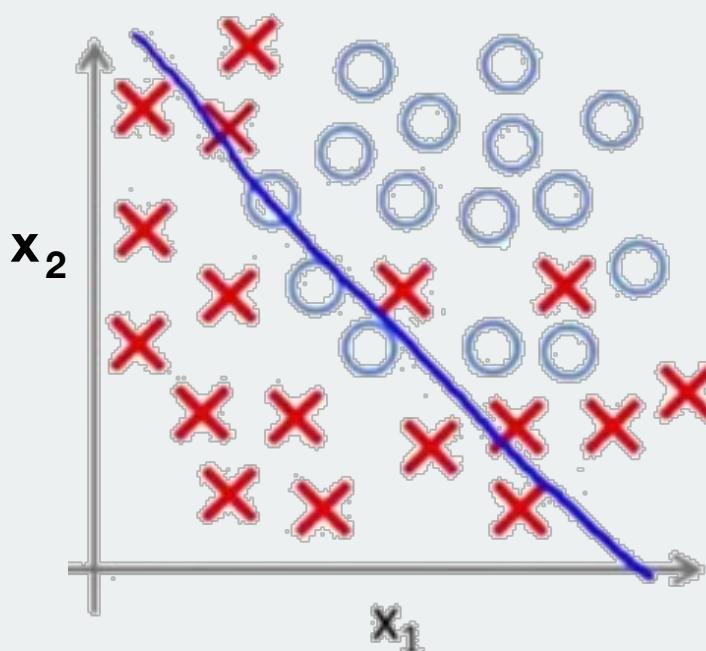
Small train error

Small test error

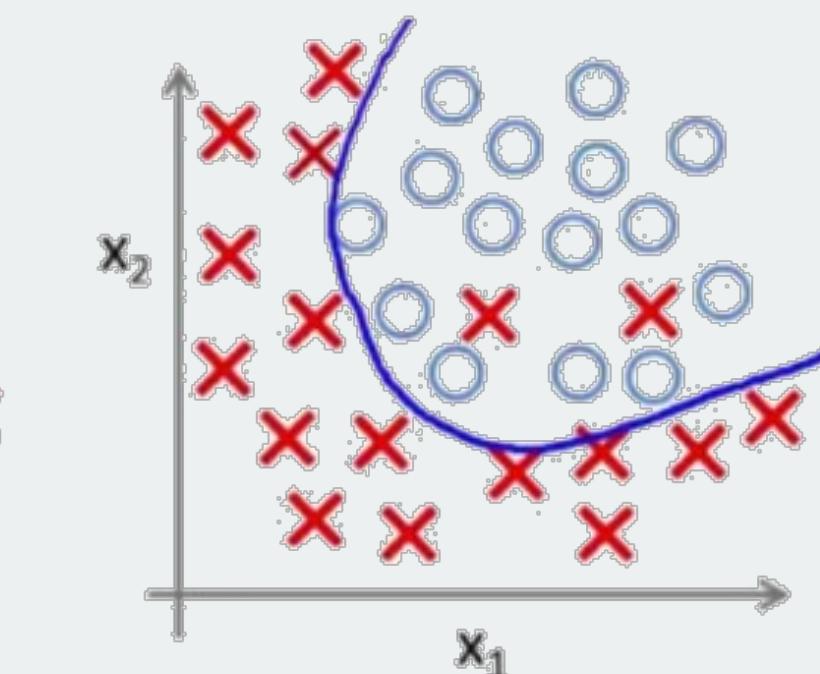
Underfitting and overfitting

Classification

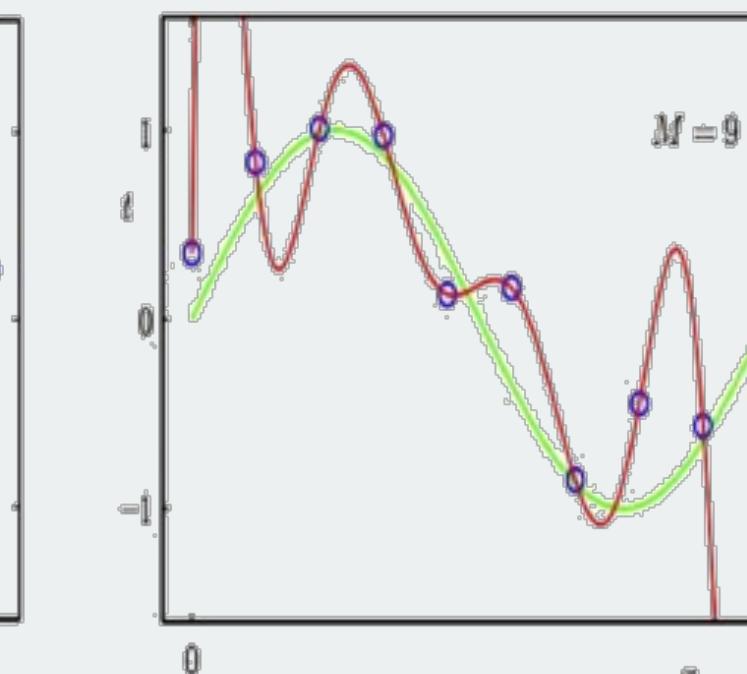
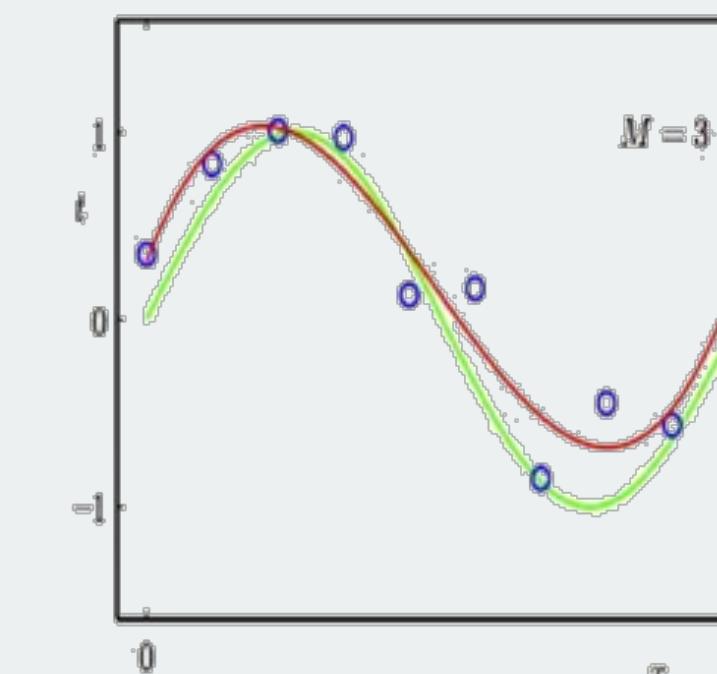
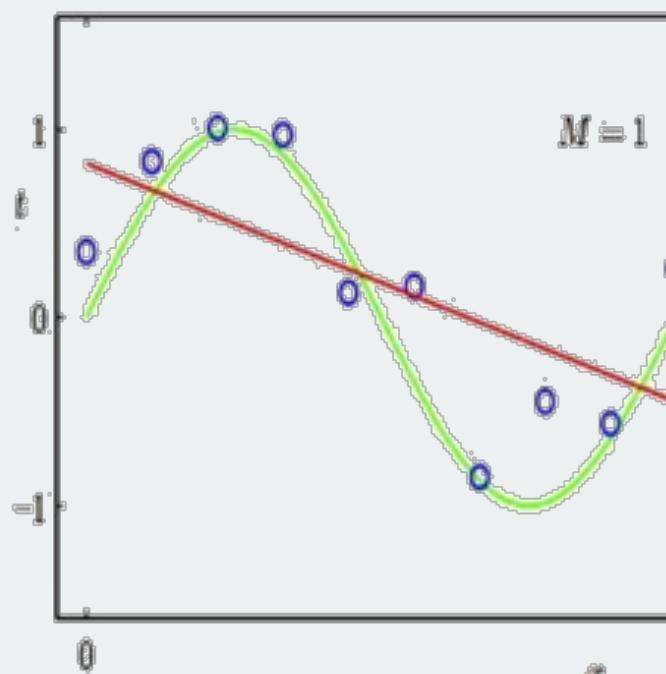
Underfitting



Overfitting



Regression



High train error

High test error

Small train error

Small test error

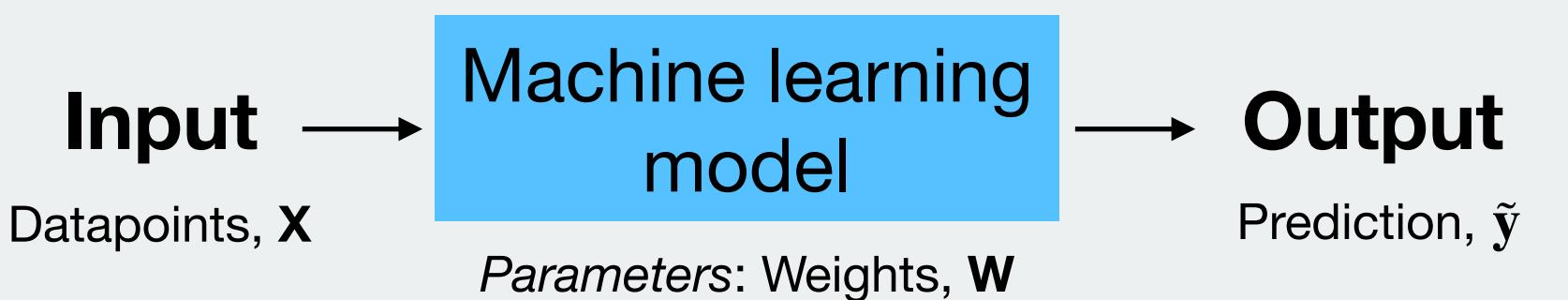
Zero train error

High test error

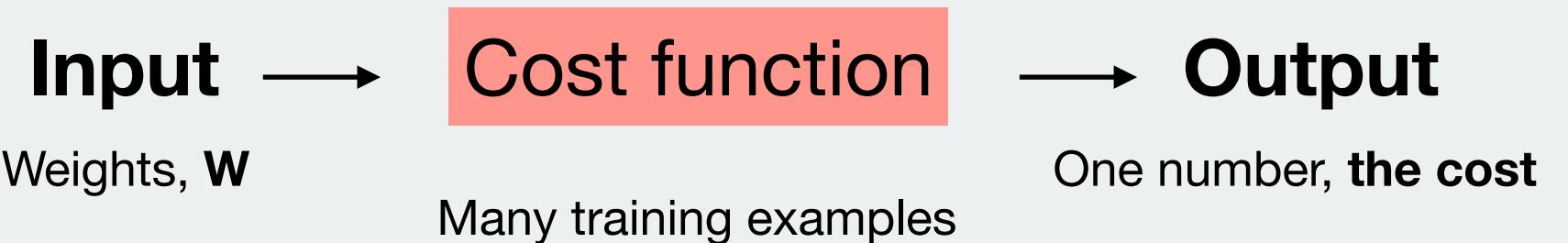
Finding model parameters

Finding model parameters

(1) The model

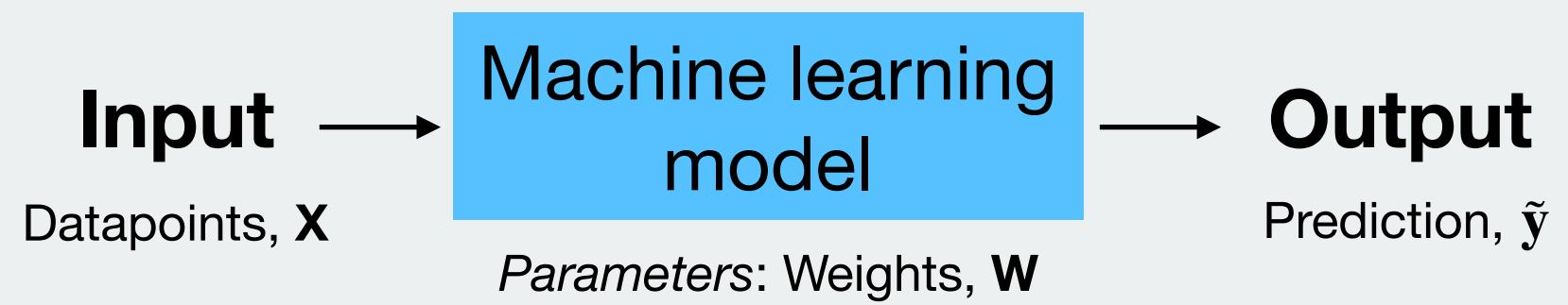


(2) Its performance

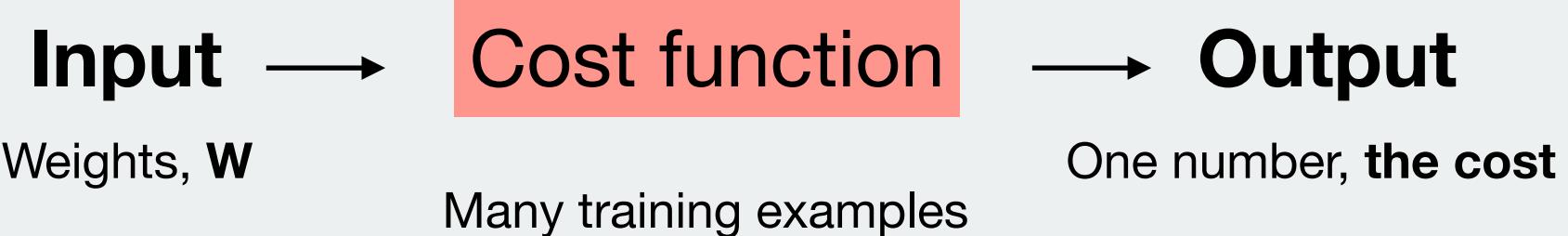


Finding model parameters

(1) The model



(2) Its performance



predicted from \mathbf{X} given \mathbf{W}

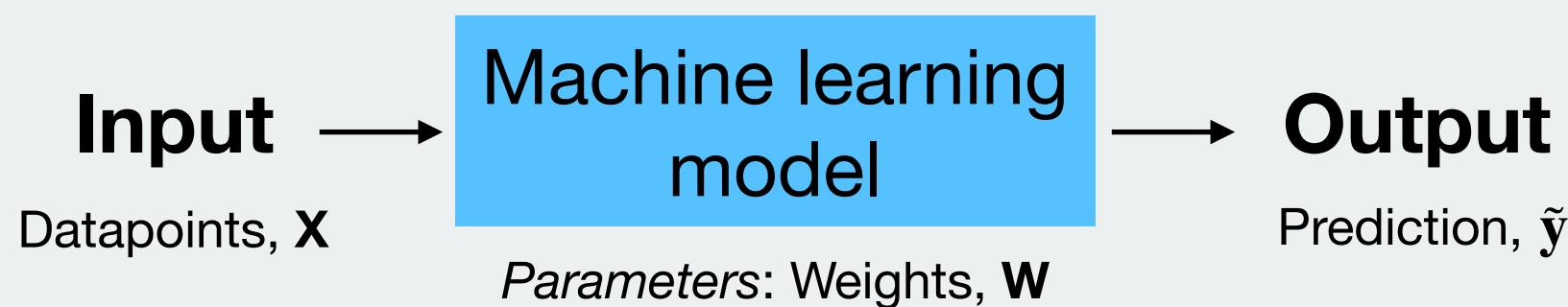
$$\tilde{\mathbf{y}} = \begin{bmatrix} 0.96 \\ 0.10 \\ 0.04 \\ \dots \\ 0.70 \\ 0.02 \\ 0.99 \end{bmatrix}$$

true

$$\mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \dots \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

Finding model parameters

(1) The model



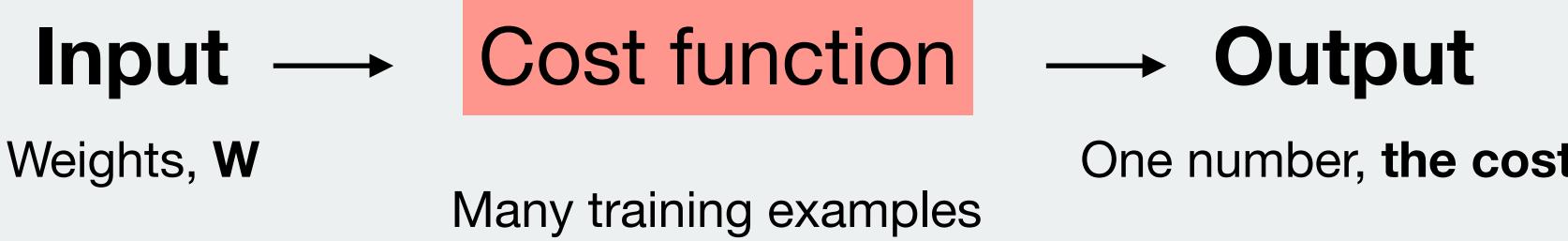
predicted from \mathbf{X} given \mathbf{W}

$$\tilde{\mathbf{y}} = \begin{bmatrix} 0.96 \\ 0.10 \\ 0.04 \\ \dots \\ 0.70 \\ 0.02 \\ 0.99 \end{bmatrix}$$

