Computational Analysis of Big Data

Week 4

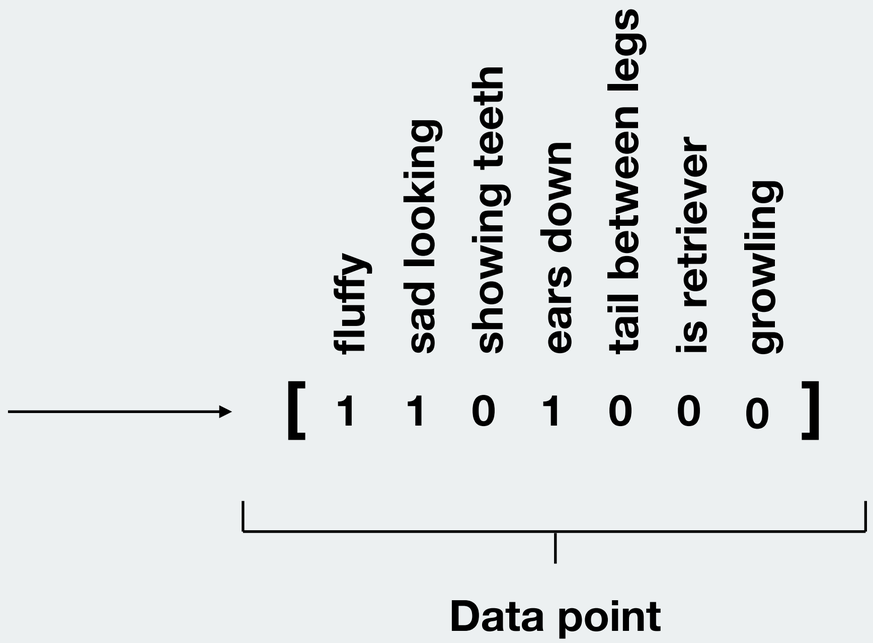
Machine Learning 1

Canonical example

Canonical example

Canonical example

Dog



Data point

Canonical example

[1 1 0 1 0 0 0]

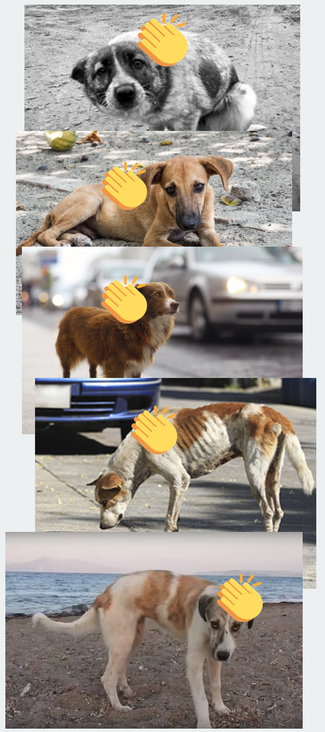
fluffy, sad looking, showing teeth, ears down, tail between legs, is retriever, growling

[0]

bites

Dog **Data point** **Outcome**

Canonical example



fluffy	sad looking	showing teeth	ears down	tail between legs	is retriever	growling
1	1	0	0	0	1	0
0	0	0	1	0	1	0
1	1	0	0	0	1	0
0	0	1	0	0	0	1
1	0	1	0	0	0	1

bites
0
0
0
1
1

Experiments

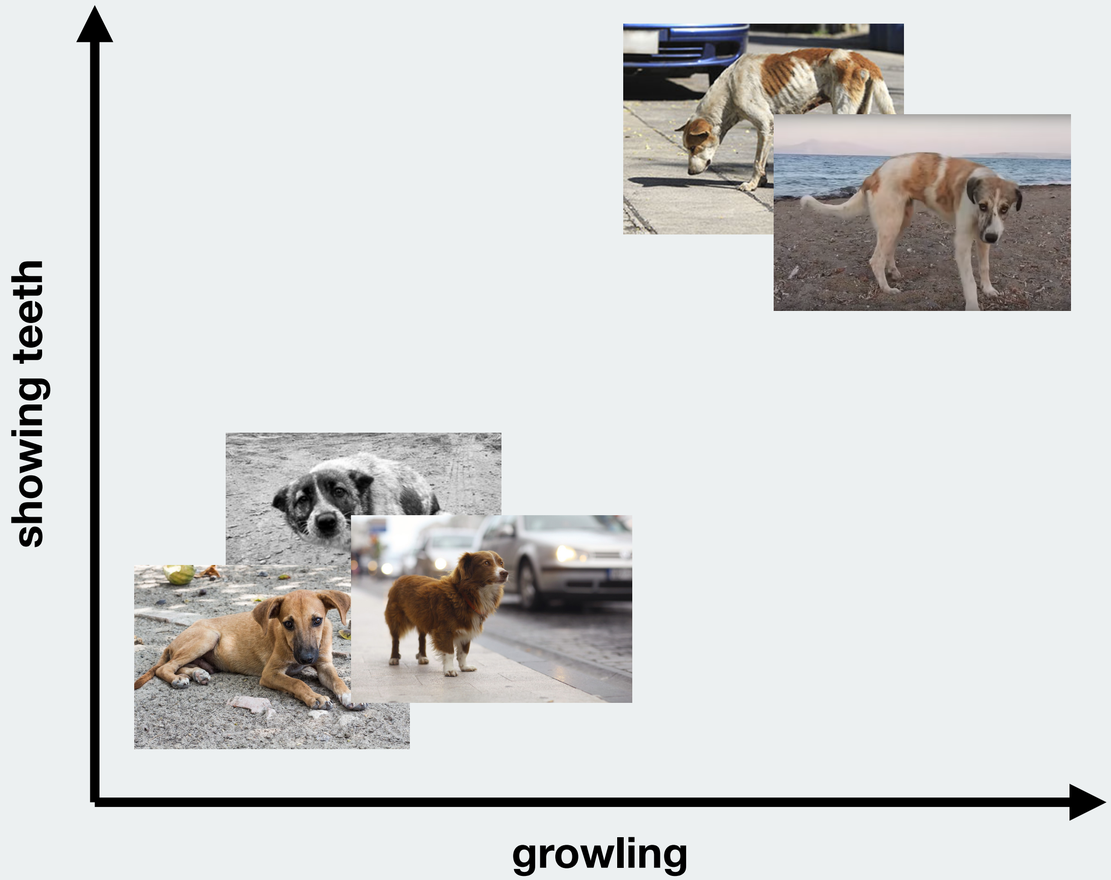
Data points

X

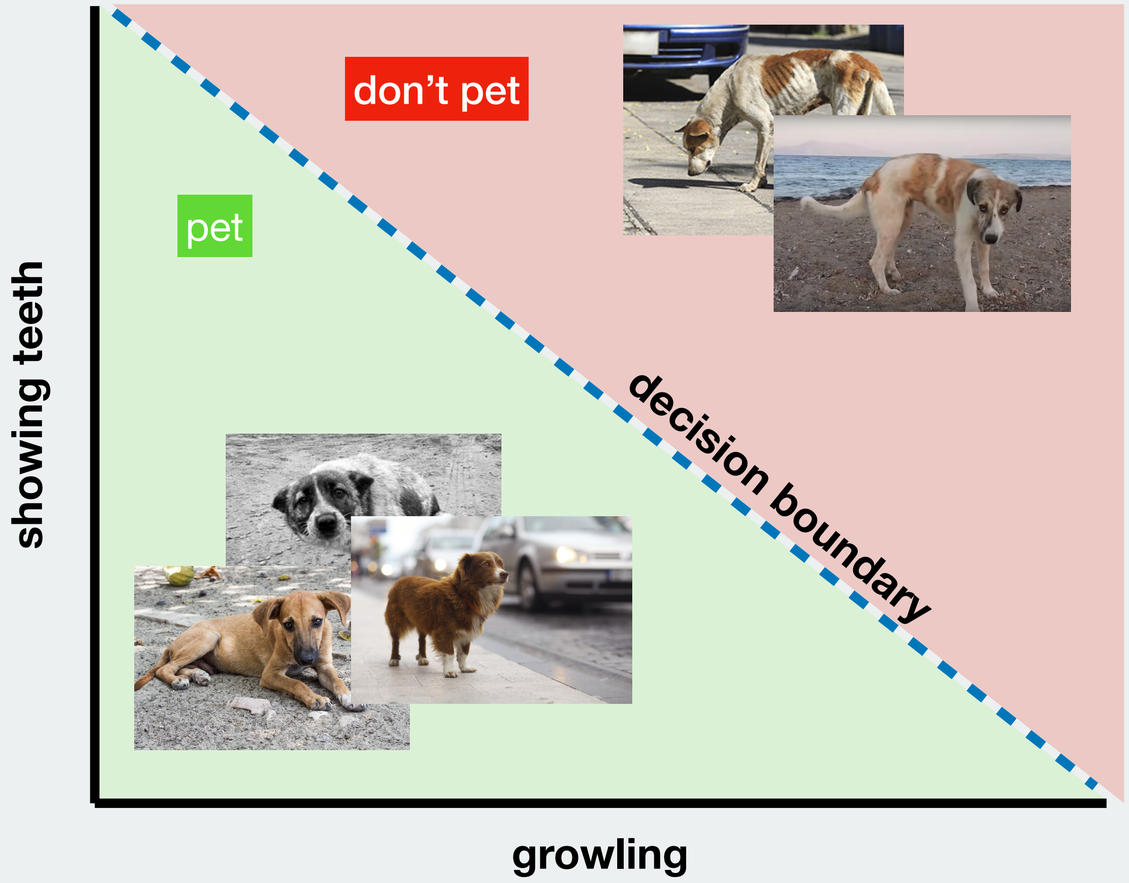
Outcomes

y

Canonical example



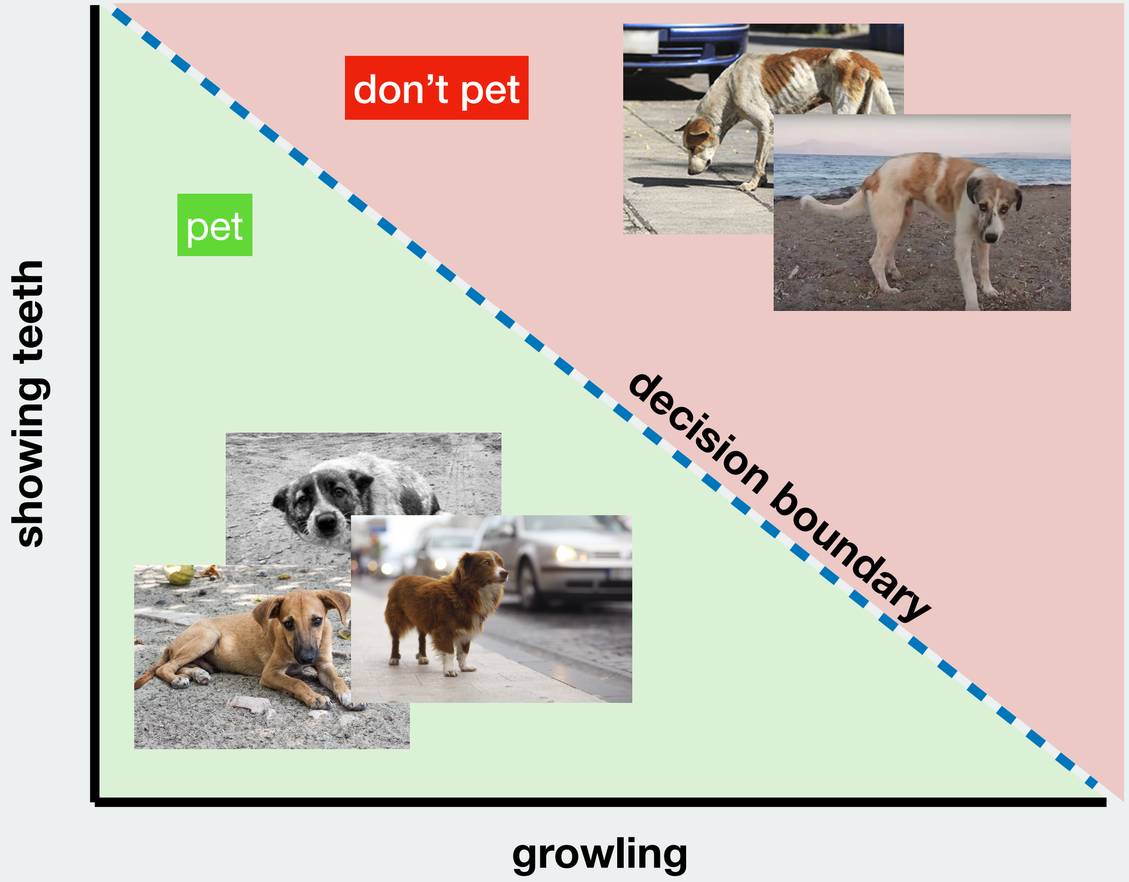
Canonical example



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

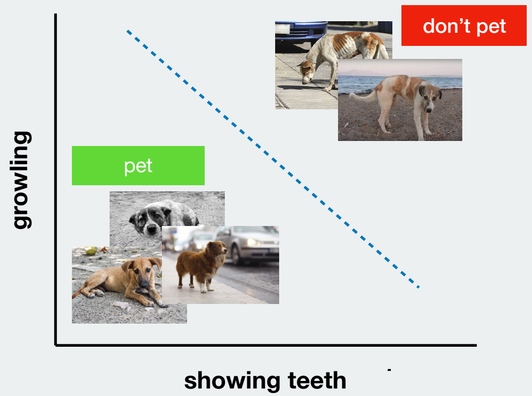
When you don't have outcomes 👏

Types of machine learning

Supervised

When you have outcomes 👏

- Classification



Unsupervised

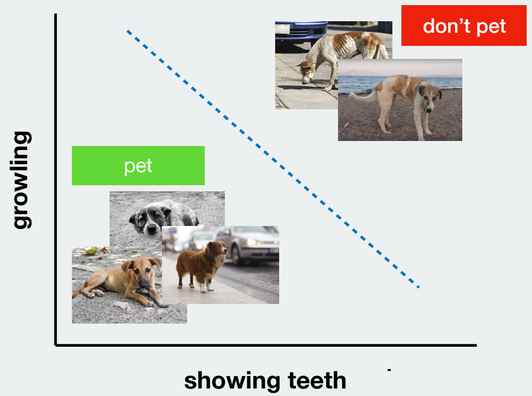
When you don't have outcomes 👏

Types of machine learning

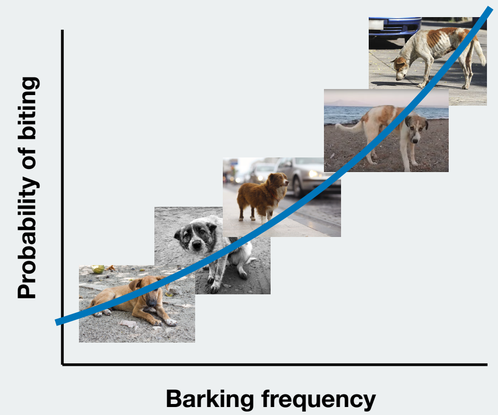
Supervised

When you have outcomes 👏

- Classification



- Regression



Unsupervised

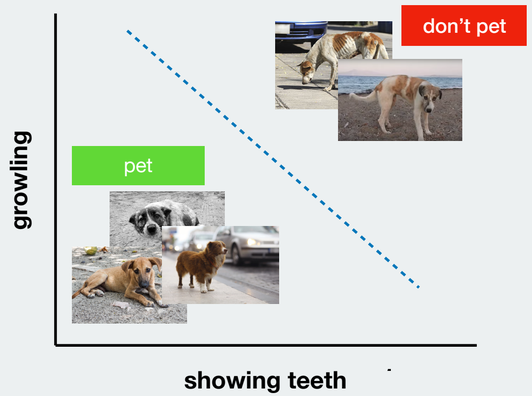
When you don't have outcomes 👏

Types of machine learning

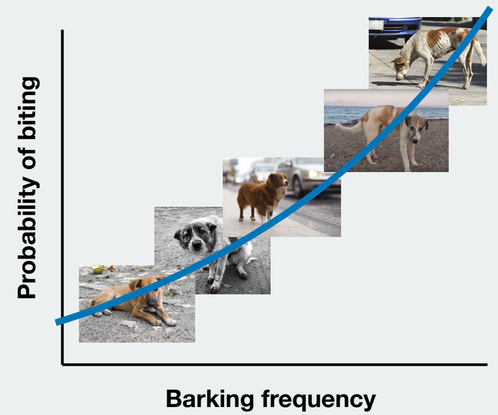
Supervised

When you have outcomes 👏

- Classification



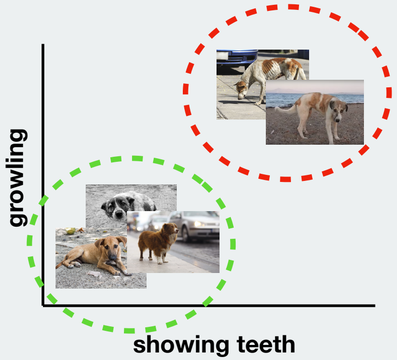
- Regression



Unsupervised

When you don't have outcomes 👏 ❌

- Clustering

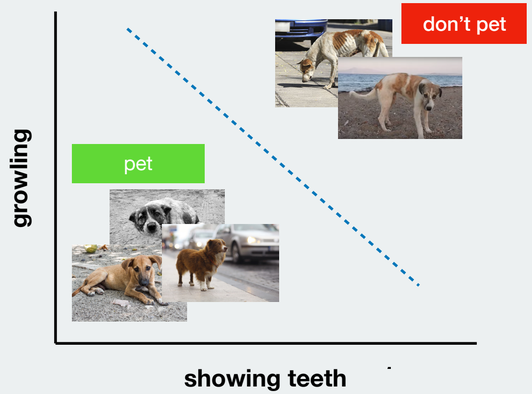


Types of machine learning

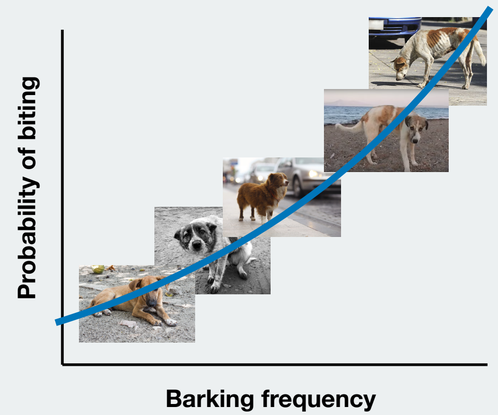
Supervised

When you have outcomes 👏

- Classification



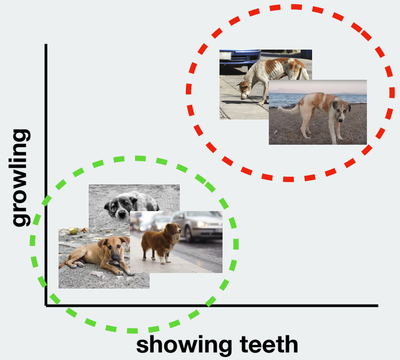
- Regression



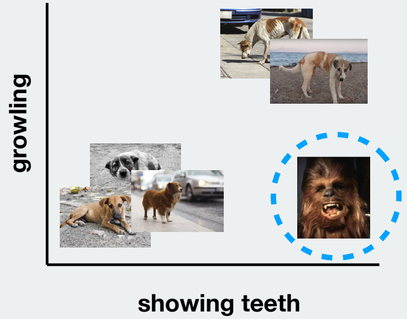
Unsupervised

When you don't have outcomes 👏 (crossed out)