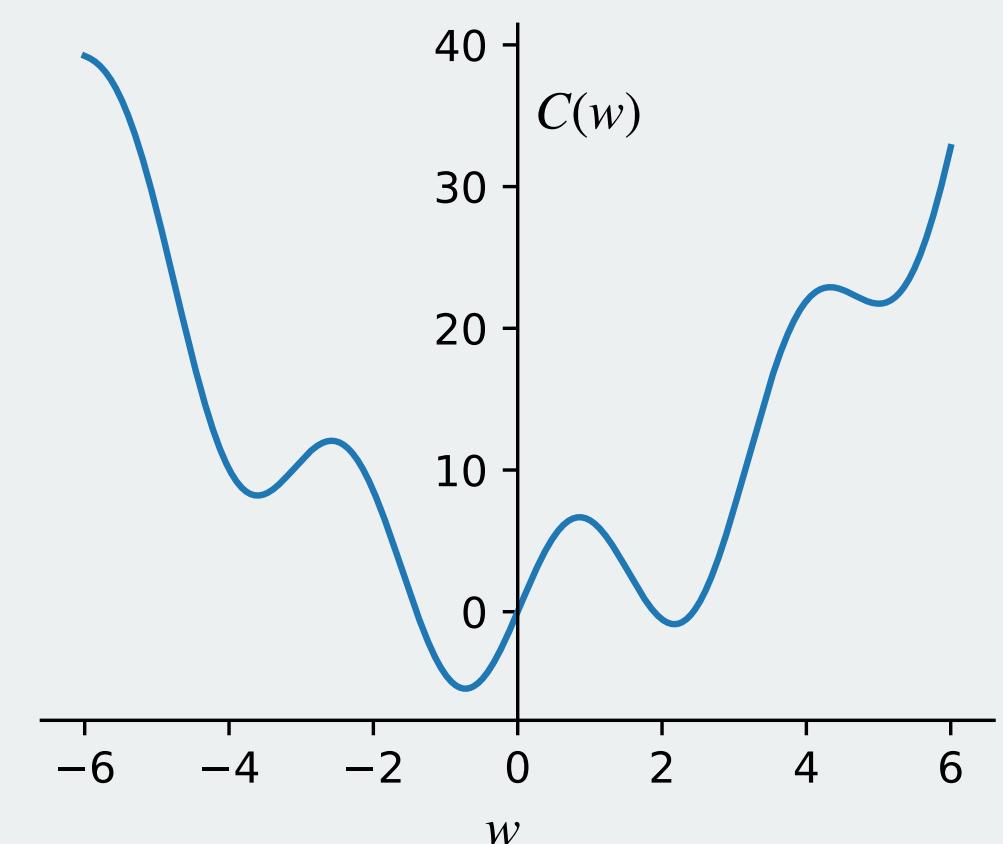
true

$$\mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \dots \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

(2) Its performance

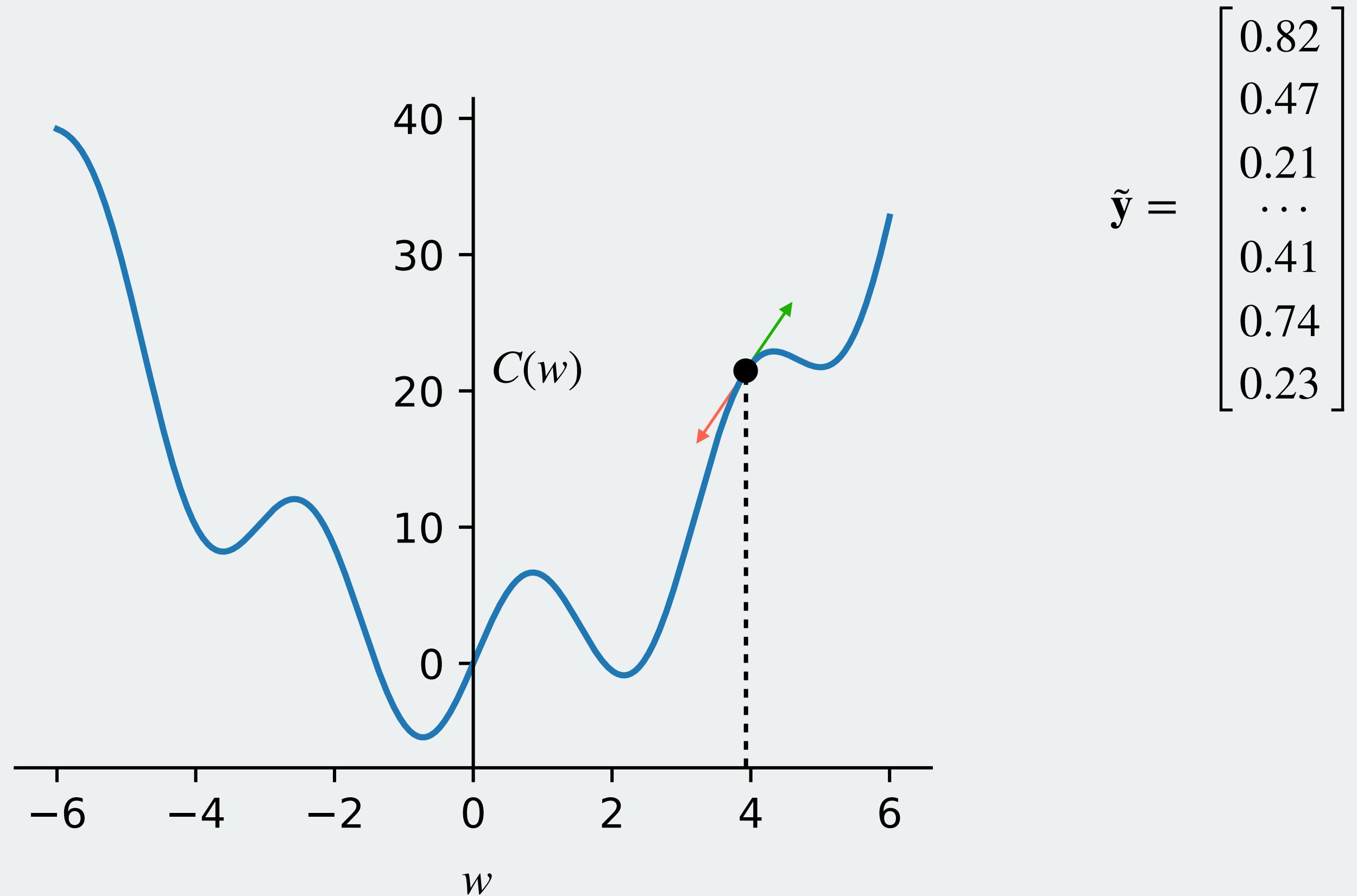


$$\begin{aligned} C(\mathbf{W}) &= \frac{1}{N} \sum_i (\tilde{y}_i - y_i)^2 \\ &= (0.96 - 1)^2 \\ &\quad + (0.10 - 0)^2 \\ &\quad + (0.04 - 0)^2 \\ &\quad + \dots \\ &\quad + (0.70 - 1)^2 \\ &\quad + (0.02 - 0)^2 \\ &\quad + (0.99 - 1)^2 \end{aligned}$$