- Clustering



- Outlier detection

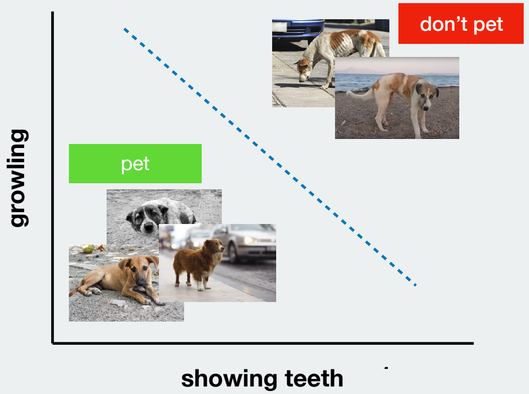


Types of machine learning

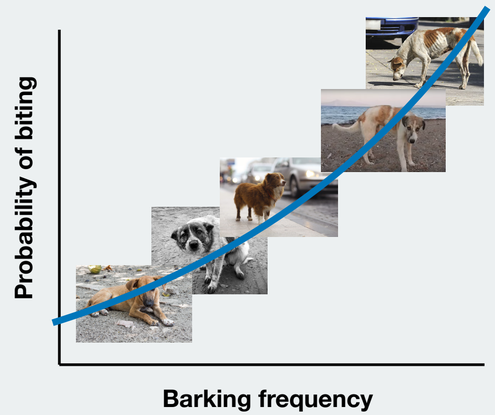
Supervised

When you have outcomes 👏

- Classification



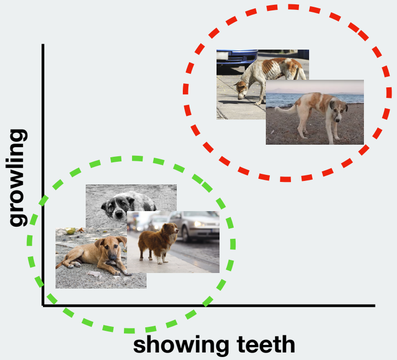
- Regression



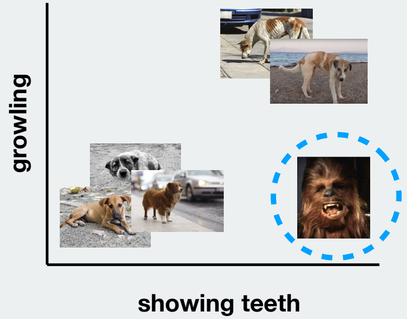
Unsupervised

When you don't have outcomes 👏

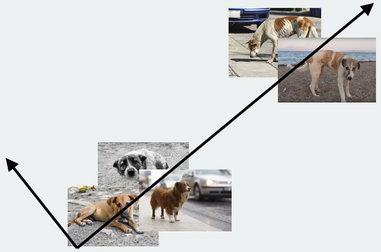
- Clustering



- Outlier detection



- Latent variable analysis

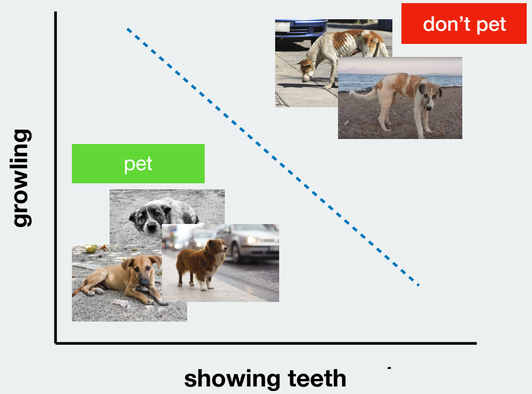


Types of machine learning

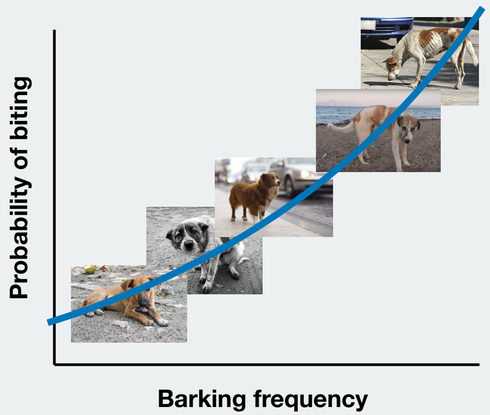
Supervised

When you have outcomes 👏

- Classification



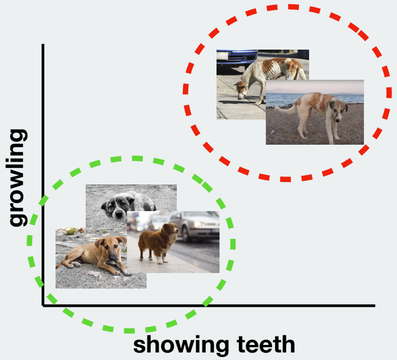
- Regression



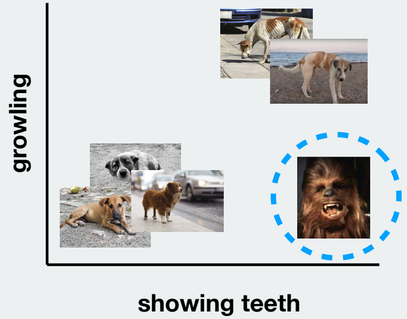
Unsupervised

When you don't have outcomes 👏

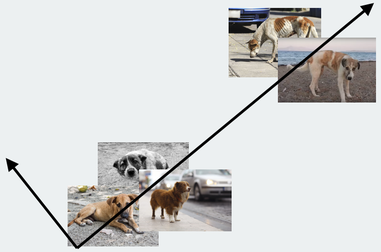
- Clustering



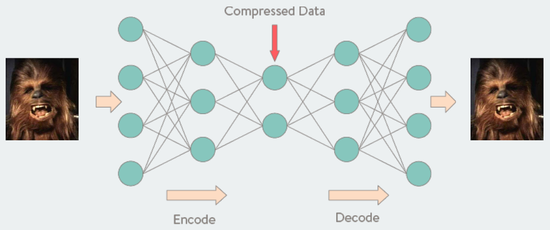
- Outlier detection



- Latent variable analysis

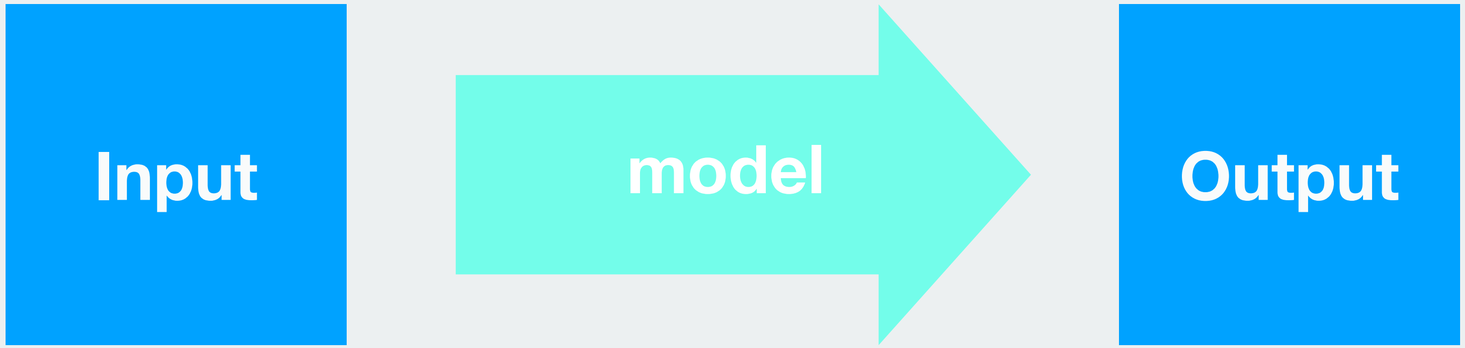


- Auto encoding



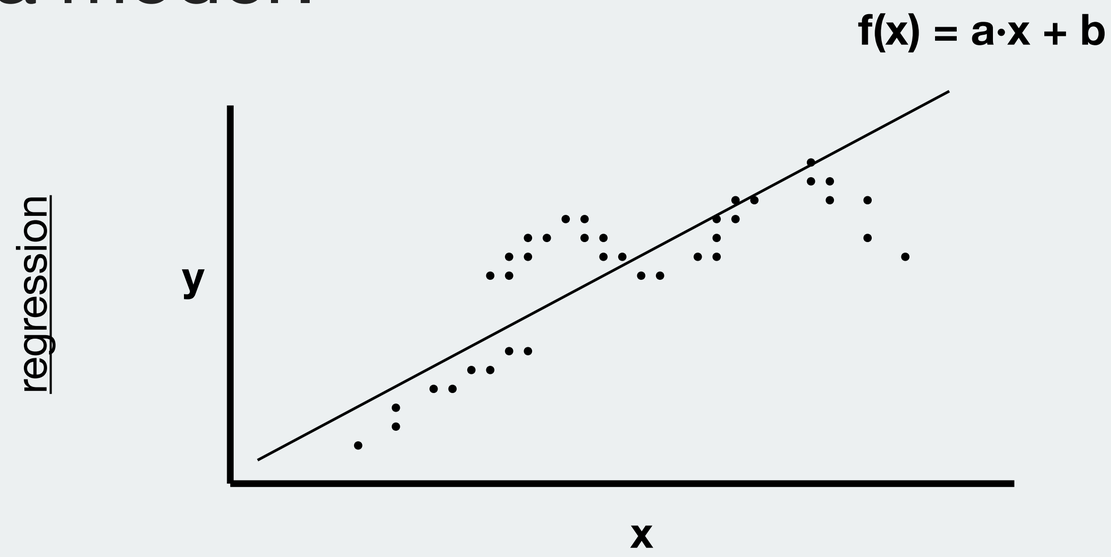
In a nutshell

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

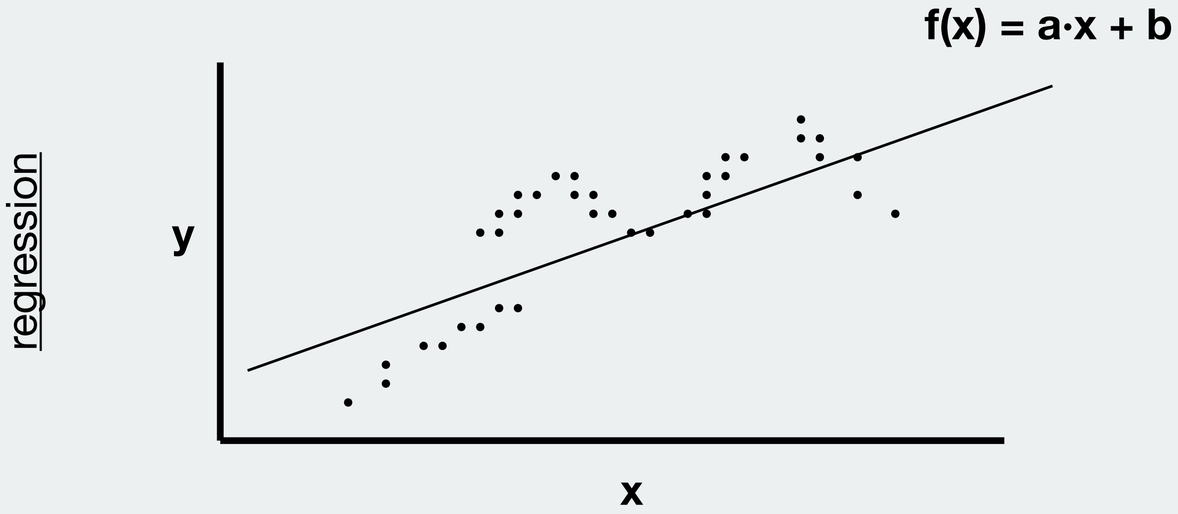


What is a model?