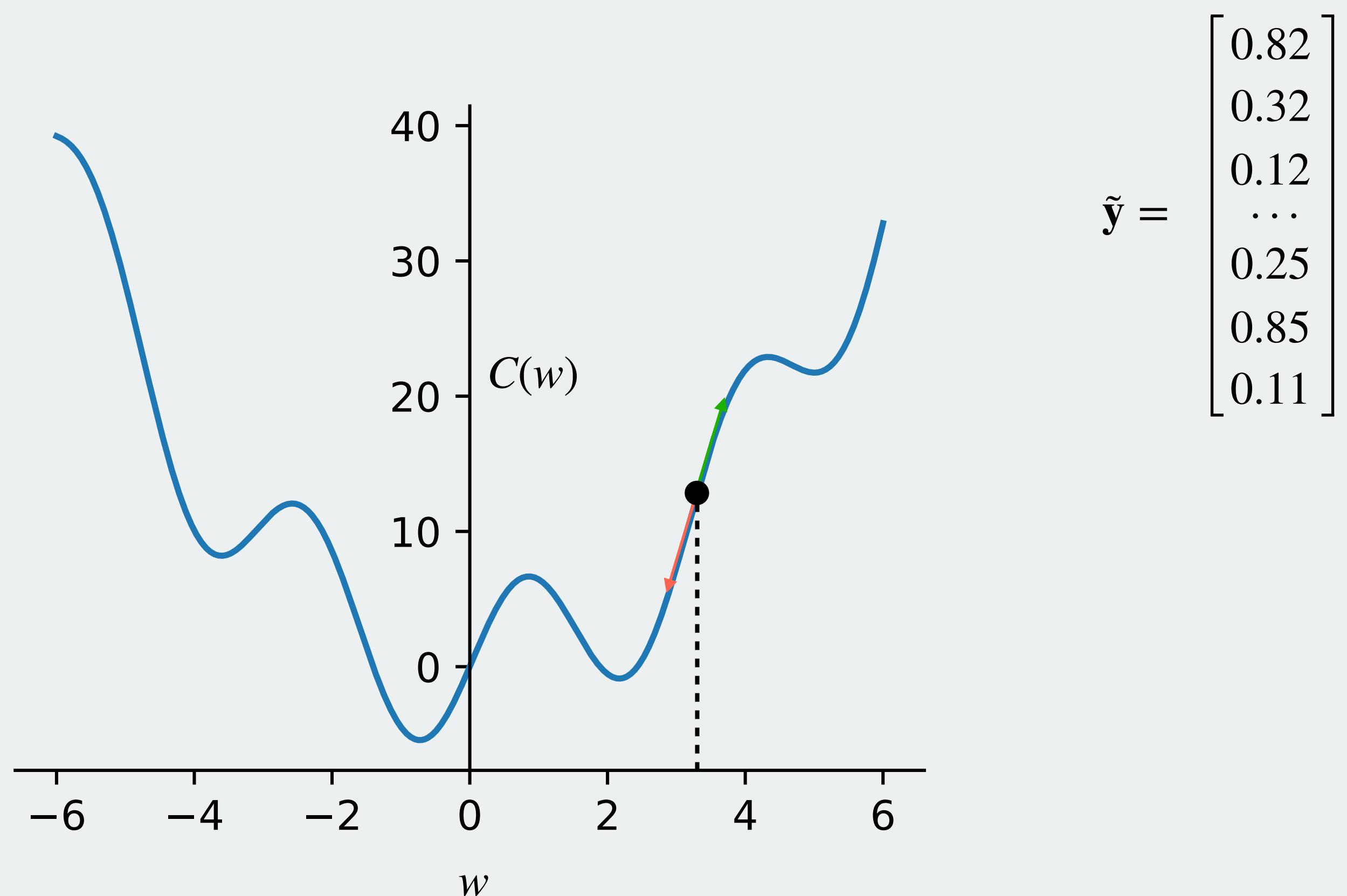
Finding model parameters

> Gradient Descent



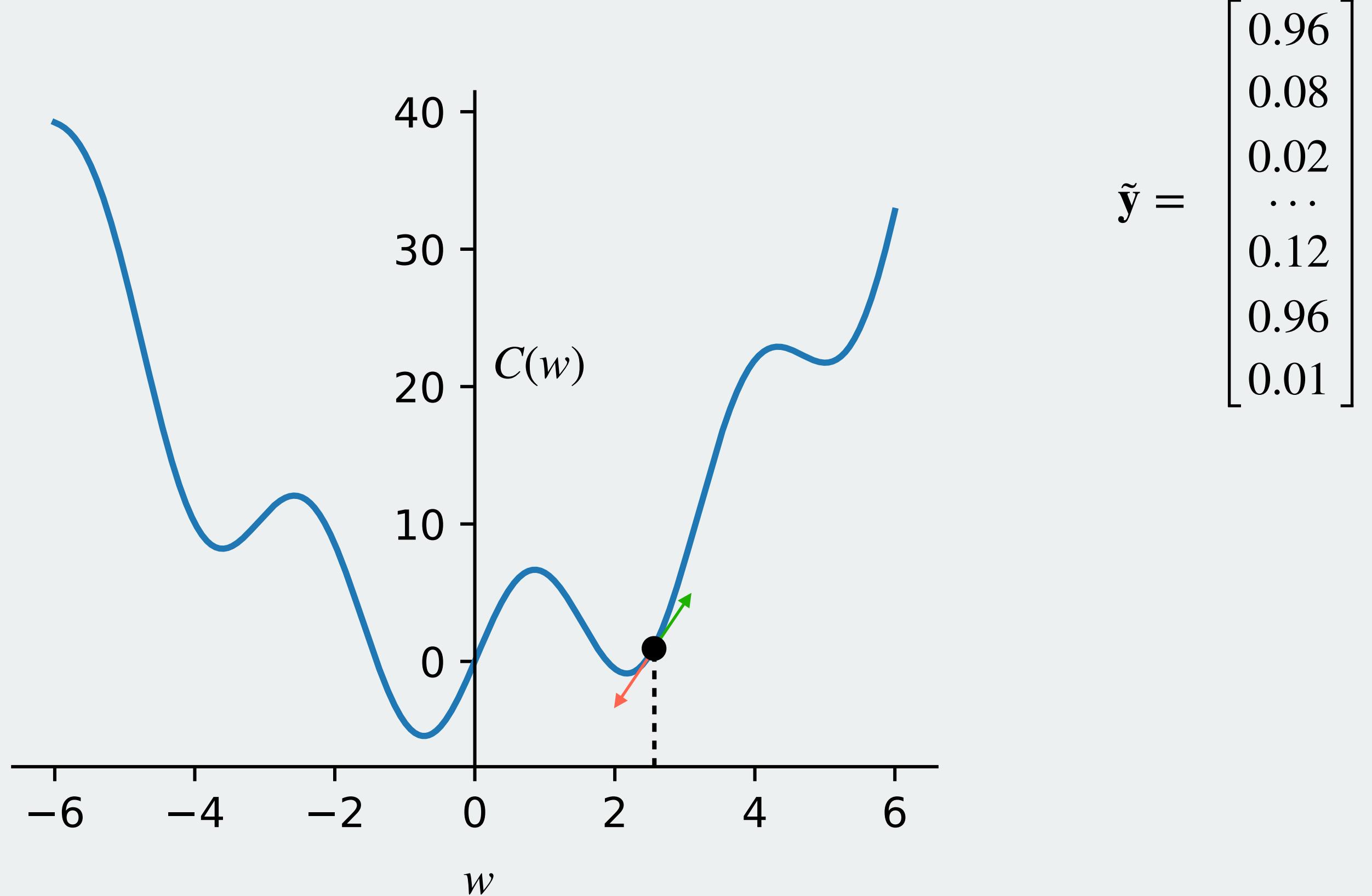
Finding model parameters

> Gradient Descent



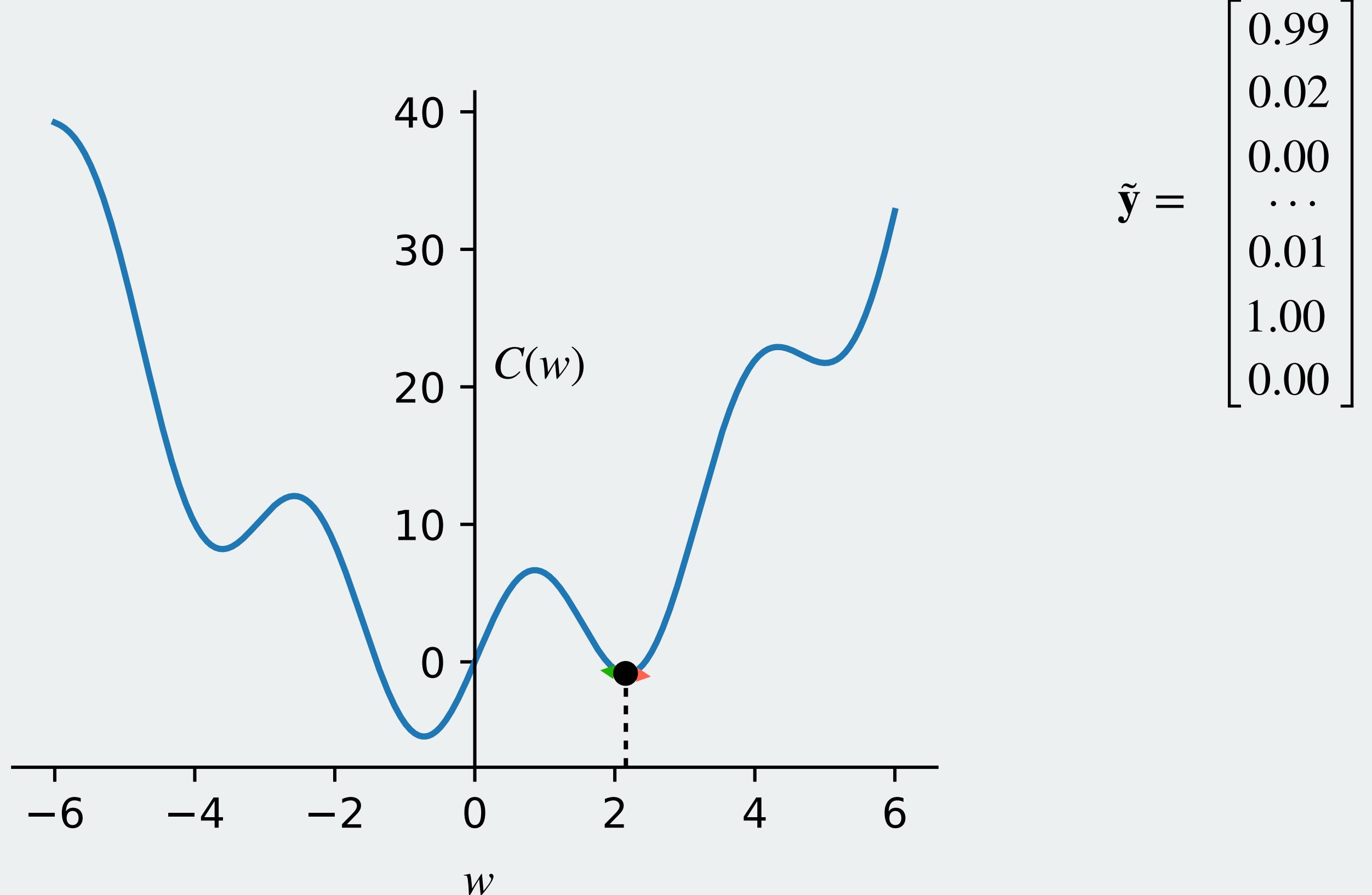
Finding model parameters

> Gradient Descent



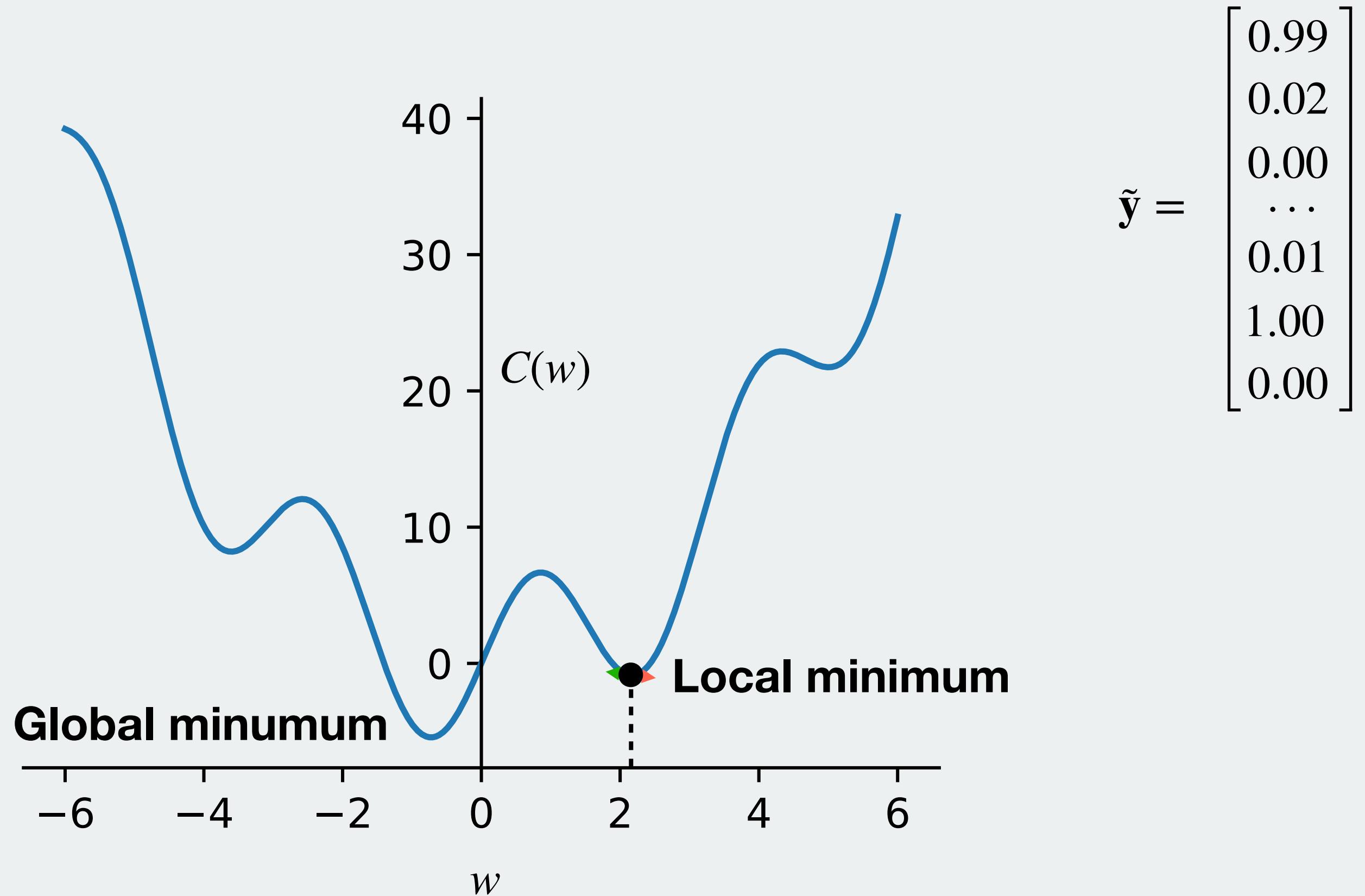
Finding model parameters

> Gradient Descent



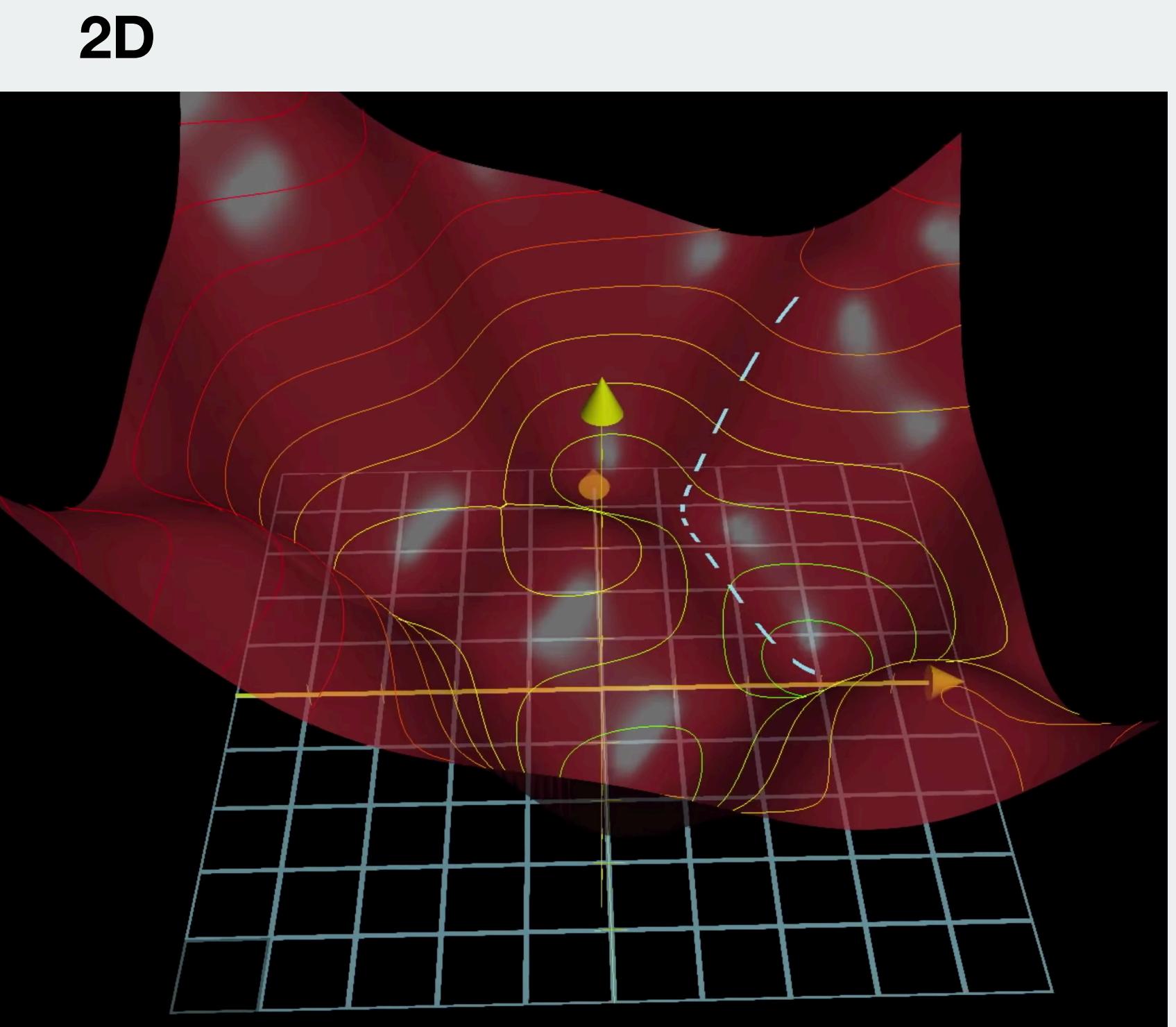
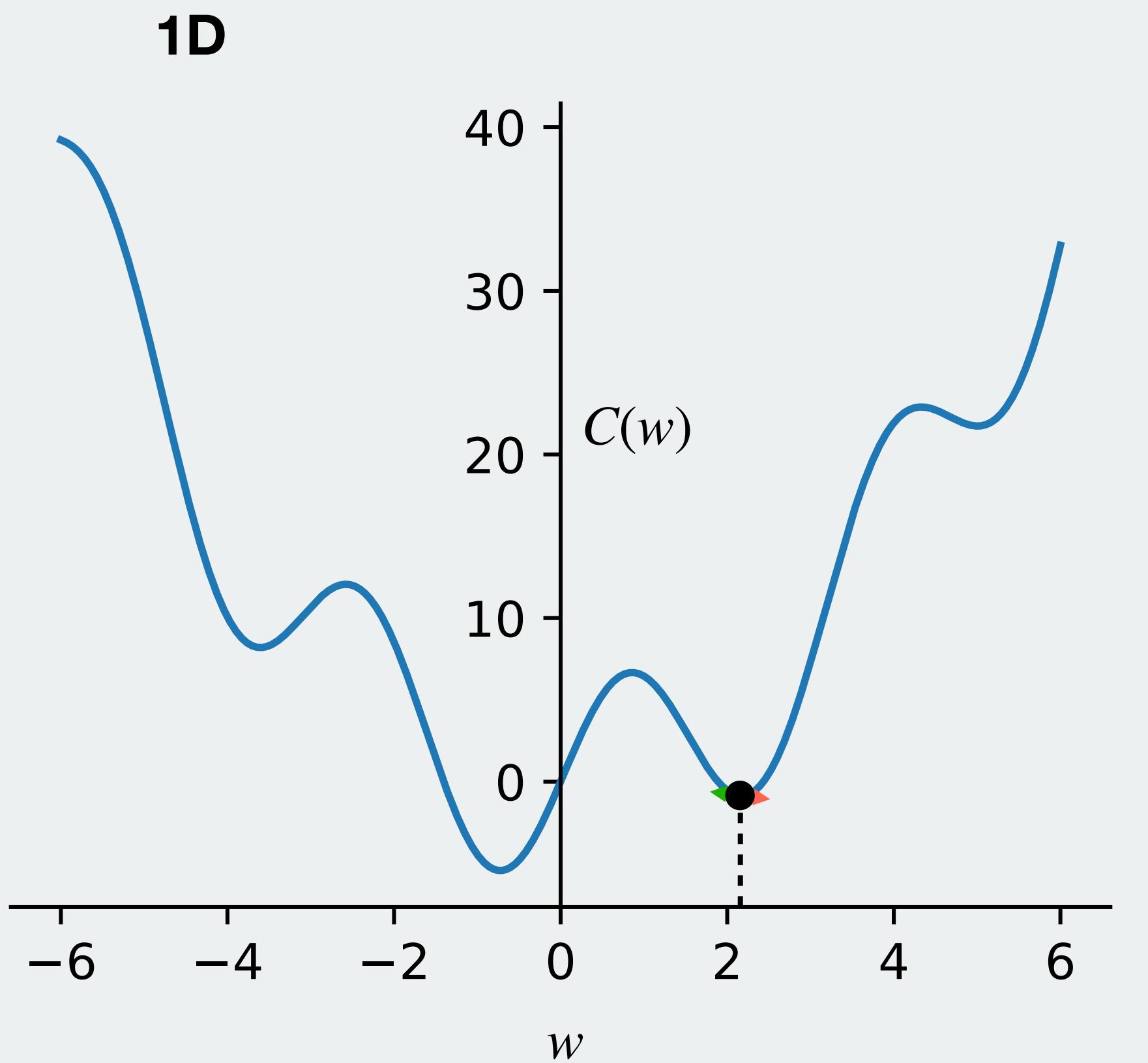
Finding model parameters

> Gradient Descent



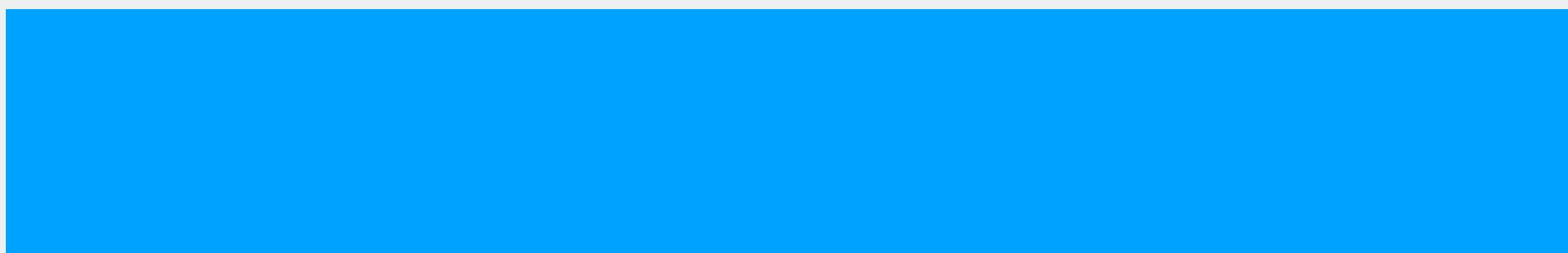
Finding model parameters

> Gradient Descent

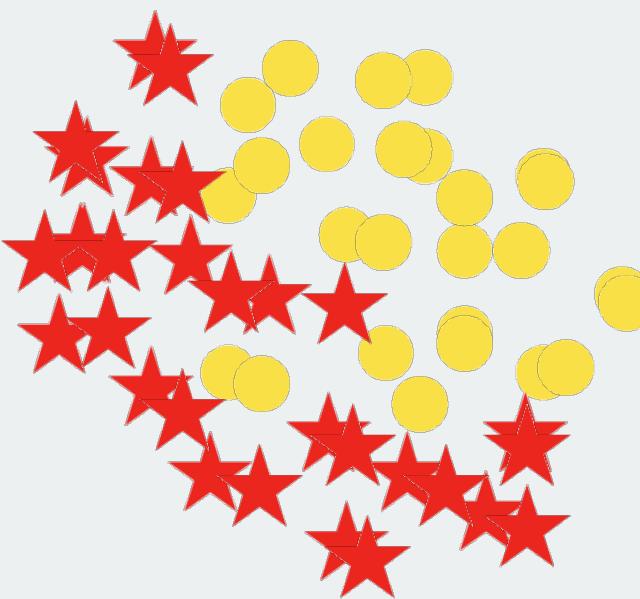


Model evaluation

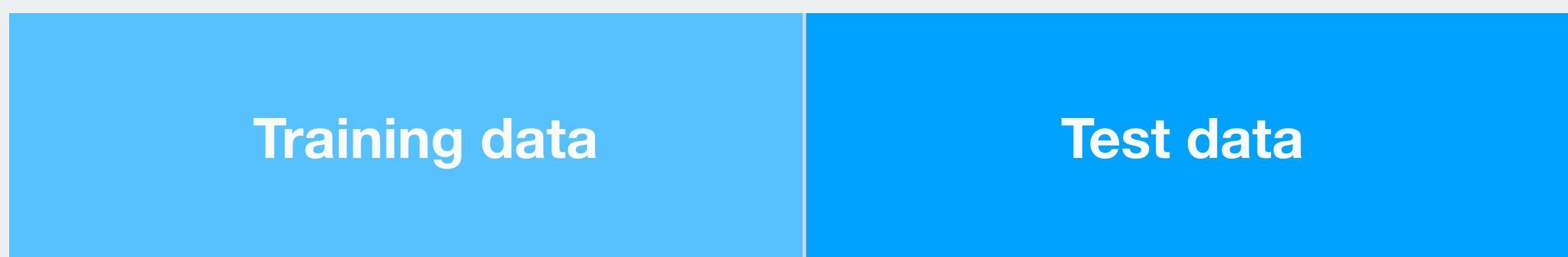
Model evaluation



Data



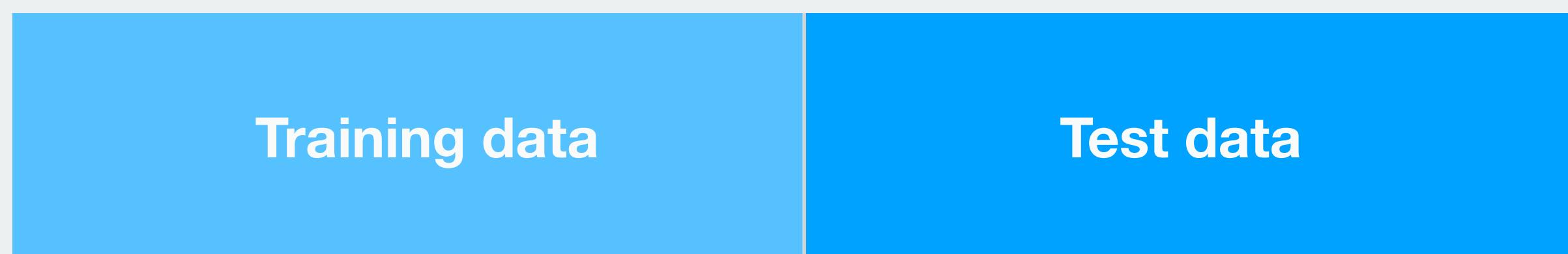
Model evaluation



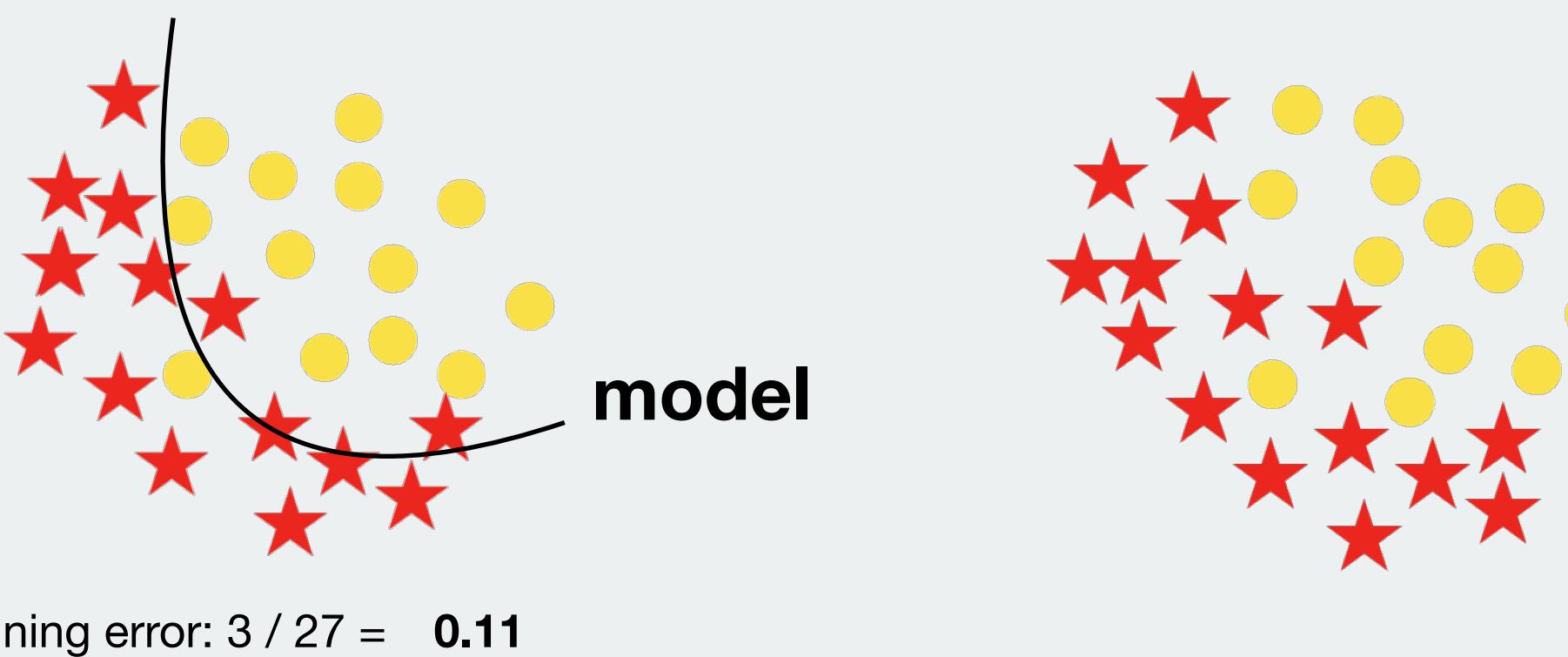
Data



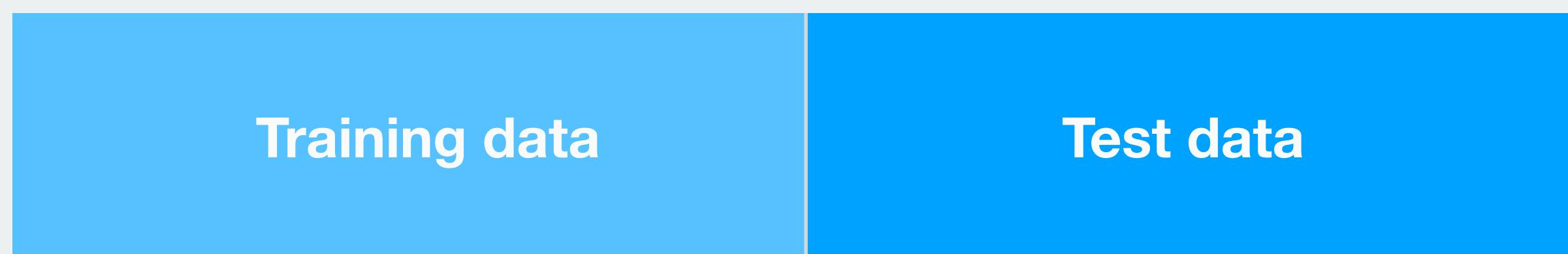
Model evaluation



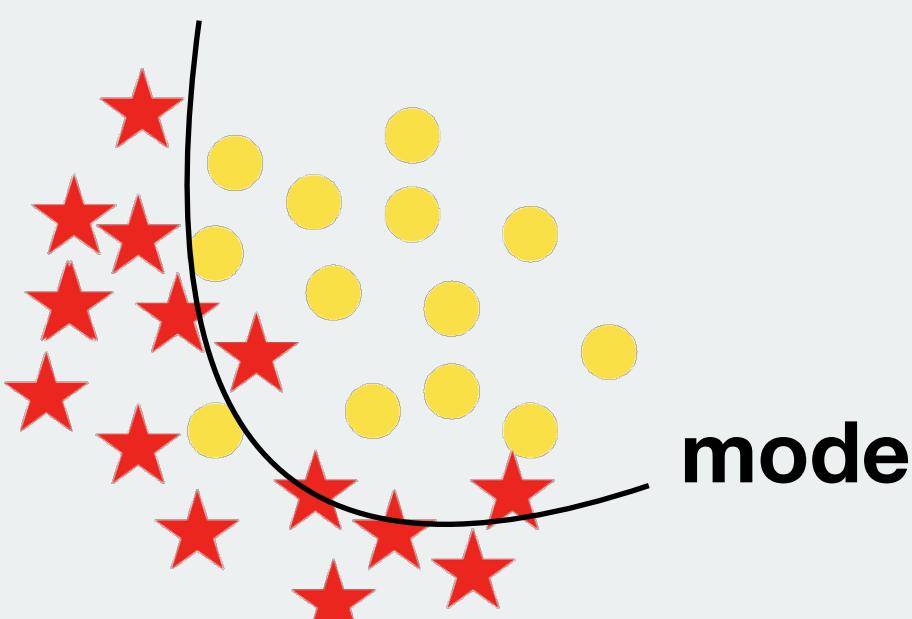
Data



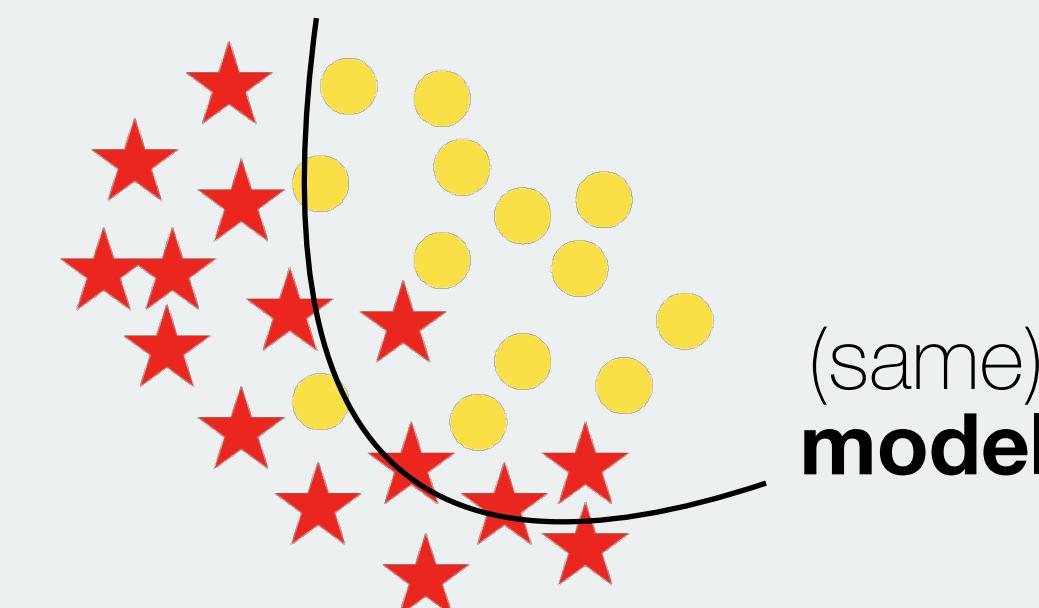
Model evaluation



Data



training error: 3 / 27 = **0.11**

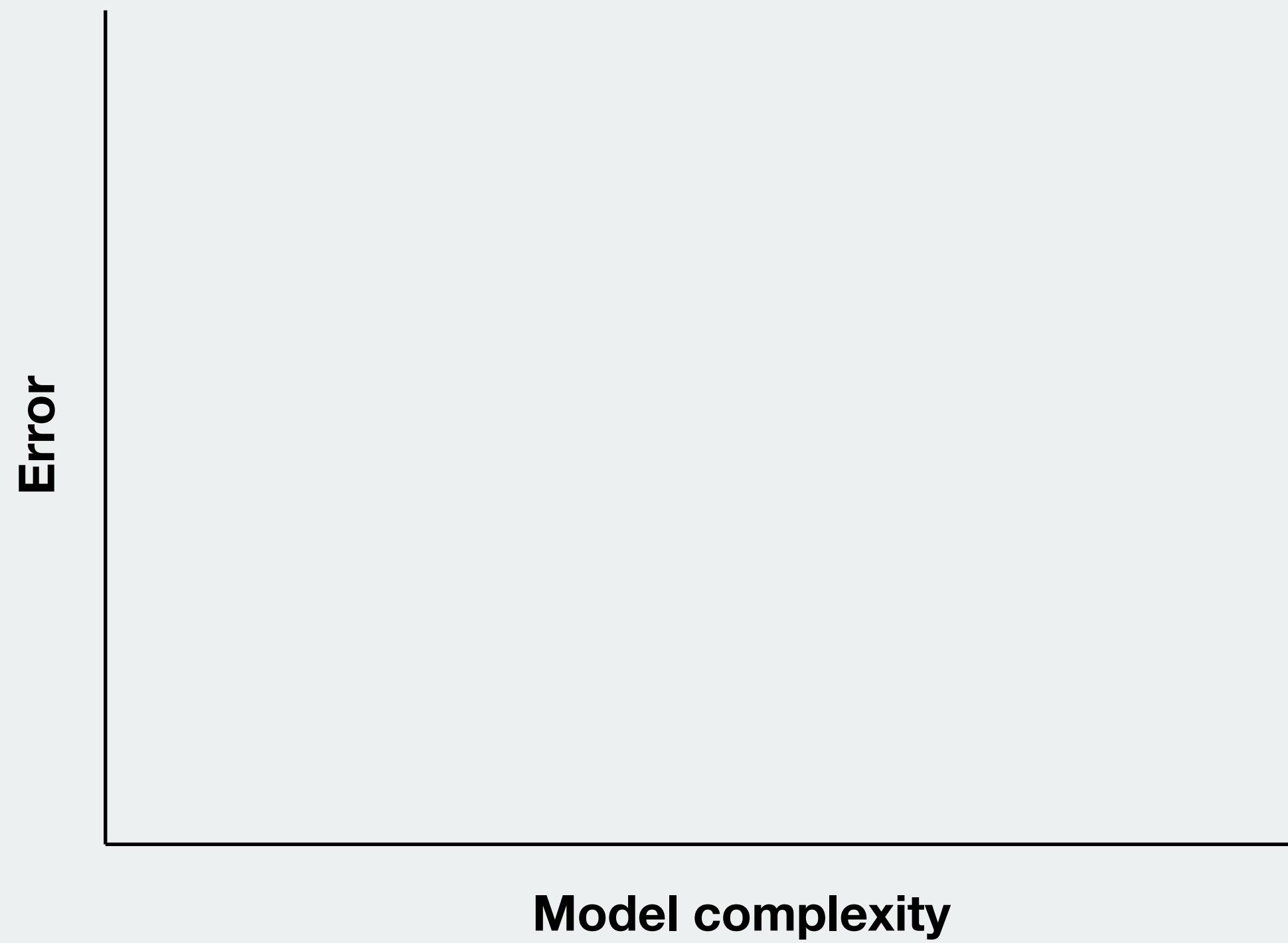


test error: 5 / 27 = **0.18**

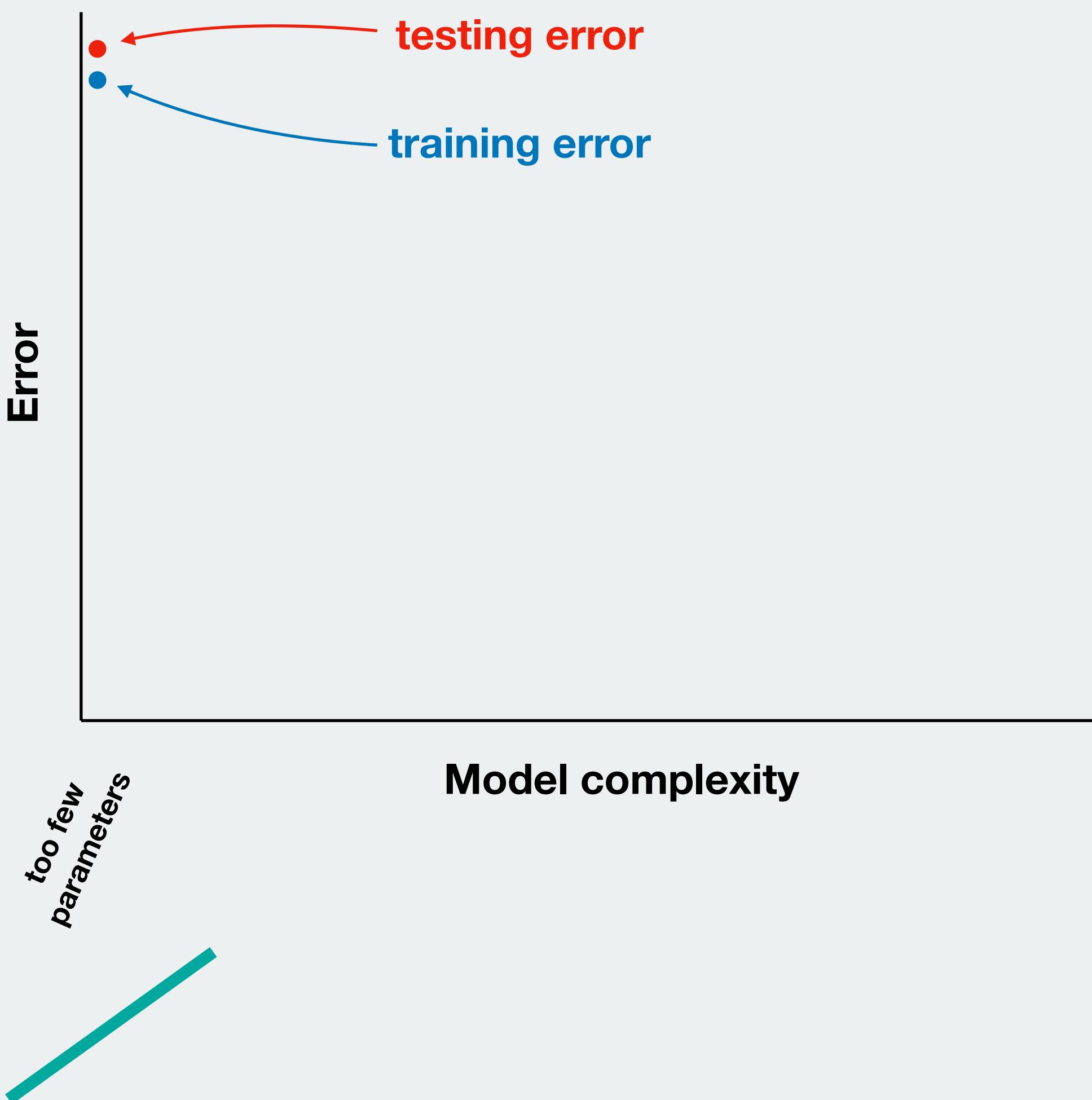
(same)
model

Avoiding underfitting and overfitting

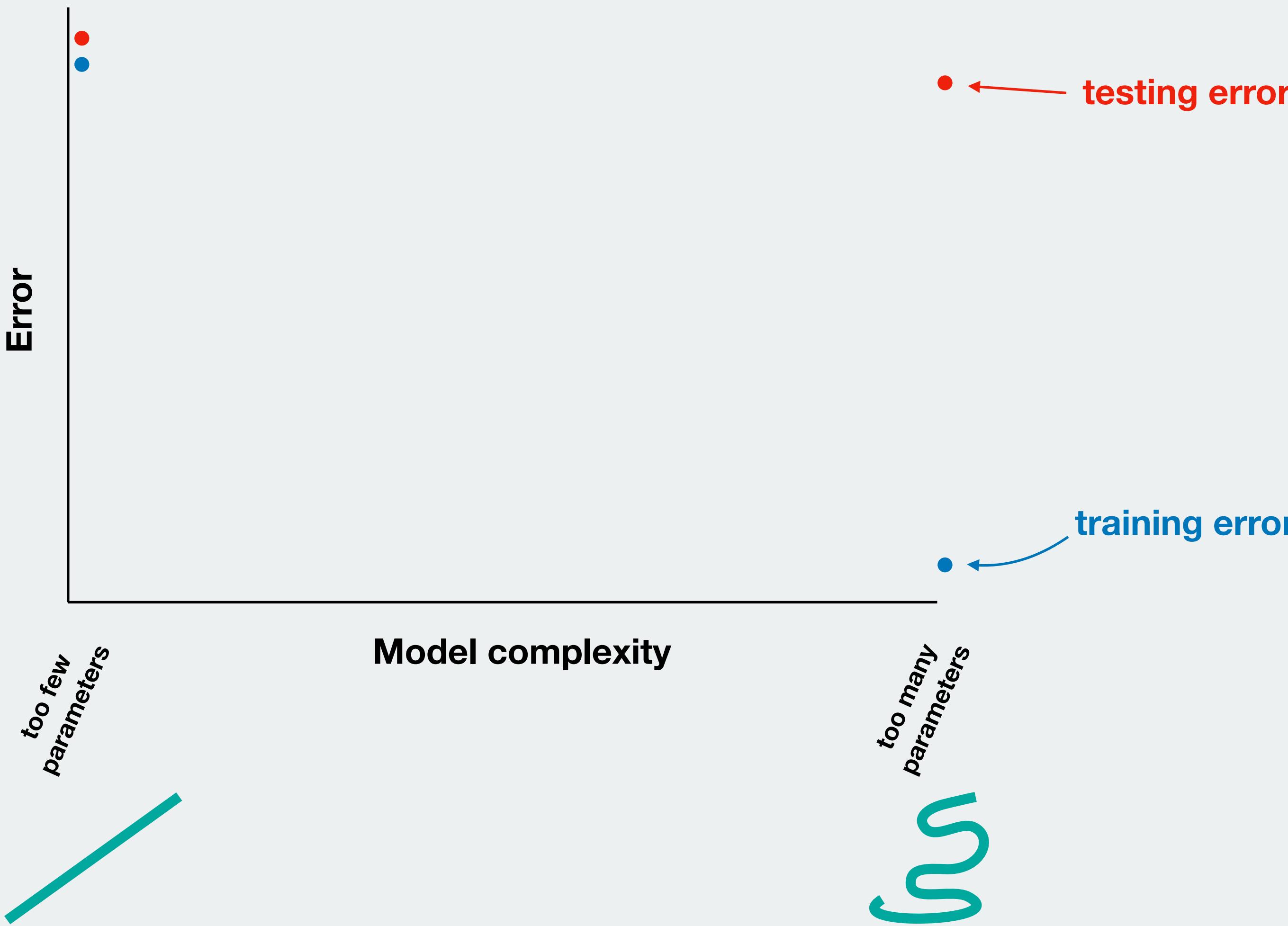
Avoiding underfitting and overfitting