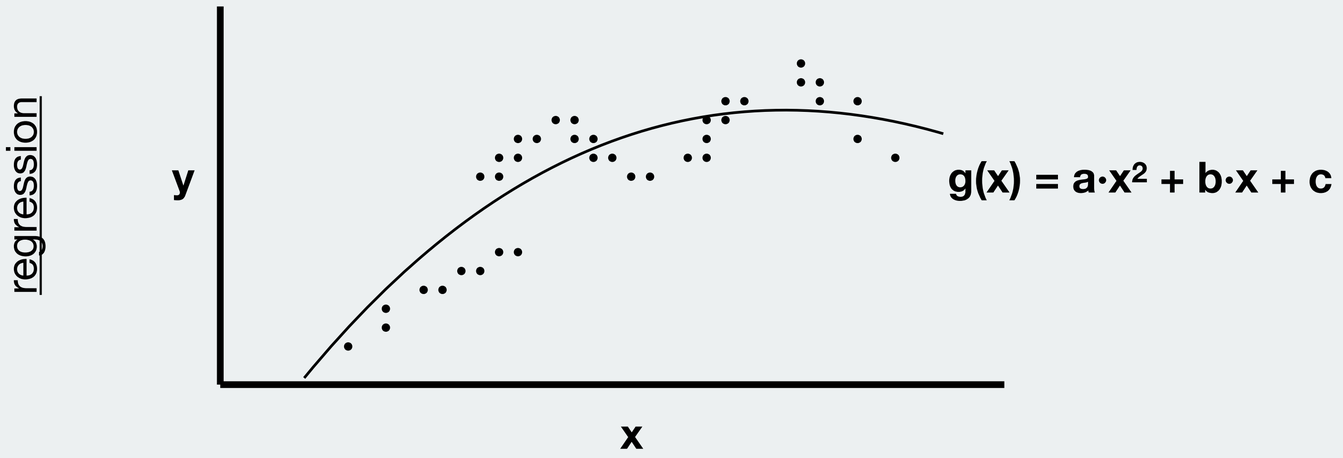
What is a model?



What is a model?

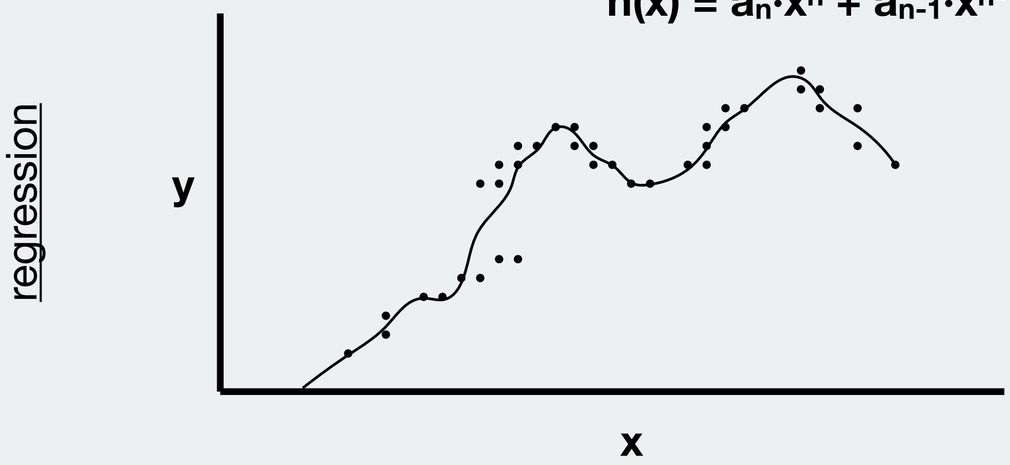


What is a model?



What is a model?

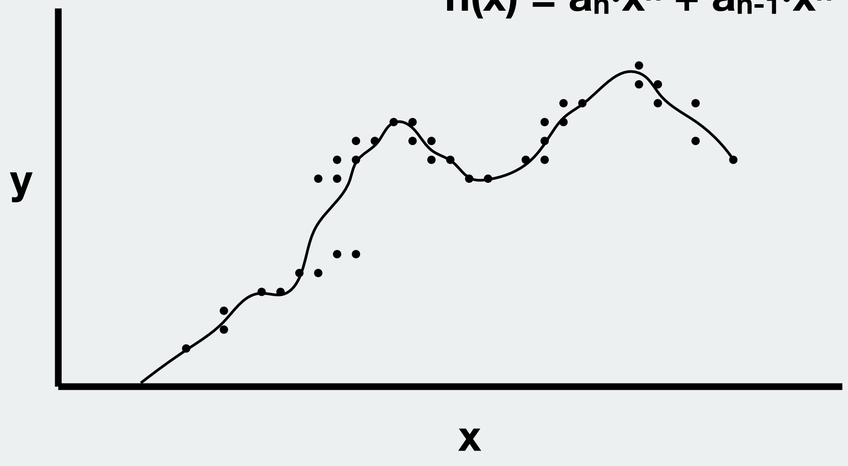
$$h(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_0 \cdot x^0$$



What is a model?

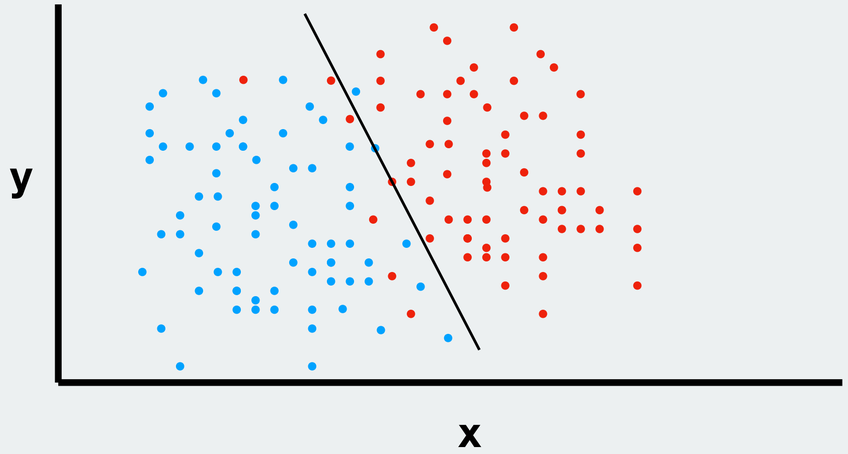
regression

$$h(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_0 \cdot x^0$$



classification

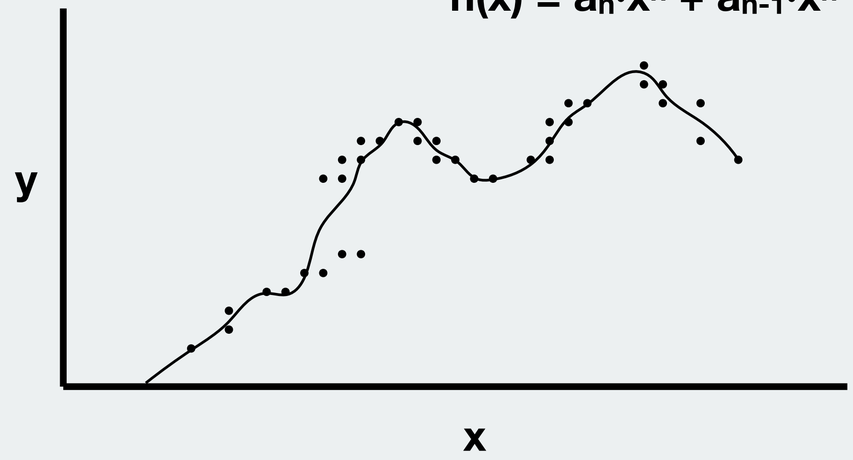
$$p_{red}(x, y) = \textit{few parameters}$$



What is a model?

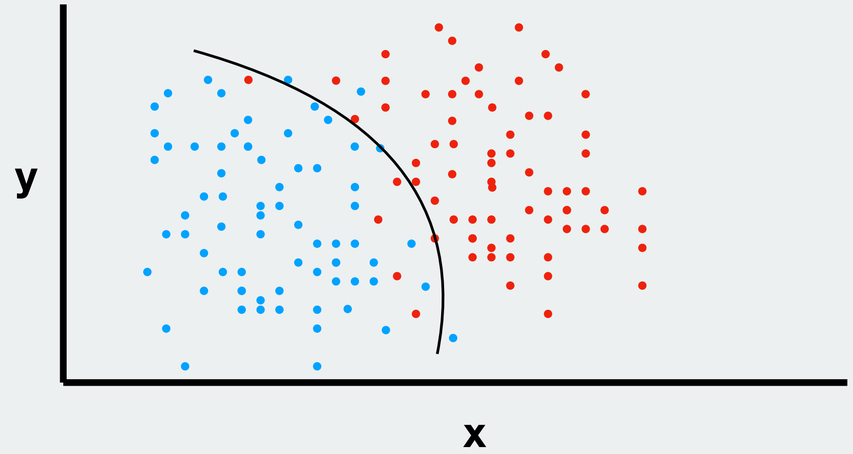
$$h(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_0 \cdot x^0$$

regression



$$p_{red}(x, y) = \textit{more parameters}$$

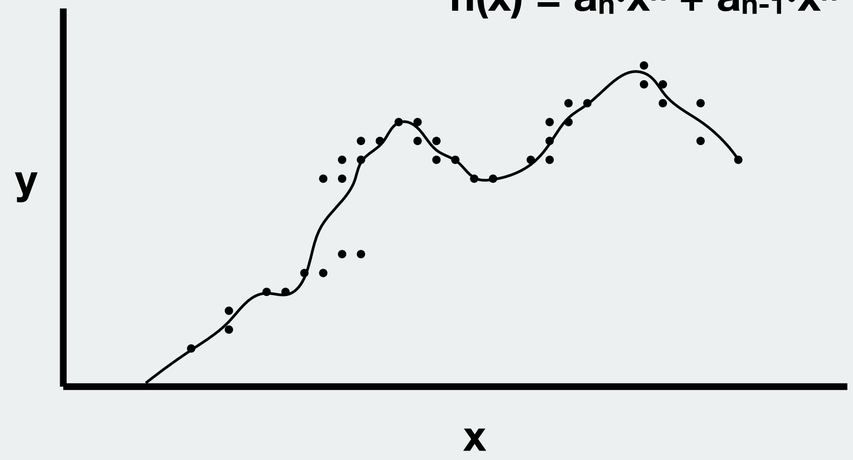
classification



What is a model?