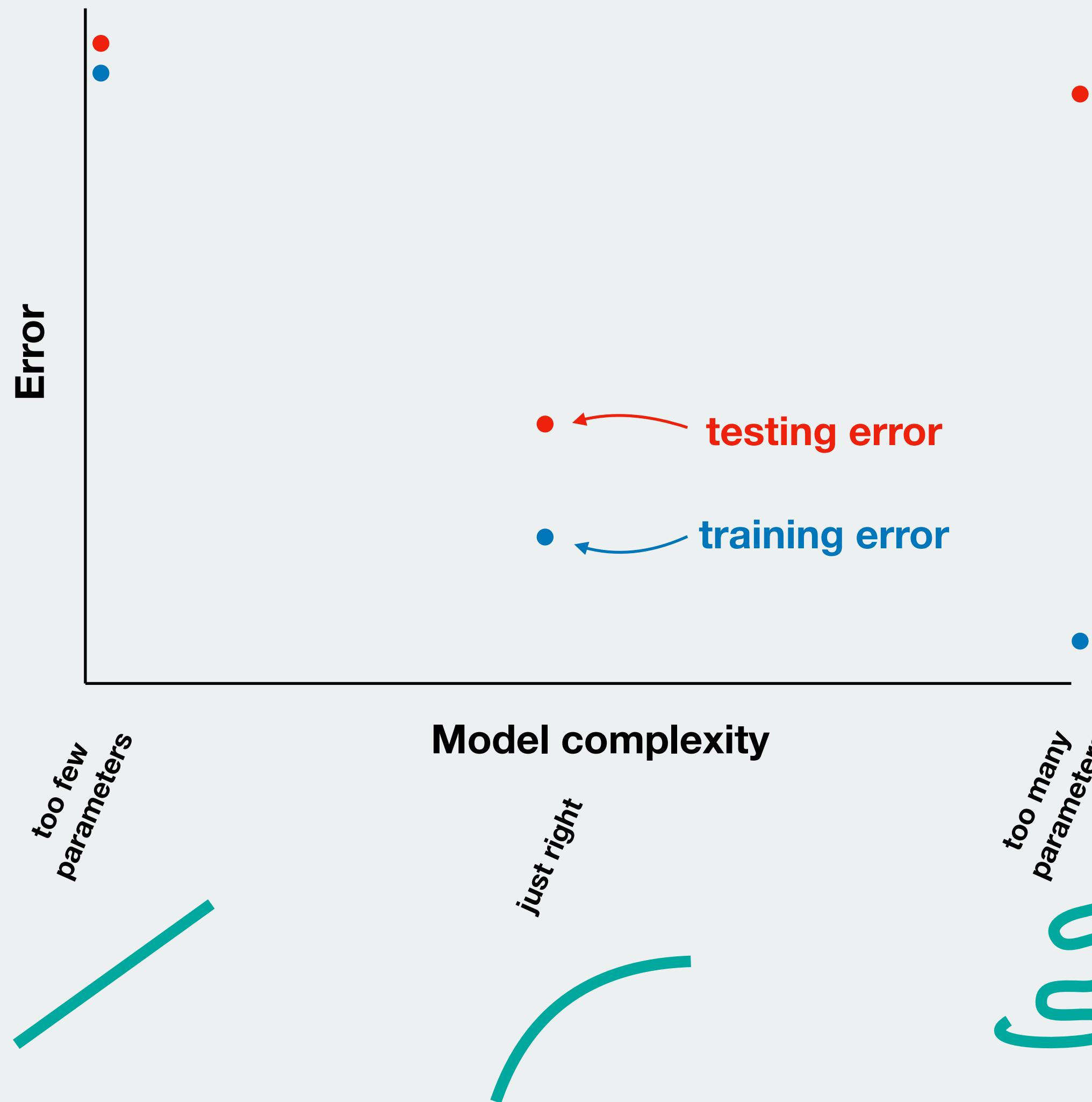
Avoiding underfitting and overfitting



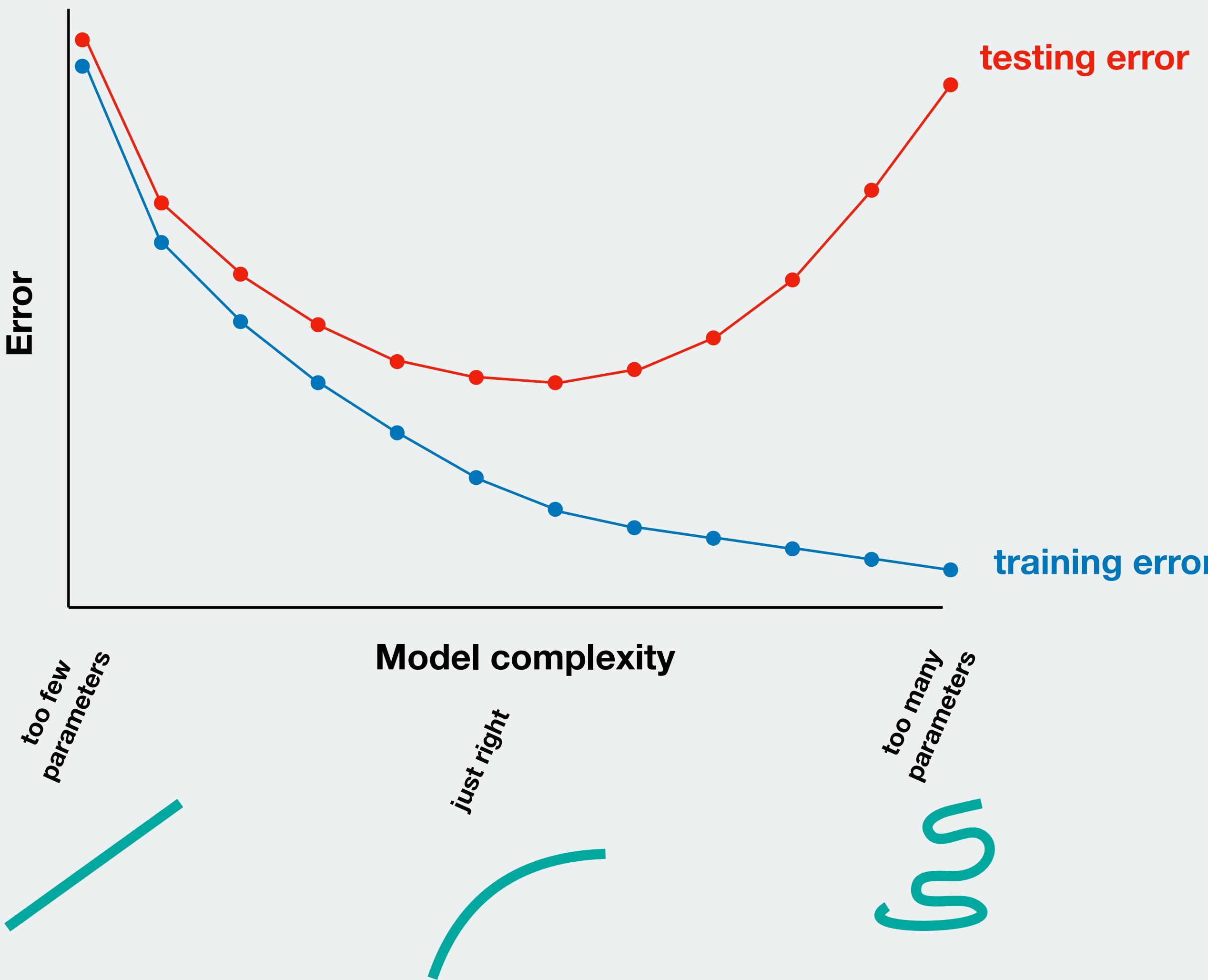
Avoiding underfitting and overfitting



Avoiding underfitting and overfitting

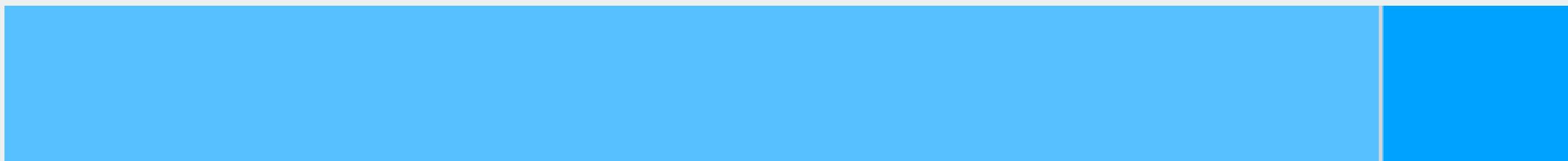


Avoiding underfitting and overfitting

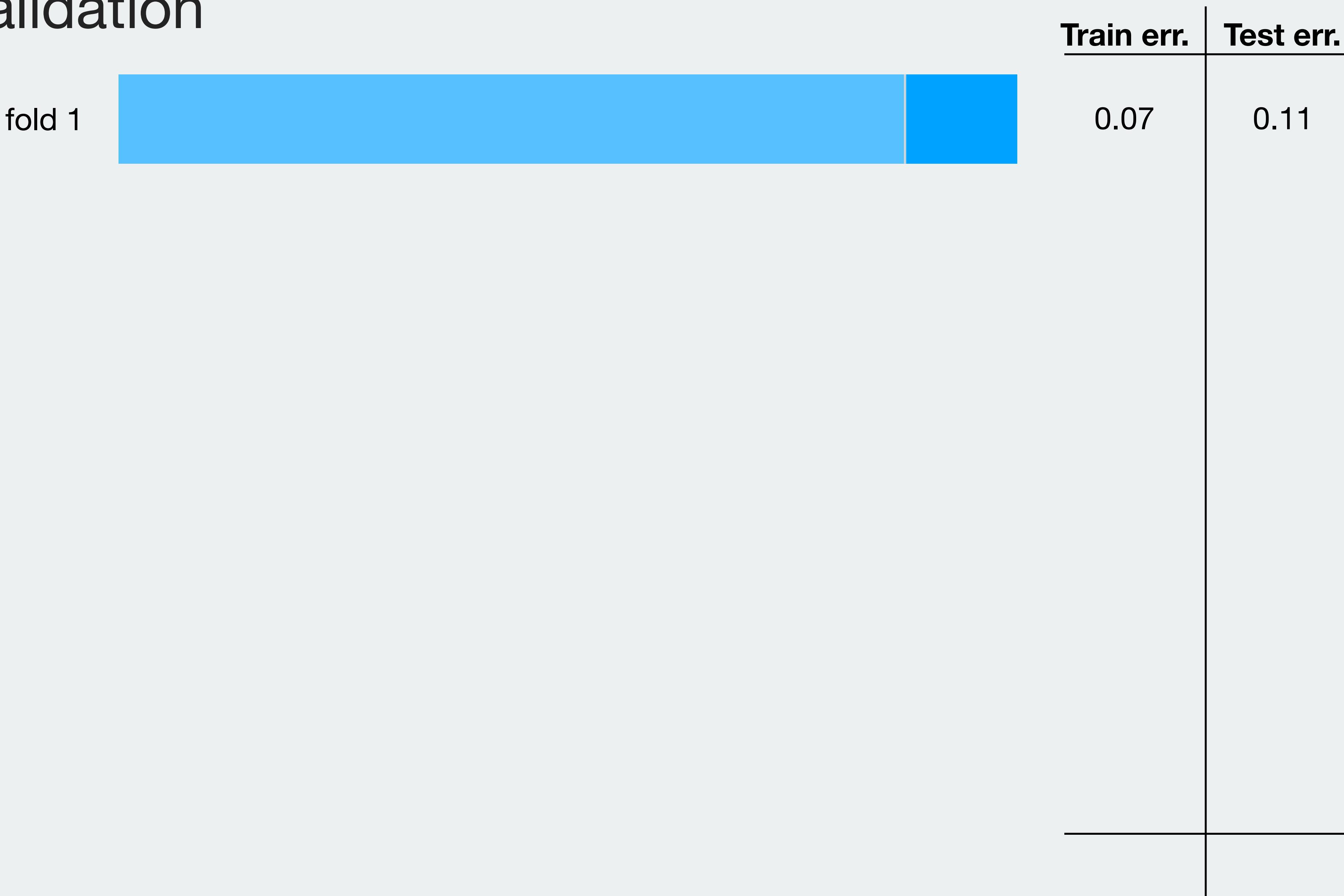


Cross validation

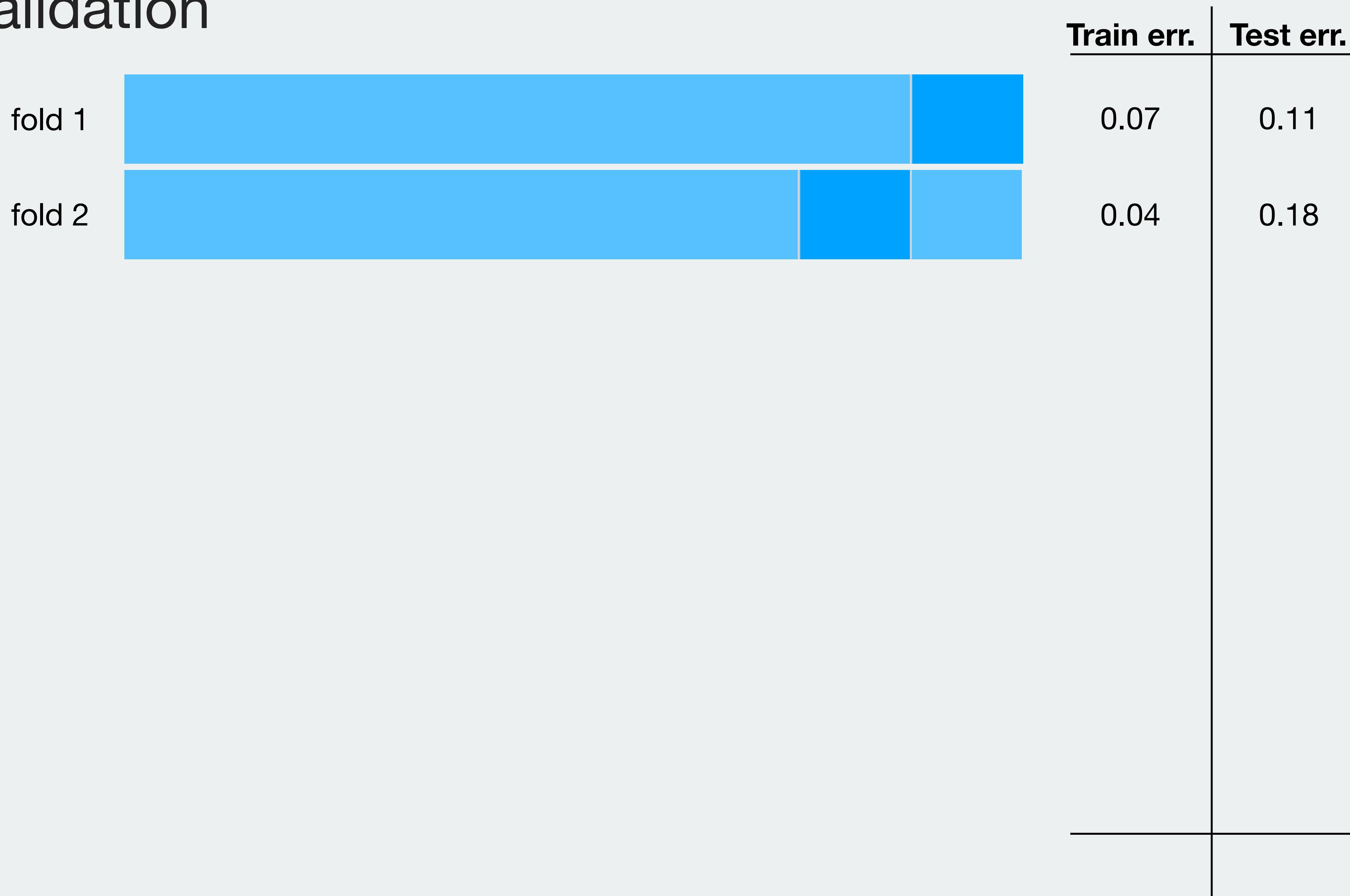
Cross validation



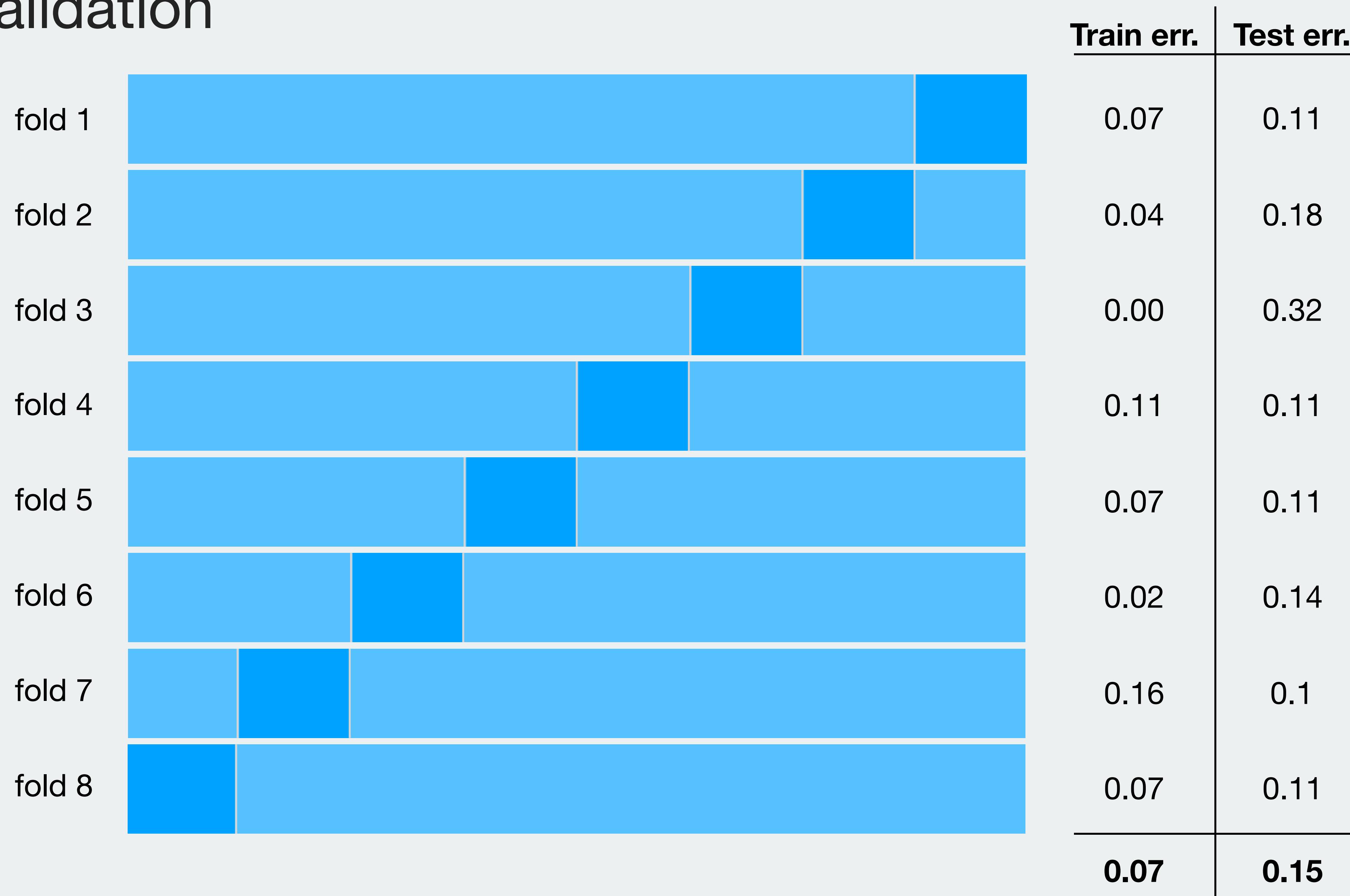
Cross validation



Cross validation

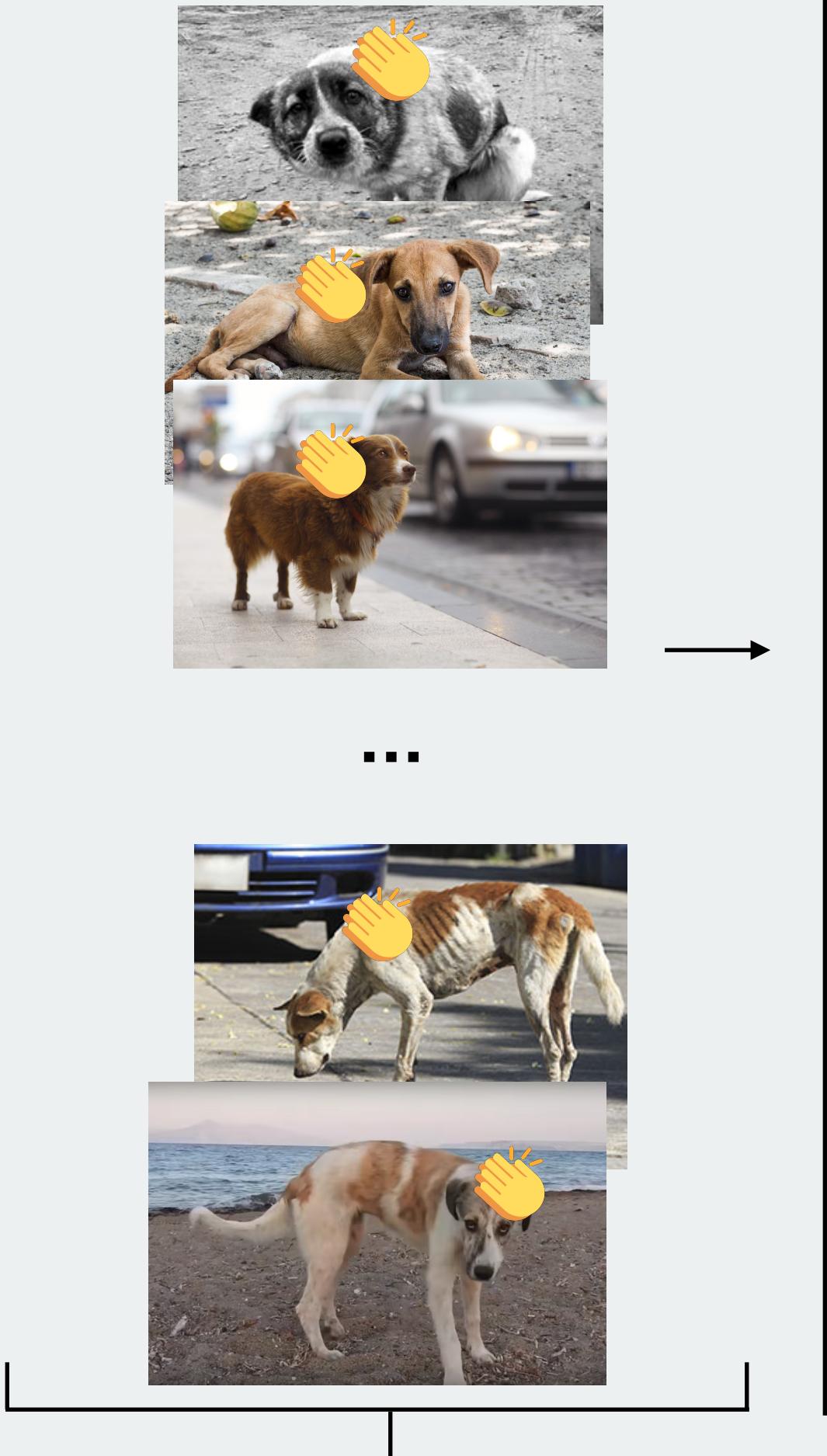


Cross validation



Model performance

Model performance



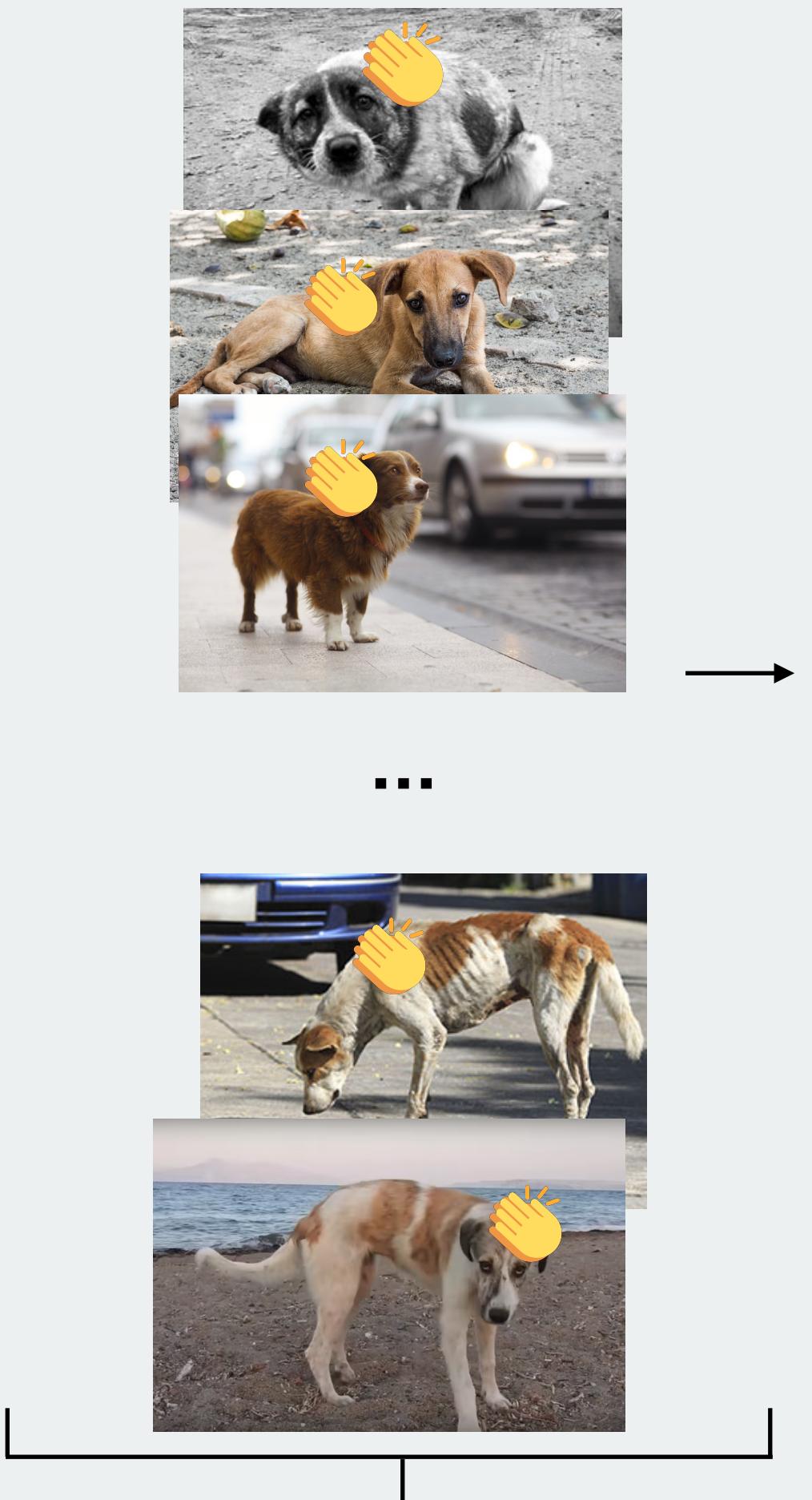
Many many experiments

bites

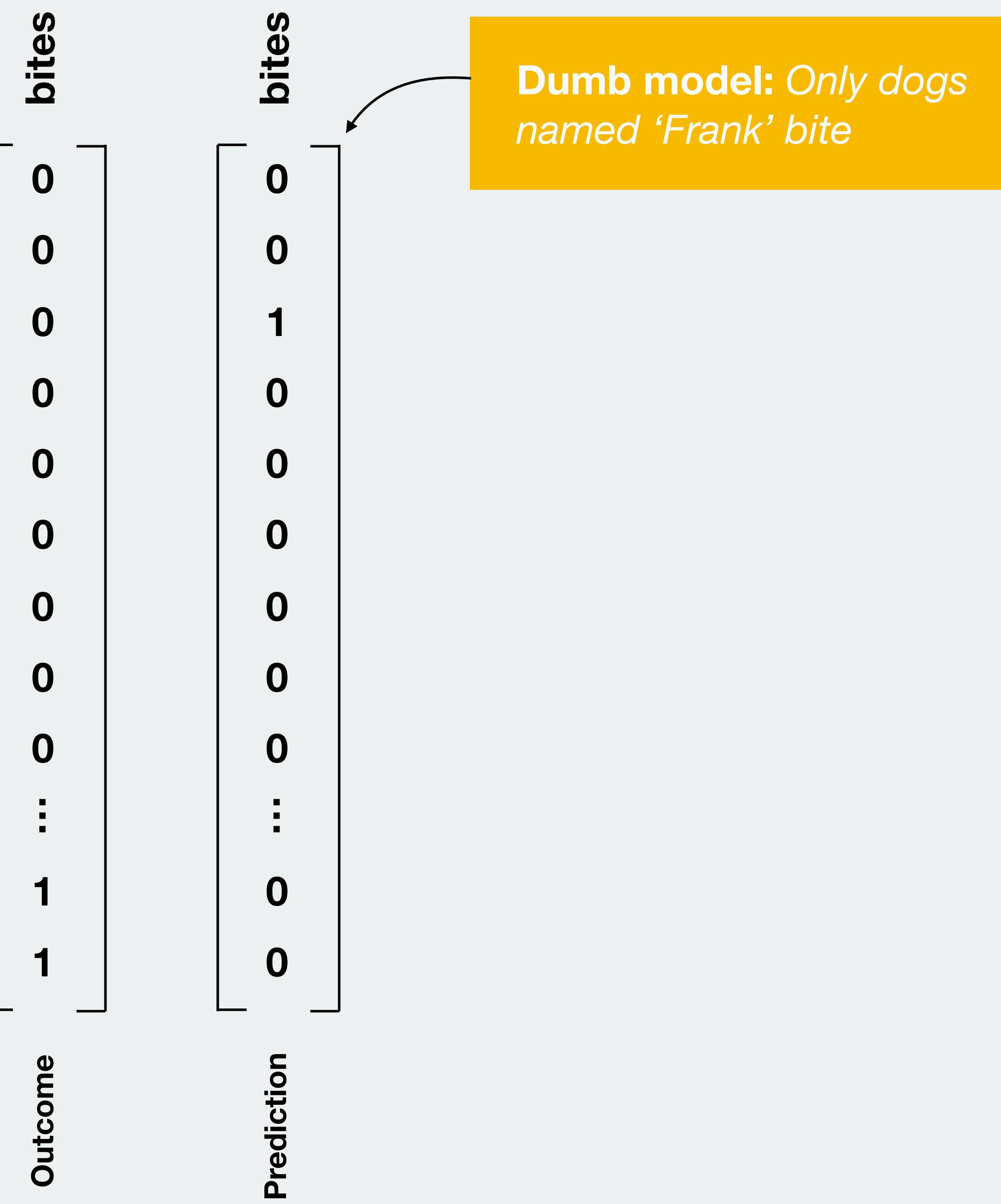
0
0
0
0
0
0
0
0
0
0
0
⋮
1
1

Outcome

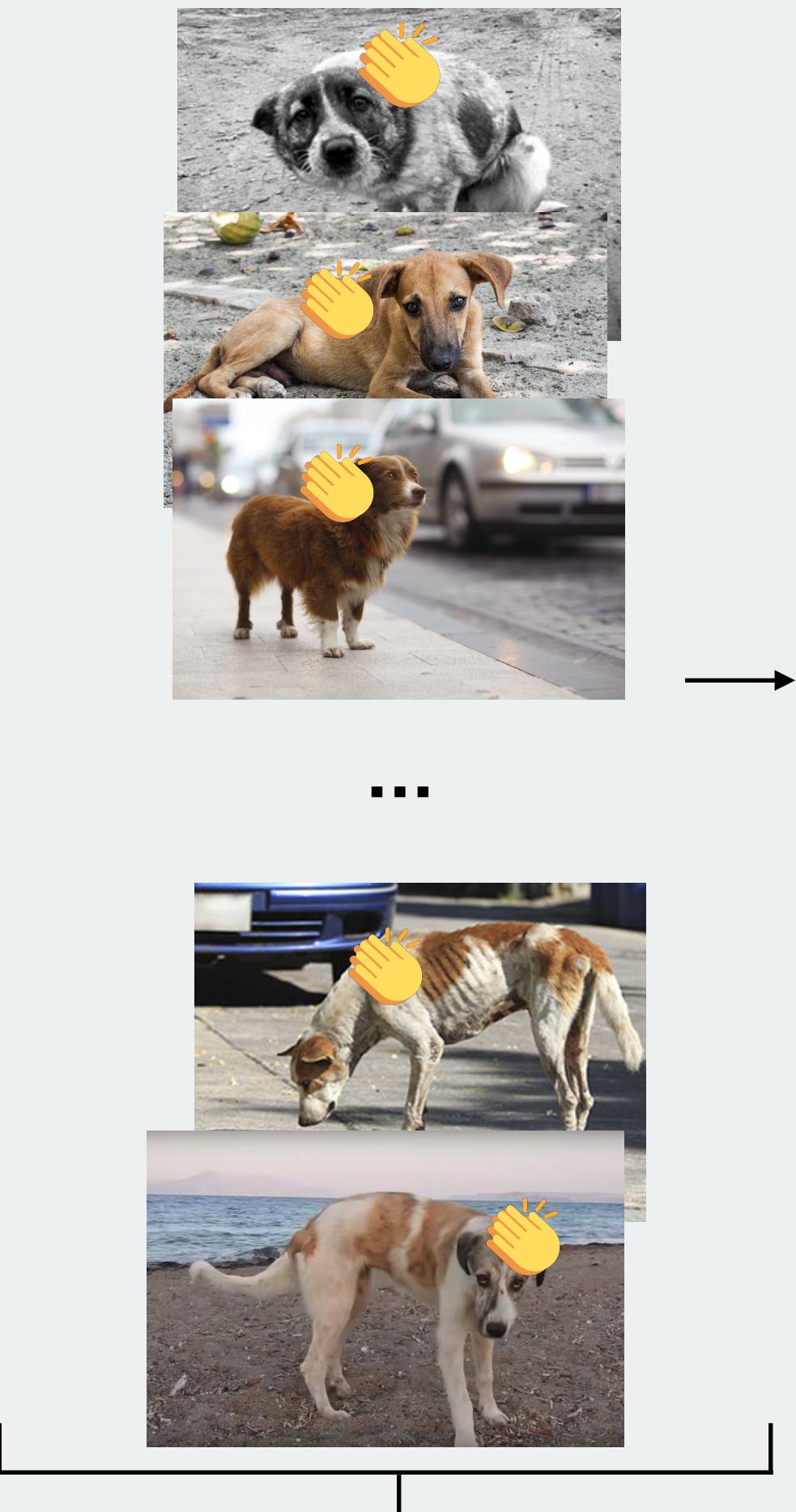