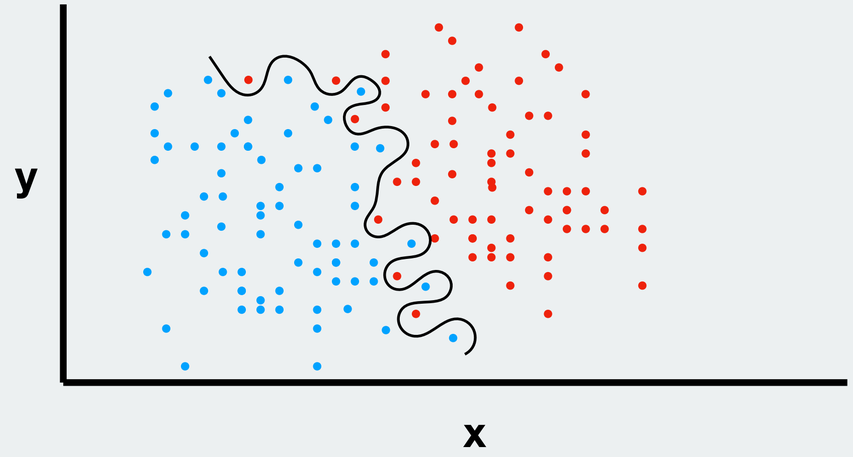
$$h(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_0 \cdot x^0$$

regression



$$p_{red}(x, y) = \textit{many parameters}$$

classification

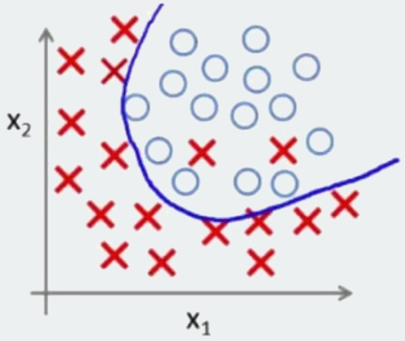


What is a model?

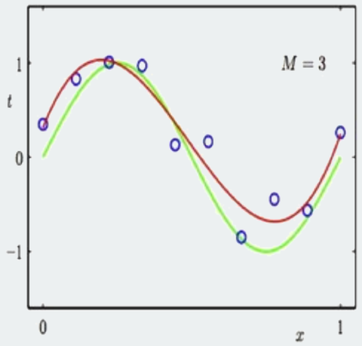
Underfitting and overfitting

Underfitting and overfitting

Classification



Regression



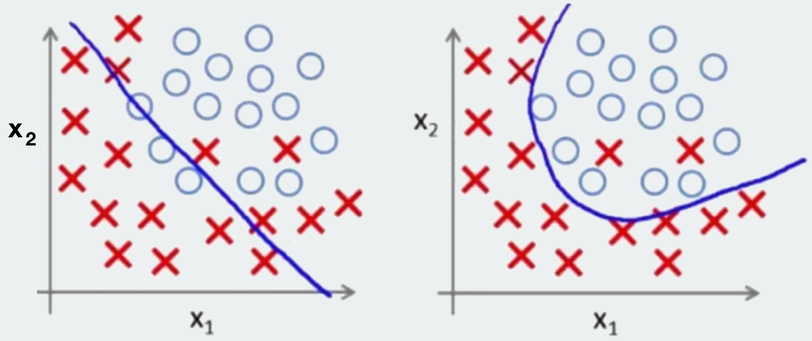
Small train error

Small test error

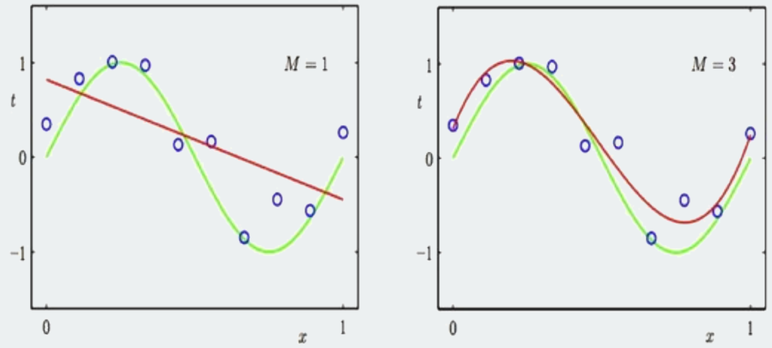
Underfitting and overfitting

Underfitting

Classification



Regression



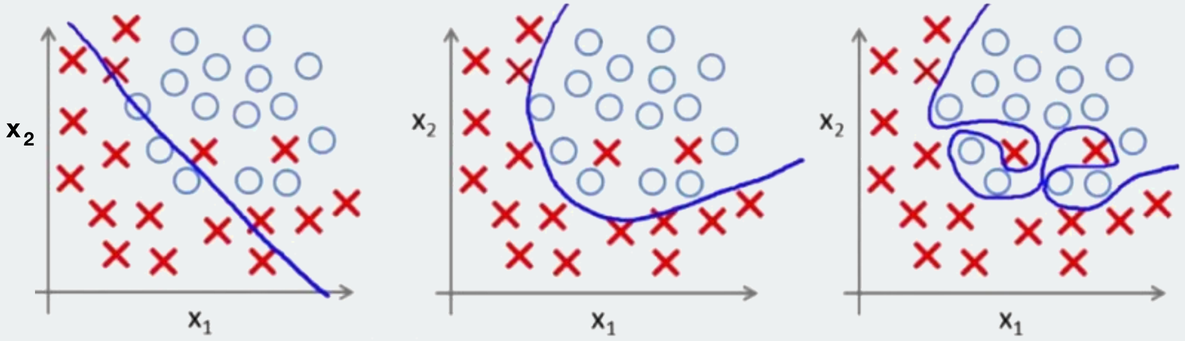
High train error **Small train error**

High test error **Small test error**

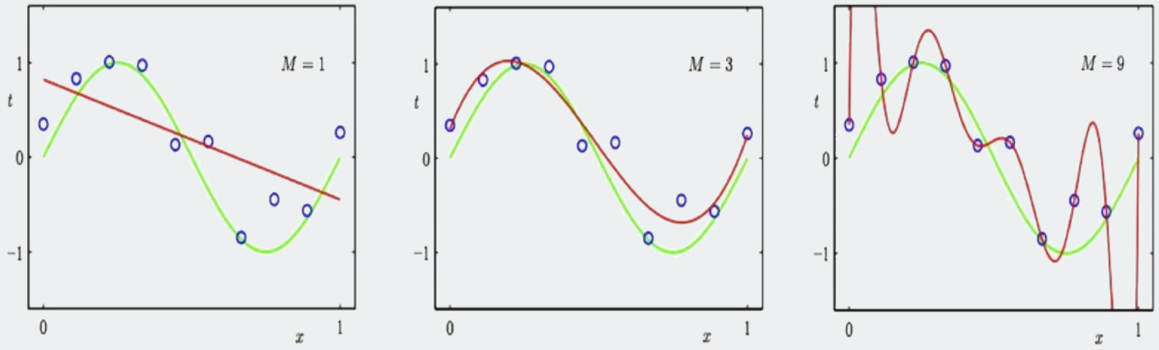
Underfitting and overfitting

Underfitting **Overfitting**

Classification



Regression



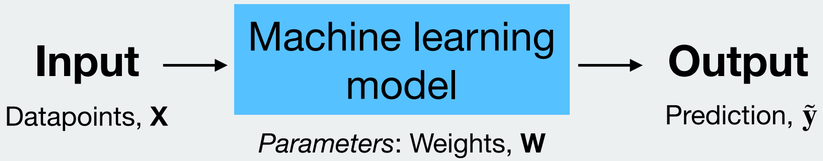
High train error **Small train error** **Zero train error**

High test error **Small test error** **High test error**

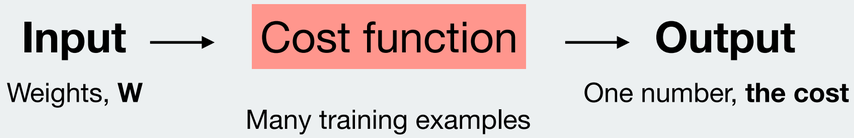
Finding model parameters

Finding model parameters

(1) The model

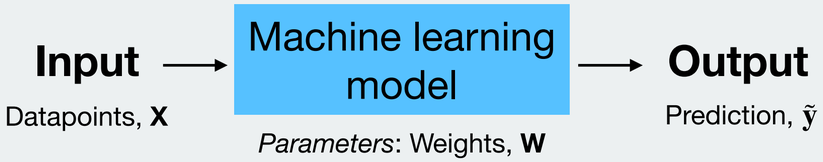


(2) Its performance

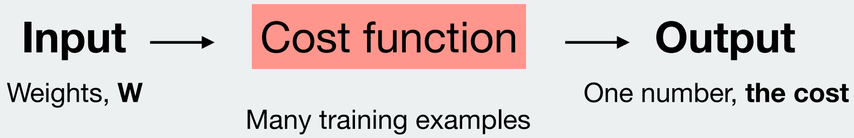


Finding model parameters

(1) The model



(2) Its performance



predicted from **X** given **W**

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

true

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

Finding model parameters

predicted from **X** given **W**

true

$$\tilde{\mathbf{y}} = \begin{bmatrix} 0.96 \\ 0.10 \\ 0.04 \\ \cdots \\ 0.70 \\ 0.02 \\ 0.99 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \cdots \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

(1) The model

Input → Machine learning model → **Output**

Datapoints, **X**

Parameters: Weights, **W**

Prediction, $\tilde{\mathbf{y}}$

(2) Its performance

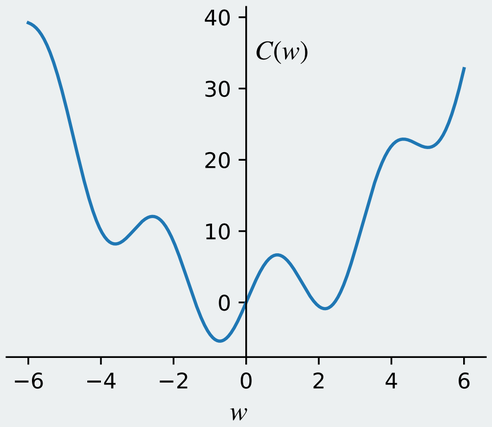
Input → Cost function → **Output**

Weights, **W**

Many training examples

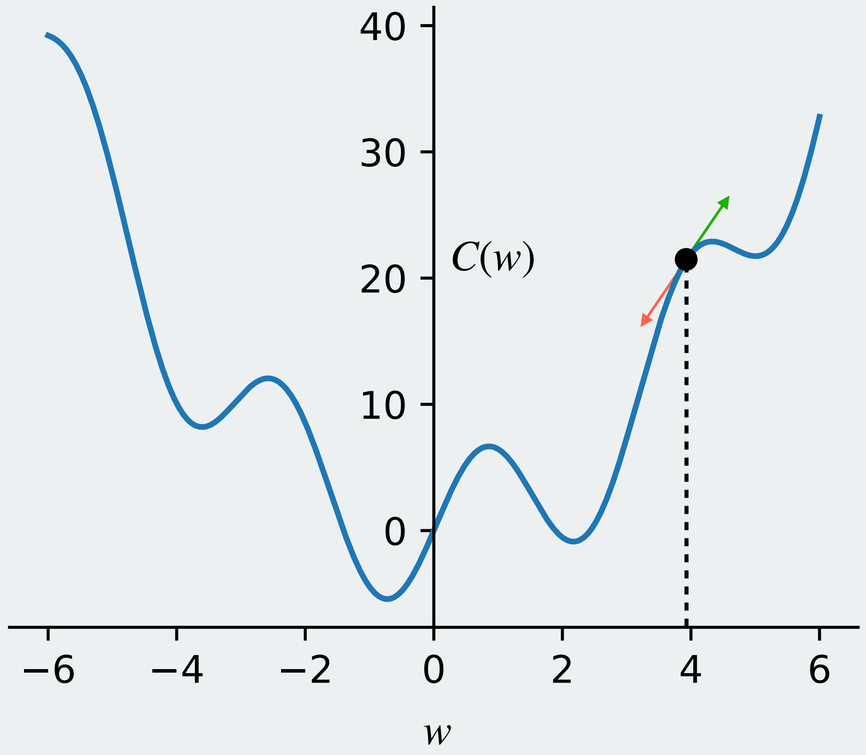
One number, **the cost**

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



Finding model parameters

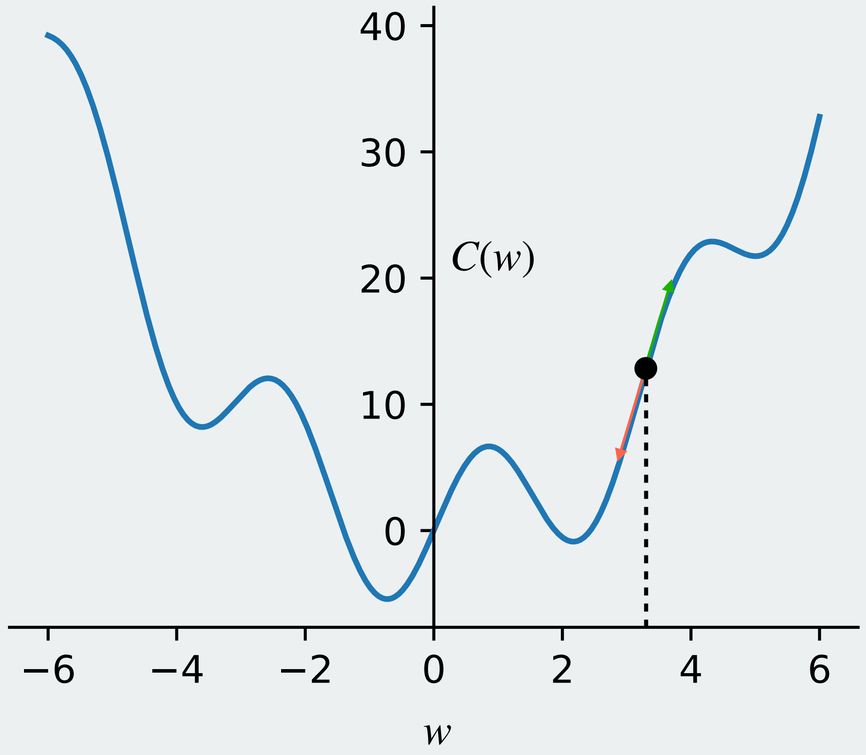
> Gradient Descent



$$\tilde{\mathbf{y}} = \begin{bmatrix} 0.82 \\ 0.47 \\ 0.21 \\ \cdots \\ 0.41 \\ 0.74 \\ 0.23 \end{bmatrix}$$

Finding model parameters

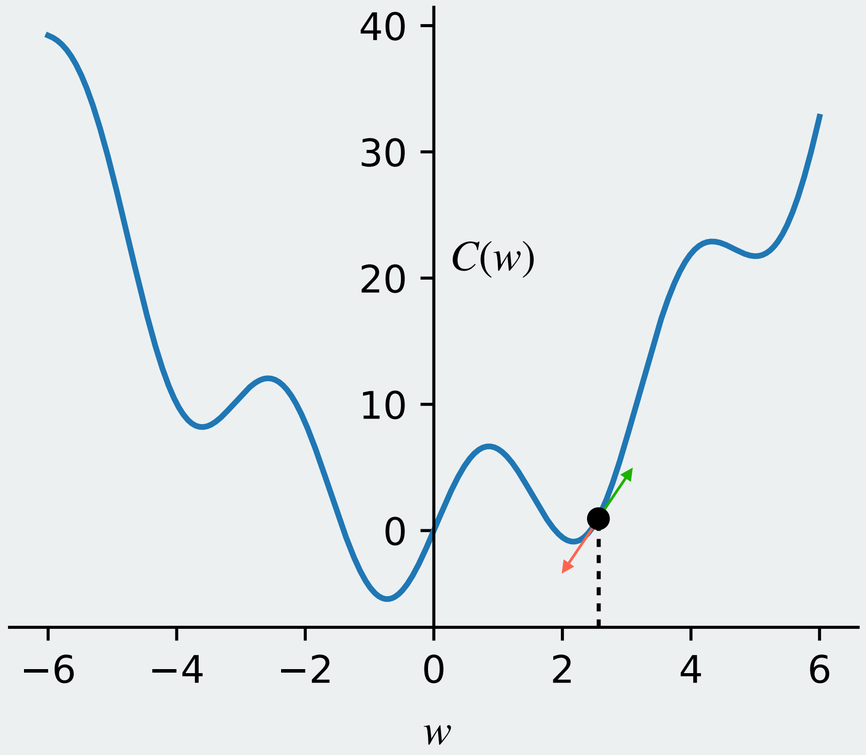
> Gradient Descent



$$\tilde{\mathbf{y}} = \begin{bmatrix} 0.82 \\ 0.32 \\ 0.12 \\ \cdots \\ 0.25 \\ 0.85 \\ 0.11 \end{bmatrix}$$

Finding model parameters

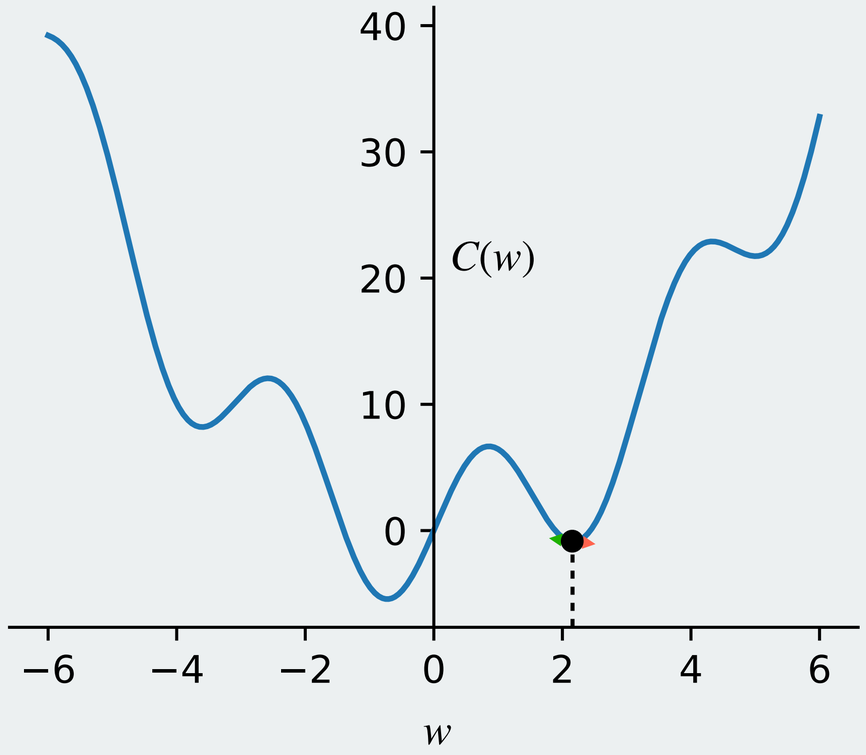
> Gradient Descent



$$\tilde{\mathbf{y}} = \begin{bmatrix} 0.96 \\ 0.08 \\ 0.02 \\ \cdots \\ 0.12 \\ 0.96 \\ 0.01 \end{bmatrix}$$