Model performance



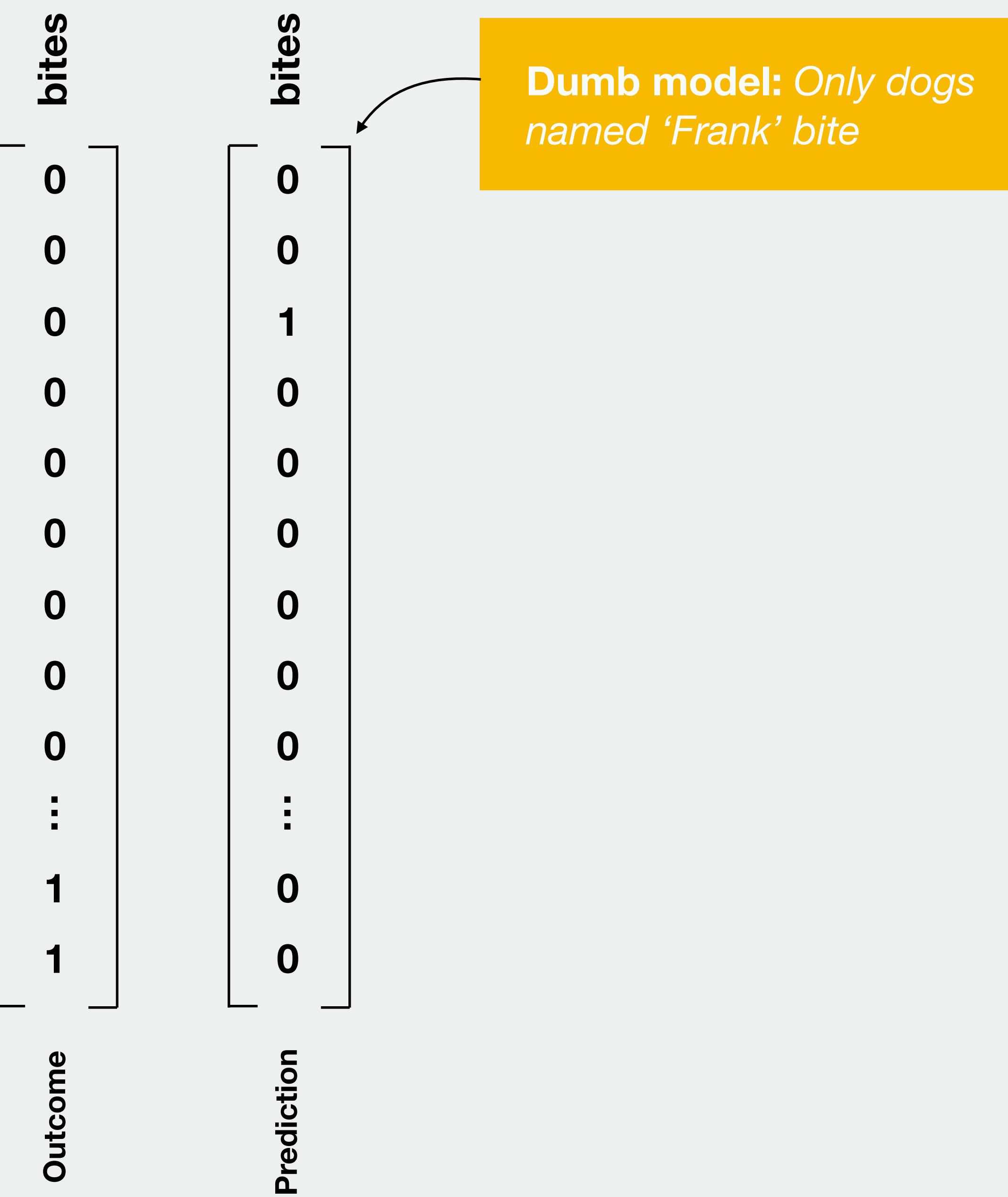
Many many experiments



Model performance



Many many experiments

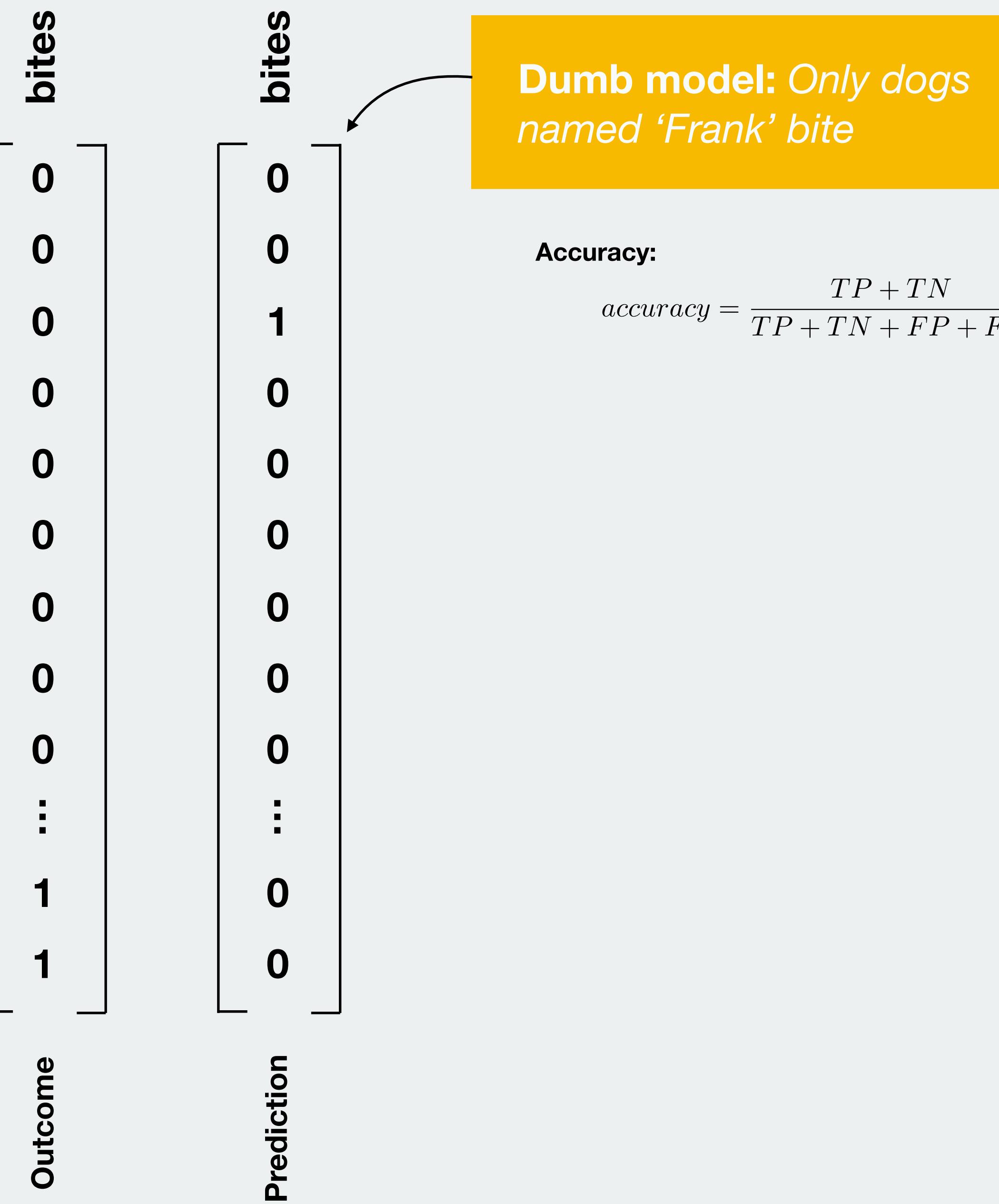


Dogs named Frank: 1%
Dogs that bite: 2%
Frank & bites: 0.02%

Model performance



Many many experiments



Dogs named Frank: 1%
Dogs that bite: 2%
Frank & bites: 0.02%

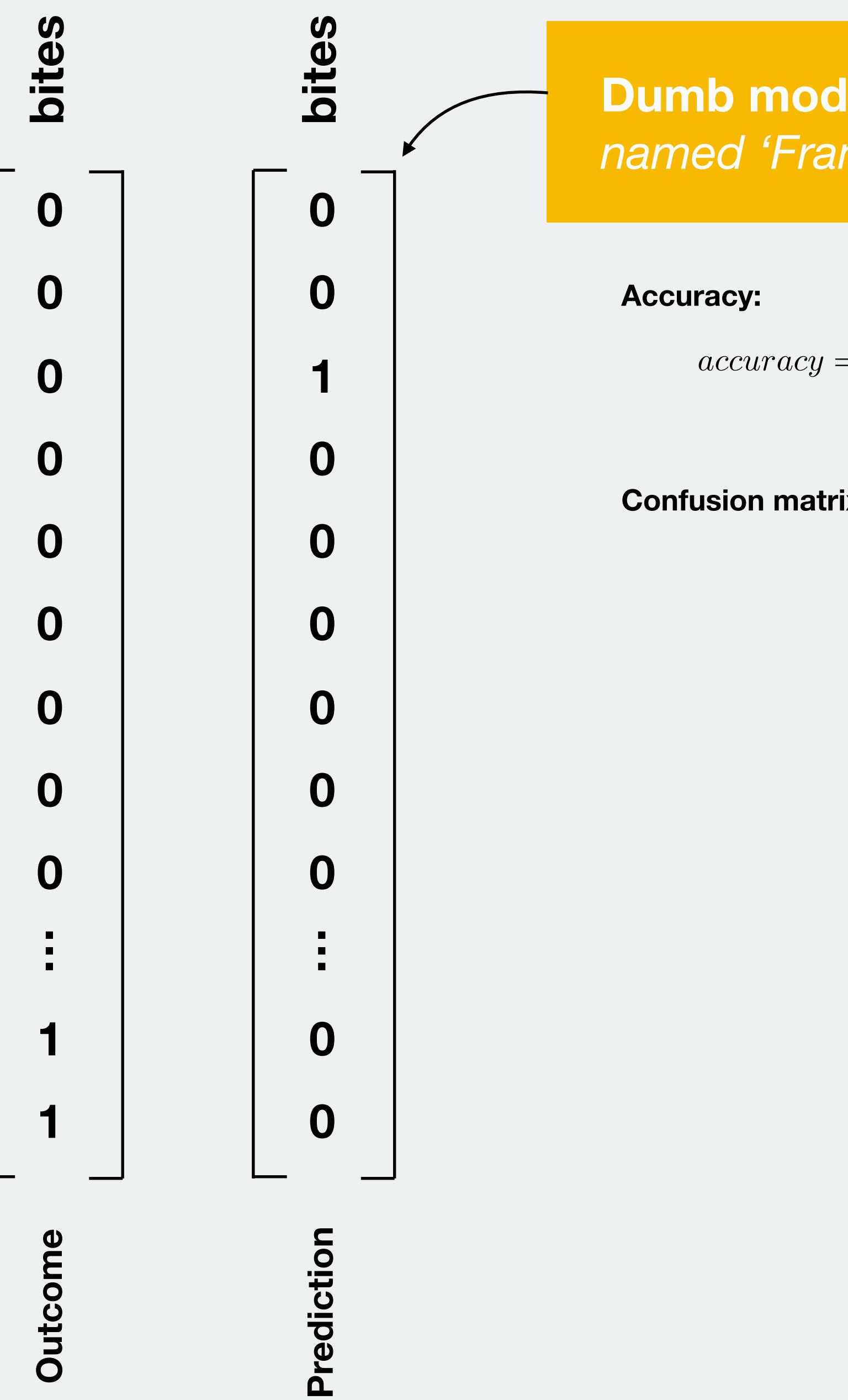
$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{200 + 970200}{1000000} = 97\%$$

high!

Model performance



Many many experiments



Dumb model: Only dogs named 'Frank' bite

Accuracy:

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{200 + 970200}{1000000} = 97\%$$

high!

Confusion matrix:

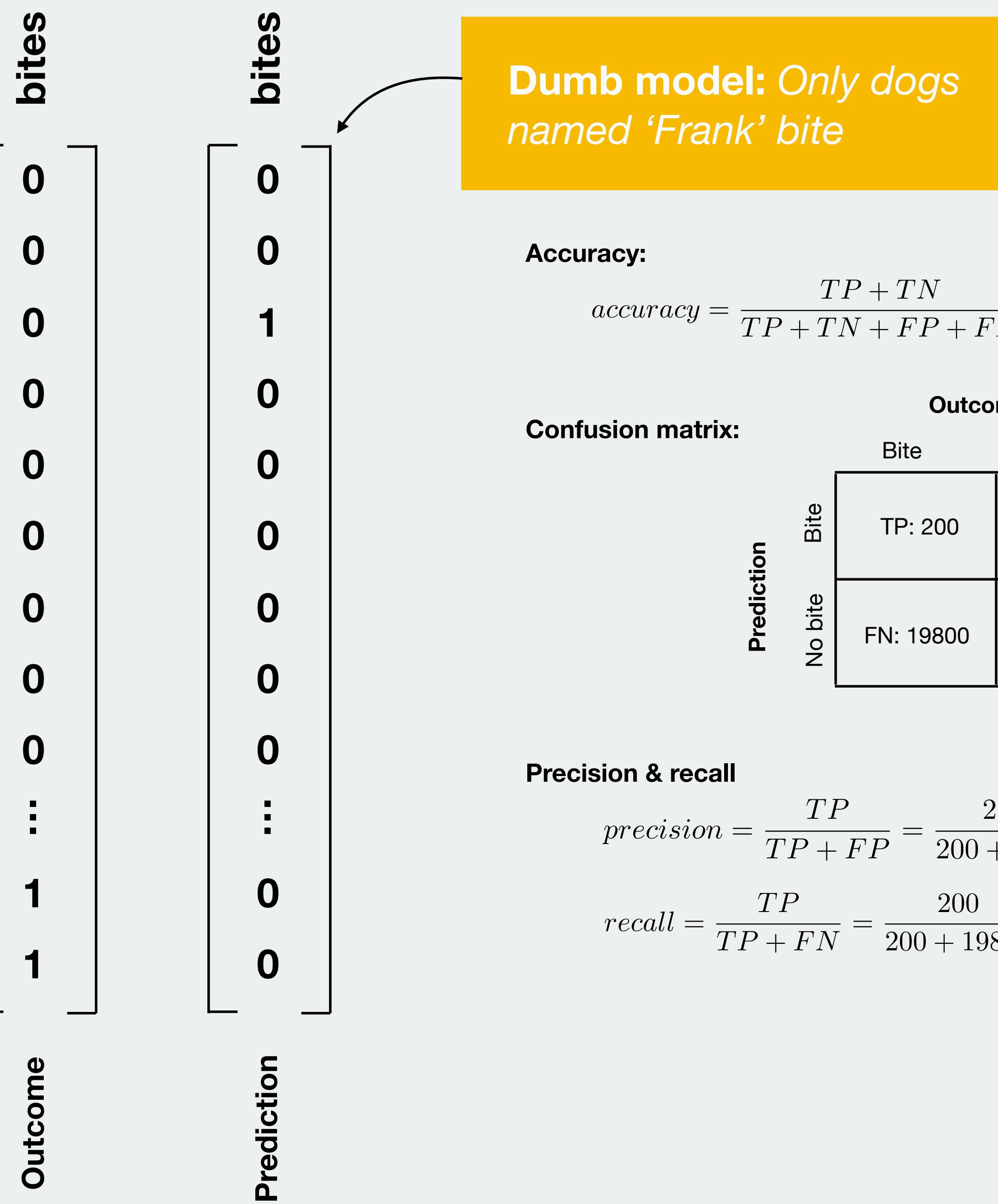
		Outcome	
		Bite	No bite
Prediction	Bite	TP: 200	FP: 9800
	No bite	FN: 19800	TN: 970200

Dogs named Frank: 1%
Dogs that bite: 2%
Frank & bites: 0.02%

Model performance



Many many experiments



Dogs named Frank: 1%
Dogs that bite: 2%
Frank & bites: 0.02%

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{200 + 970200}{1000000} = 97\%$$

high!

Confusion matrix

		Outcome
		Bite
Bite	Bite	TP: 200
	No bite	FP: 9800
No bite	Bite	FN: 19800
	No bite	TN: 970200

Precision & recall

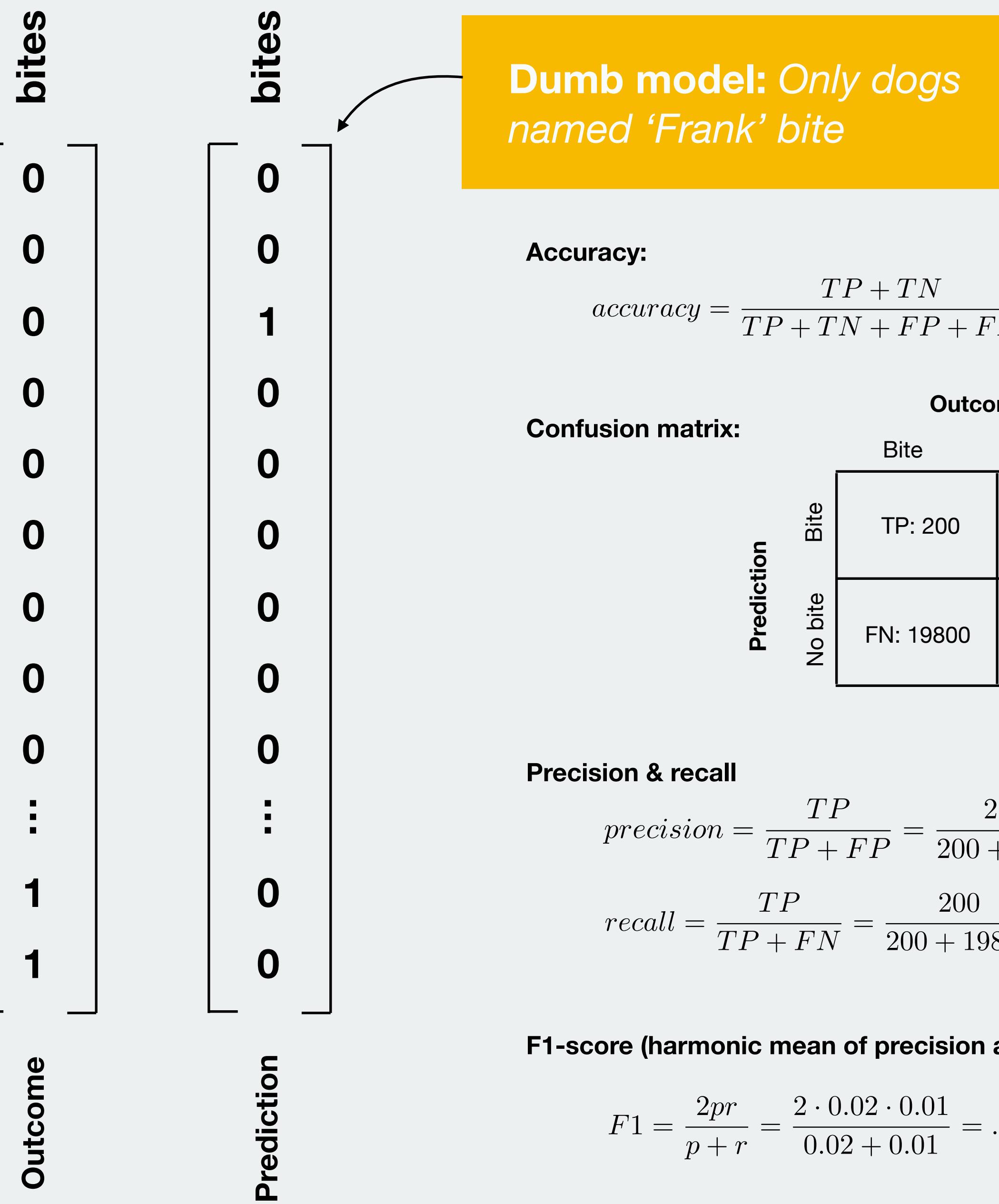
$$precision = \frac{TP}{TP + FP} = \frac{200}{200 + 9800} = 2\%$$

$$recall = \frac{TP}{TP + FN} = \frac{200}{200 + 19800} = 1\%$$

Model performance



Many many experiments



Dogs named Frank: 1%
Dogs that bite: 2%
Frank & bites: 0.02%

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{200 + 970200}{1000000} = 97\%$$

high!