Finding model parameters

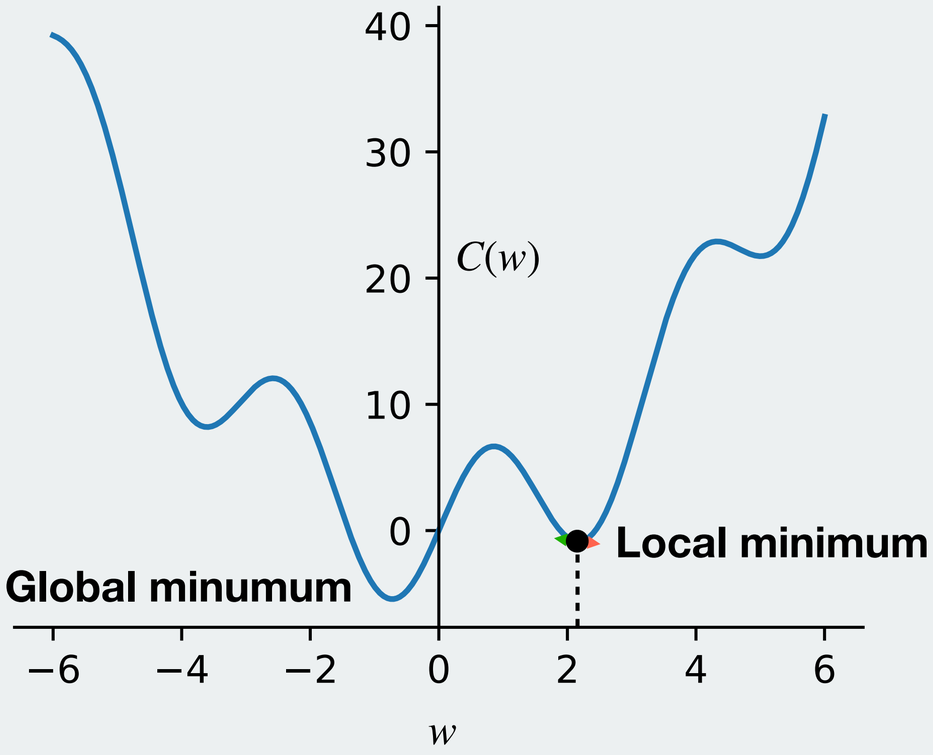
> Gradient Descent



$$\tilde{\mathbf{y}} = \begin{bmatrix} 0.99 \\ 0.02 \\ 0.00 \\ \cdots \\ 0.01 \\ 1.00 \\ 0.00 \end{bmatrix}$$

Finding model parameters

> Gradient Descent

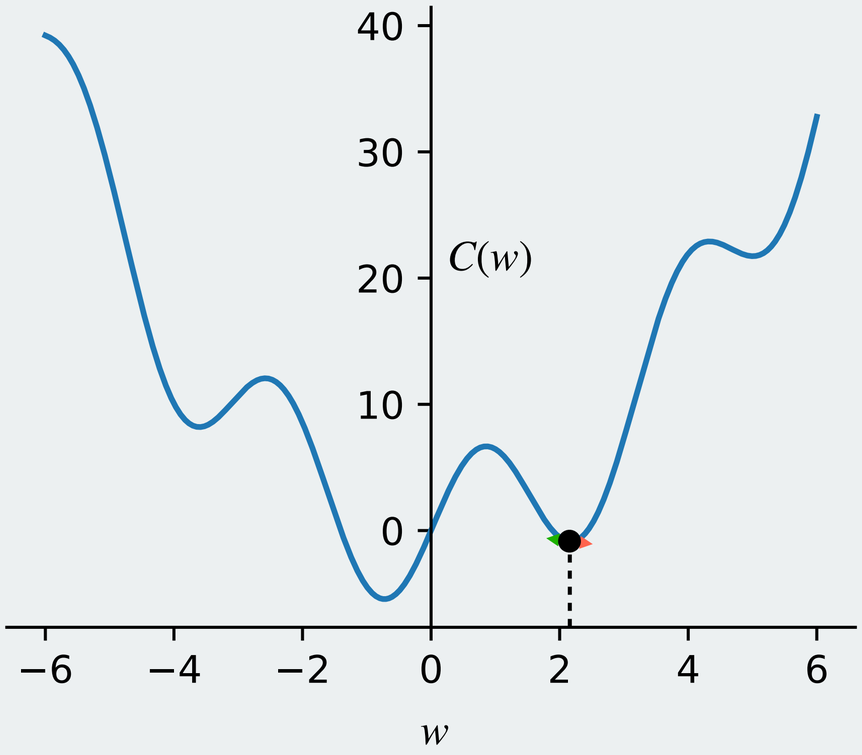


$$\tilde{\mathbf{y}} = \begin{bmatrix} 0.99 \\ 0.02 \\ 0.00 \\ \cdots \\ 0.01 \\ 1.00 \\ 0.00 \end{bmatrix}$$