Confusion matrix

	Bite	No bite
Bite	TP: 200	FP: 9800
No bite	FN: 19800	TN: 970200

Precision & recall

$$precision = \frac{TP}{TP + FP} = \frac{200}{200 + 9800} = 2\%$$

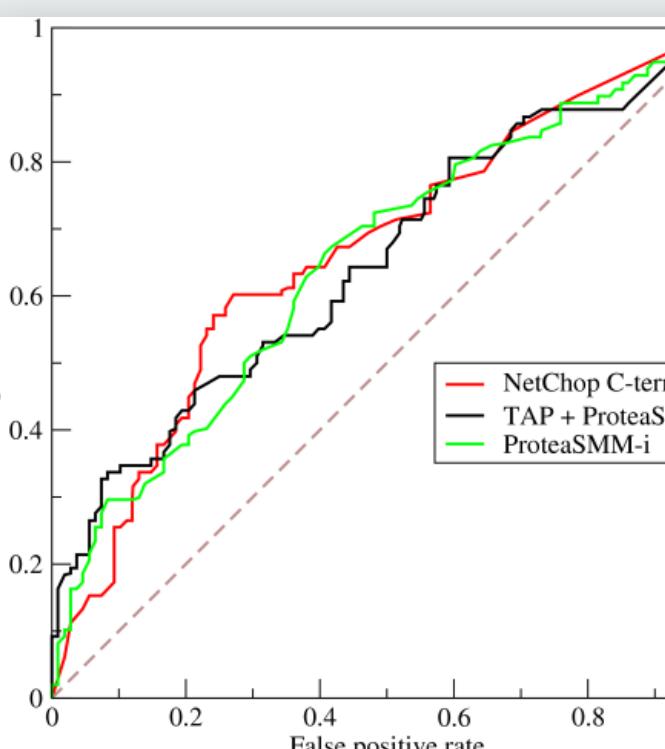
$$recall = \frac{TP}{TP + FN} = \frac{200}{200 + 19800} = 1\%$$

F1-score (harmonic mean of precision and recall):

$$F1 = \frac{2pr}{p+r} = \frac{2 \cdot 0.02 \cdot 0.01}{0.02 + 0.01} = .013$$

low!

Performance metrics summary

Metric	Description	Note
Accuracy	$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$ <p>“Fraction of classifications that are correct”</p>	Bad when classes are not balanced. Should always be compared with a balance baseline. Pretty easy to understand
Precision	$\text{precision} = \frac{TP}{TP + FP}$ <p>“What fraction of the dogs accused of biting, would actually bite?”</p>	Reveals a single useful aspect of a models performance
Recall	$\text{recall} = \frac{TP}{TP + FN}$ <p>“What fraction of dogs that would bite, were accused of biting?”</p>	Reveals a single useful aspect of a models performance
F1 score	$F1 = \frac{2pr}{p + r}$ <p>“Harmonic average of precision and recall”</p>	Mathematical construct = no intuitive interpretation “Honest” measure of performance
Receiver operating characteristic (ROC)	 <p>“Detection rate vs. false alarm rate for varying decision thresholds”</p>	Visual performance metric that gives an good impression of performance. If area under curve is > 0.5, classifier is better than random.

Code example

Code example

Import data

```
In [119]: import numpy as np
import sklearn

iris = sklearn.datasets.load_iris()
Last executed 2018-02-11 22:22:57 in 6ms
```

Code example

```
In [119]: import numpy as np
import sklearn

iris = sklearn.datasets.load_iris()
Last executed 2018-02-11 22:22:57 in 6ms
```

Visualize data

```
In [120]: iris['data']
Last executed 2018-02-11 22:23:22 in 27ms
```

```
Out[120]: array([[ 5.1,  3.5,  1.4,  0.2],
       [ 4.9,  3. ,  1.4,  0.2],
       [ 4.7,  3.2,  1.3,  0.2],
       [ 4.6,  3.1,  1.5,  0.2],
       [ 5. ,  3.6,  1.4,  0.2],
       [ 5.4,  3.9,  1.7,  0.4],
       [ 4.6,  3.4,  1.4,  0.3],
       [ 5. ,  3.4,  1.5,  0.2],
       [ 4.4,  2.9,  1.4,  0.2],
       [ 4.9,  3.1,  1.5,  0.1],
       [ 5.4,  3.7,  1.5,  0.2],
       [ 4.8,  3.4,  1.6,  0.2],
       [ 4.8,  3. ,  1.4,  0.1],
       [ 4.3,  3. ,  1.1,  0.1],
       [ 5.8,  4. ,  1.2,  0.2],
       [ 5.7,  4.4,  1.5,  0.4],
       [ 5.4,  3.9,  1.3,  0.4],
       [ 5.1,  3.5,  1.4,  0.3],
       [ 5.7,  3.8,  1.7,  0.3],
       [ 5.1,  3.8,  1.5,  0.3],
```

Code example

```
In [119]: import numpy as np
import sklearn

iris = sklearn.datasets.load_iris()

Last executed 2018-02-11 22:22:57 in 6ms
```



```
In [120]: iris['data']

Last executed 2018-02-11 22:23:22 in 27ms
```



```
Out[120]: array([[ 5.1,  3.5,  1.4,  0.2],
       [ 4.9,  3. ,  1.4,  0.2],
       [ 4.7,  3.2,  1.3,  0.2],
       [ 4.6,  3.1,  1.5,  0.2],
       [ 5. ,  3.6,  1.4,  0.2],
       [ 5.4,  3.9,  1.7,  0.4],
       [ 4.6,  3.4,  1.4,  0.3],
       [ 5. ,  3.4,  1.5,  0.2],
       [ 4.4,  2.9,  1.4,  0.2],
       [ 4.9,  3.1,  1.5,  0.1],
       [ 5.4,  3.7,  1.5,  0.2],
       [ 4.8,  3.4,  1.6,  0.2],
       [ 4.8,  3. ,  1.4,  0.1],
       [ 4.3,  3. ,  1.1,  0.1],
       [ 5.8,  4. ,  1.2,  0.2],
       [ 5.7,  4.4,  1.5,  0.4],
       [ 5.4,  3.9,  1.3,  0.4],
       [ 5.1,  3.5,  1.4,  0.3],
       [ 5.7,  3.8,  1.7,  0.3],
       [ 5.1,  3.8,  1.5,  0.3],
```

Visualize data

Code example

Implement model

```
In [157]: from sklearn.ensemble import RandomForestClassifier

N = iris['data'].shape[0]

train_indices = range(0, N, 2)
test_indices = range(1, N, 2)

model = RandomForestClassifier()
model.fit(iris['data'][train_indices, :], iris['target'][train_indices])

y_true = iris['target'][test_indices]
y_pred = model.predict(iris['data'][test_indices, :])

print "Accuracy:", sum(y_pred == y_true) * 1. / len(y_true)
```

Code example

Implement model

```
In [157]: from sklearn.ensemble import RandomForestClassifier

N = iris['data'].shape[0]

train_indices = range(0, N, 2)
test_indices = range(1, N, 2)

model = RandomForestClassifier()
model.fit(iris['data'][train_indices, :], iris['target'][train_indices])

y_true = iris['target'][test_indices]
y_pred = model.predict(iris['data'][test_indices, :])

print "Accuracy:", sum(y_pred == y_true) * 1. / len(y_true)
```

Code example

Implement model

```
In [157]: from sklearn.ensemble import RandomForestClassifier

N = iris['data'].shape[0]

train_indices = range(0, N, 2)
test_indices = range(1, N, 2)

model = RandomForestClassifier()
model.fit(iris['data'][train_indices, :], iris['target'][train_indices])

y_true = iris['target'][test_indices]
y_pred = model.predict(iris['data'][test_indices, :])

print "Accuracy:", sum(y_pred == y_true) * 1. / len(y_true)
```

Code example

Implement model

```
In [157]: from sklearn.ensemble import RandomForestClassifier

N = iris['data'].shape[0]

train_indices = range(0, N, 2)
test_indices = range(1, N, 2)

model = RandomForestClassifier()
model.fit(iris['data'][train_indices, :], iris['target'][train_indices])

y_true = iris['target'][test_indices]
y_pred = model.predict(iris['data'][test_indices, :])

print "Accuracy:", sum(y_pred == y_true) * 1. / len(y_true)
```

Last executed 2018-02-11 22:31:30 in 52ms

Accuracy: 0.96

```
In [160]: 1 train_indices[:10]
Last executed 2018-02-11 22:42:42 in 5ms
Out[160]: [0, 2, 4, 6, 8, 10, 12, 14, 16, 18]

In [161]: 1 test_indices[:10]
Last executed 2018-02-11 22:42:58 in 5ms
Out[161]: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

Code example

Implement model

```
In [157]: from sklearn.ensemble import RandomForestClassifier

N = iris['data'].shape[0]

train_indices = range(0, N, 2)
test_indices = range(1, N, 2)

model = RandomForestClassifier()
model.fit(iris['data'][train_indices, :], iris['target'][train_indices])

y_true = iris['target'][test_indices]
y_pred = model.predict(iris['data'][test_indices, :])

print "Accuracy:", sum(y_pred == y_true) * 1. / len(y_true)
```

Code example

Implement model

```
In [157]: from sklearn.ensemble import RandomForestClassifier

N = iris['data'].shape[0]

train_indices = range(0, N, 2)
test_indices = range(1, N, 2)

model = RandomForestClassifier()
model.fit(iris['data'][train_indices, :], iris['target'][train_indices])

y_true = iris['target'][test_indices]
y_pred = model.predict(iris['data'][test_indices, :])

print "Accuracy:", sum(y_pred == y_true) * 1. / len(y_true)
```

Last executed 2018-02-11 22:31:30 in 52ms

Accuracy: 0.96

Code example

Implement model

```
In [157]: from sklearn.ensemble import RandomForestClassifier

N = iris['data'].shape[0]

train_indices = range(0, N, 2)
test_indices = range(1, N, 2)

model = RandomForestClassifier()
model.fit(iris['data'][train_indices, :], iris['target'][train_indices])

y_true = iris['target'][test_indices]
y_pred = model.predict(iris['data'][test_indices, :])

print "Accuracy:", sum(y_pred == y_true) * 1. / len(y_true)
```

Code example

Implement model

```
In [157]: from sklearn.ensemble import RandomForestClassifier

N = iris['data'].shape[0]

train_indices = range(0, N, 2)
test_indices = range(1, N, 2)

model = RandomForestClassifier()
model.fit(iris['data'][train_indices, :], iris['target'][train_indices])

y_true = iris['target'][test_indices]
y_pred = model.predict(iris['data'][test_indices, :])

print "Accuracy:", sum(y_pred == y_true) * 1. / len(y_true)
```

Last executed 2018-02-11 22:31:30 in 52ms

Accuracy: 0.96

Code example

Implement model

```
In [157]: from sklearn.ensemble import RandomForestClassifier

N = iris['data'].shape[0]

train_indices = range(0, N, 2)
test_indices = range(1, N, 2)

model = RandomForestClassifier()
model.fit(iris['data'][train_indices, :], iris['target'][train_indices])

y_true = iris['target'][test_indices]
y_pred = model.predict(iris['data'][test_indices, :])

print "Accuracy:", sum(y_pred == y_true) * 1. / len(y_true)
```

Last executed 2018-02-11 22:31:30 in 52ms

Accuracy: 0.96

Code example

Implement model

```
In [157]: from sklearn.ensemble import RandomForestClassifier

N = iris['data'].shape[0]

train_indices = range(0, N, 2)
test_indices = range(1, N, 2)

model = RandomForestClassifier()
model.fit(iris['data'][train_indices, :], iris['target'][train_indices])

y_true = iris['target'][test_indices]
y_pred = model.predict(iris['data'][test_indices, :])

print "Accuracy:", sum(y_pred == y_true) * 1. / len(y_true)
```

Last executed 2018-02-11 22:31:30 in 52ms

Accuracy: 0.96

Code example

Implement model

```
In [157]: from sklearn.ensemble import RandomForestClassifier

N = iris['data'].shape[0]

train_indices = range(0, N, 2)
test_indices = range(1, N, 2)

model = RandomForestClassifier()
model.fit(iris['data'][train_indices, :], iris['target'][train_indices])

y_true = iris['target'][test_indices]
y_pred = model.predict(iris['data'][test_indices, :])

print "Accuracy:", sum(y_pred == y_true) * 1. / len(y_true)
```

Last executed 2018-02-11 22:31:30 in 52ms

Accuracy: 0.96

Code example

Implement model

```
In [157]: from sklearn.ensemble import RandomForestClassifier

N = iris['data'].shape[0]

train_indices = range(0, N, 2)
test_indices = range(1, N, 2)

model = RandomForestClassifier()
model.fit(iris['data'][train_indices, :], iris['target'][train_indices])

y_true = iris['target'][test_indices]
y_pred = model.predict(iris['data'][test_indices, :])

print "Accuracy:", sum(y_pred == y_true) * 1. / len(y_true)
```

Last executed 2018-02-11 22:31:30 in 52ms

Accuracy: 0.96

Code example

Implement model

```
In [157]: from sklearn.ensemble import RandomForestClassifier

N = iris['data'].shape[0]

train_indices = range(0, N, 2)
test_indices = range(1, N, 2)

model = RandomForestClassifier()
model.fit(iris['data'][train_indices, :], iris['target'][train_indices])

y_true = iris['target'][test_indices]
y_pred = model.predict(iris['data'][test_indices, :])

print "Accuracy:", sum(y_pred == y_true) * 1. / len(y_true)
```

Last executed 2018-02-11 22:31:30 in 52ms

Accuracy: 0.96

Code example

Implement model

```
In [157]: from sklearn.ensemble import RandomForestClassifier

N = iris['data'].shape[0]

train_indices = range(0, N, 2)
test_indices = range(1, N, 2)

model = RandomForestClassifier()
model.fit(iris['data'][train_indices, :], iris['target'][train_indices])

y_true = iris['target'][test_indices]
y_pred = model.predict(iris['data'][test_indices, :])

print "Accuracy:", sum(y_pred == y_true) * 1. / len(y_true)
```

Last executed 2018-02-11 22:31:30 in 52ms

Code example

Implement model

```
In [157]: from sklearn.ensemble import RandomForestClassifier

N = iris['data'].shape[0]

train_indices = range(0, N, 2)
test_indices = range(1, N, 2)

model = RandomForestClassifier()
model.fit(iris['data'][train_indices, :], iris['target'][train_indices])

y_true = iris['target'][test_indices]
y_pred = model.predict(iris['data'][test_indices, :])

print "Accuracy:", sum(y_pred == y_true) * 1. / len(y_true)
```

Code example

Implement model

```
In [157]: from sklearn.ensemble import RandomForestClassifier

N = iris['data'].shape[0]

train_indices = range(0, N, 2)
test_indices = range(1, N, 2)

model = RandomForestClassifier()
model.fit(iris['data'][train_indices, :], iris['target'][train_indices])

y_true = iris['target'][test_indices]
y_pred = model.predict(iris['data'][test_indices, :])

print "Accuracy:", sum(y_pred == y_true) * 1. / len(y_true)
```

Last executed 2018-02-11 22:31:30 in 52ms

Accuracy: 0.96

```
In [164]: 1 y_pred == y_true
Last executed 2018-02-11 22:57:16 in 5ms

Out[164]: array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
       True,  True,  True,  True,  True,  True,  True,  True,  True,
       True,  True,  True,  True,  True,  True,  True,  True,  True,
       True,  True,  True,  True,  True,  True,  True,  True,  True,
       True,  True, False,  True,  True,  True,  True,  True,  True,
       True,  True,  True,  True,  True,  True,  True,  True,  True,
       True,  True,  True,  True,  True,  False,  True,  True,  True,
       True,  True,  True, False,  True,  True,  True,  True,  True,
       True,  True,  True], dtype=bool)
```

Code example

```
In [119]: import numpy as np
import sklearn

iris = sklearn.datasets.load_iris()
Last executed 2018-02-11 22:22:57 in 6ms

In [120]: iris['data']
Last executed 2018-02-11 22:23:22 in 27ms
Out[120]: array([[ 5.1,  3.5,  1.4,  0.2],
       [ 4.9,  3. ,  1.4,  0.2],
       [ 4.7,  3.2,  1.3,  0.2],
       [ 4.6,  3.1,  1.5,  0.2],
       [ 5. ,  3.6,  1.4,  0.2],
       [ 5.4,  3.9,  1.7,  0.4],
       [ 4.6,  3.4,  1.4,  0.3],
       [ 5. ,  3.4,  1.5,  0.2],
       [ 4.4,  2.9,  1.4,  0.2],
       [ 4.9,  3.1,  1.5,  0.1],
       [ 5.4,  3.7,  1.5,  0.2],
       [ 4.8,  3.4,  1.6,  0.2],
       [ 4.8,  3. ,  1.4,  0.1],
       [ 4.3,  3. ,  1.1,  0.1],
       [ 5.8,  4. ,  1.2,  0.2],
       [ 5.7,  4.4,  1.5,  0.4],
       [ 5.4,  3.9,  1.3,  0.4],
       [ 5.1,  3.5,  1.4,  0.3],
       [ 5.7,  3.8,  1.7,  0.3],
       [ 5.1,  3.8,  1.5,  0.3],
```

```
In [122]: iris['target']
Last executed 2018-02-11 22:24:05 in 9ms
```

```
Out[122]: array([0,
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
```

Implement model

```
In [157]: from sklearn.ensemble import RandomForestClassifier

N = iris['data'].shape[0]

train_indices = range(0, N, 2)
test_indices = range(1, N, 2)

model = RandomForestClassifier()
model.fit(iris['data'][train_indices, :], iris['target'][train_indices])

y_true = iris['target'][test_indices]
y_pred = model.predict(iris['data'][test_indices, :])

print "Accuracy:", sum(y_pred == y_true) * 1. / len(y_true)
```

```
Last executed 2018-02-11 22:31:30 in 52ms
```

Accuracy: 0.96

```
In [165]: 1 sum(y_pred == y_true)
Last executed 2018-02-11 22:58:06 in 8ms
```

```
Out[165]: 72
```

Code example

Implement model

```
In [157]: from sklearn.ensemble import RandomForestClassifier

N = iris['data'].shape[0]

train_indices = range(0, N, 2)
test_indices = range(1, N, 2)

model = RandomForestClassifier()
model.fit(iris['data'][train_indices, :], iris['target'][train_indices])

y_true = iris['target'][test_indices]
y_pred = model.predict(iris['data'][test_indices, :])

print "Accuracy:", sum(y_pred == y_true) * 1. / len(y_true)
```

```
In [165]: 1 sum(y_pred == y_true)
Out[165]: 72
```

```
In [166]: 1 len(y_true)
```

1	len(y_true)
---	-------------

Last executed 2018-02-1

```
Out[166]: 75
```

Code example

Implement model

```
In [157]: from sklearn.ensemble import RandomForestClassifier

N = iris['data'].shape[0]

train_indices = range(0, N, 2)
test_indices = range(1, N, 2)

model = RandomForestClassifier()
model.fit(iris['data'][train_indices, :], iris['target'][train_indices])

y_true = iris['target'][test_indices]
y_pred = model.predict(iris['data'][test_indices, :])

print "Accuracy:", sum(y_pred == y_true) * 1. / len(y_true)
```