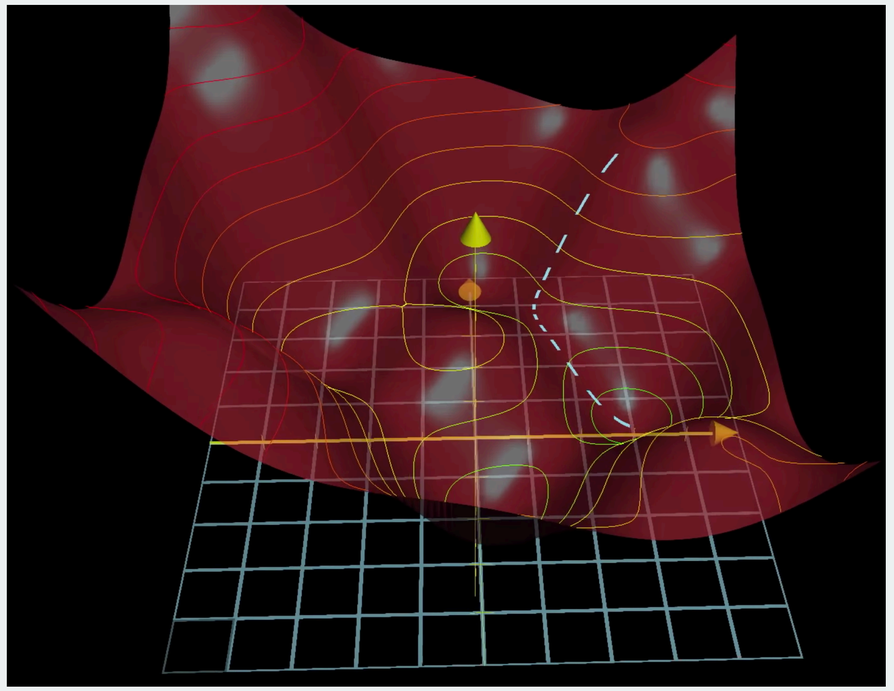
Finding model parameters

> Gradient Descent

1D



2D

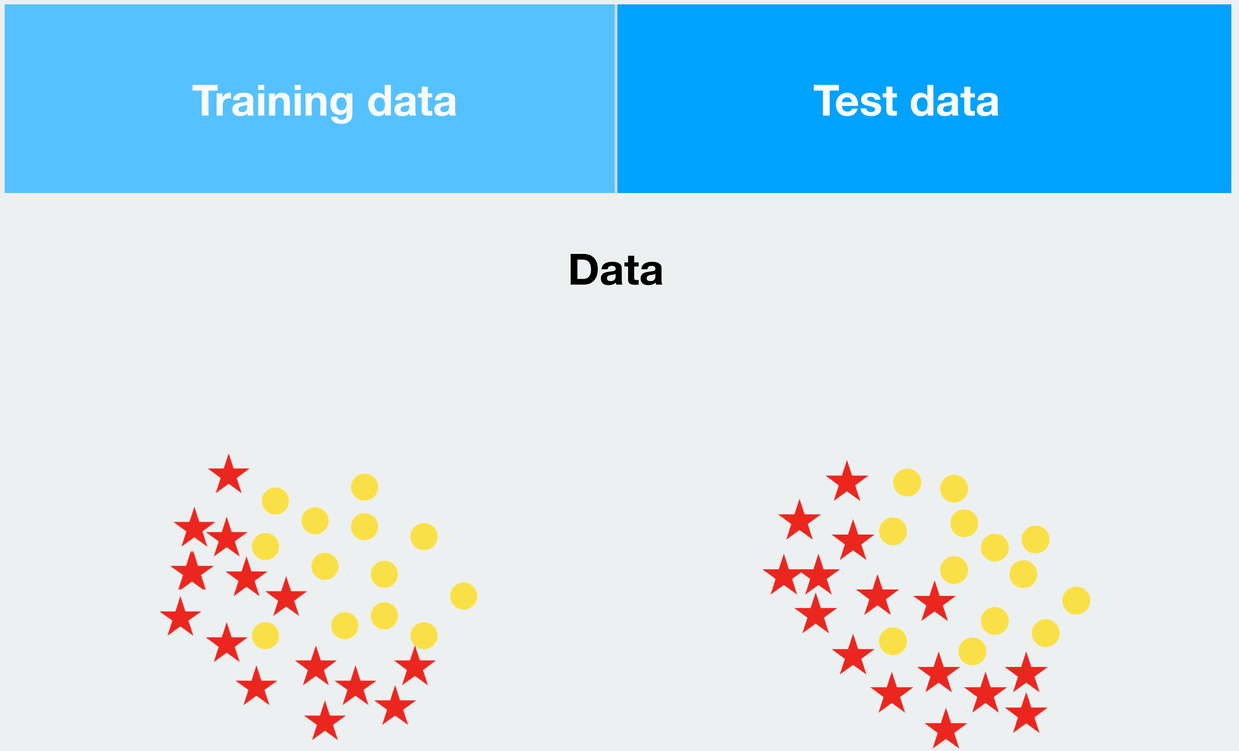


Model evaluation

Model evaluation

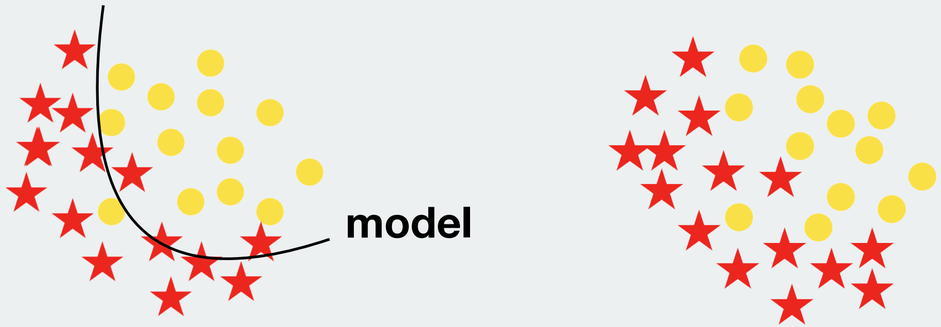
Data

Model evaluation



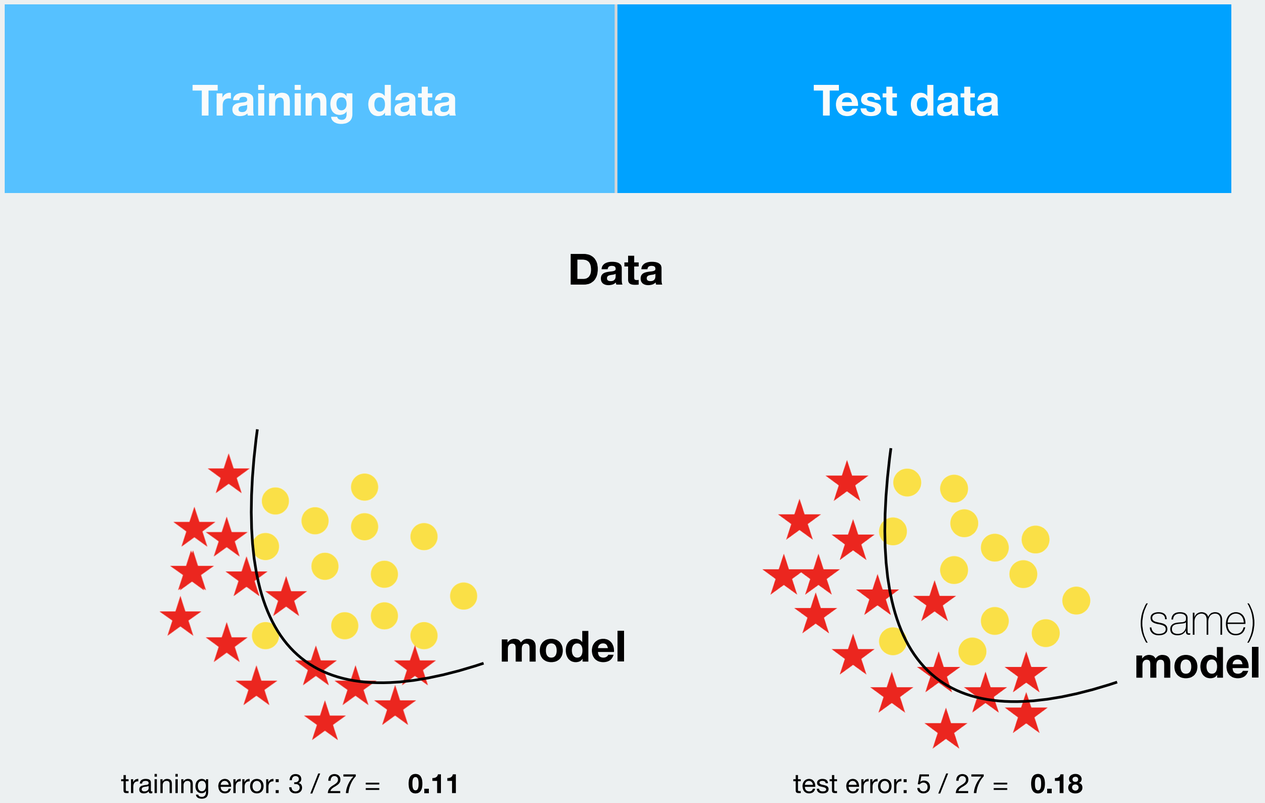
Model evaluation

Data



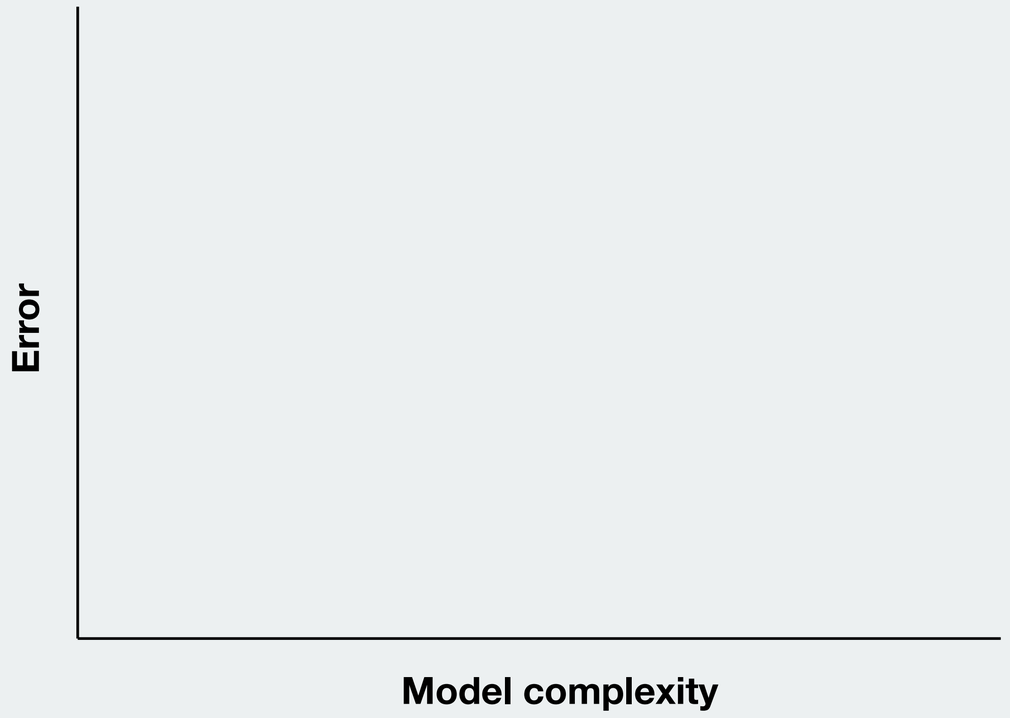
training error: 3 / 27 = **0.11**

Model evaluation

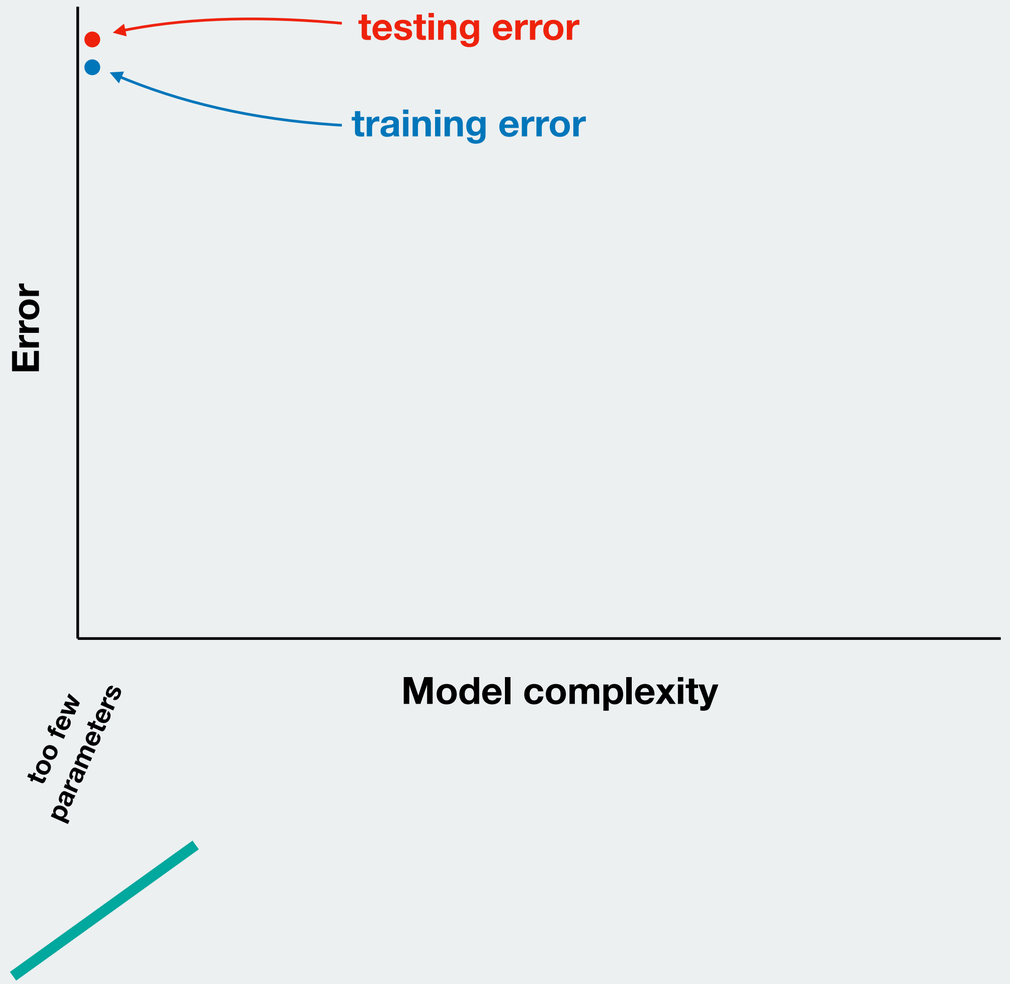


Avoiding underfitting and overfitting

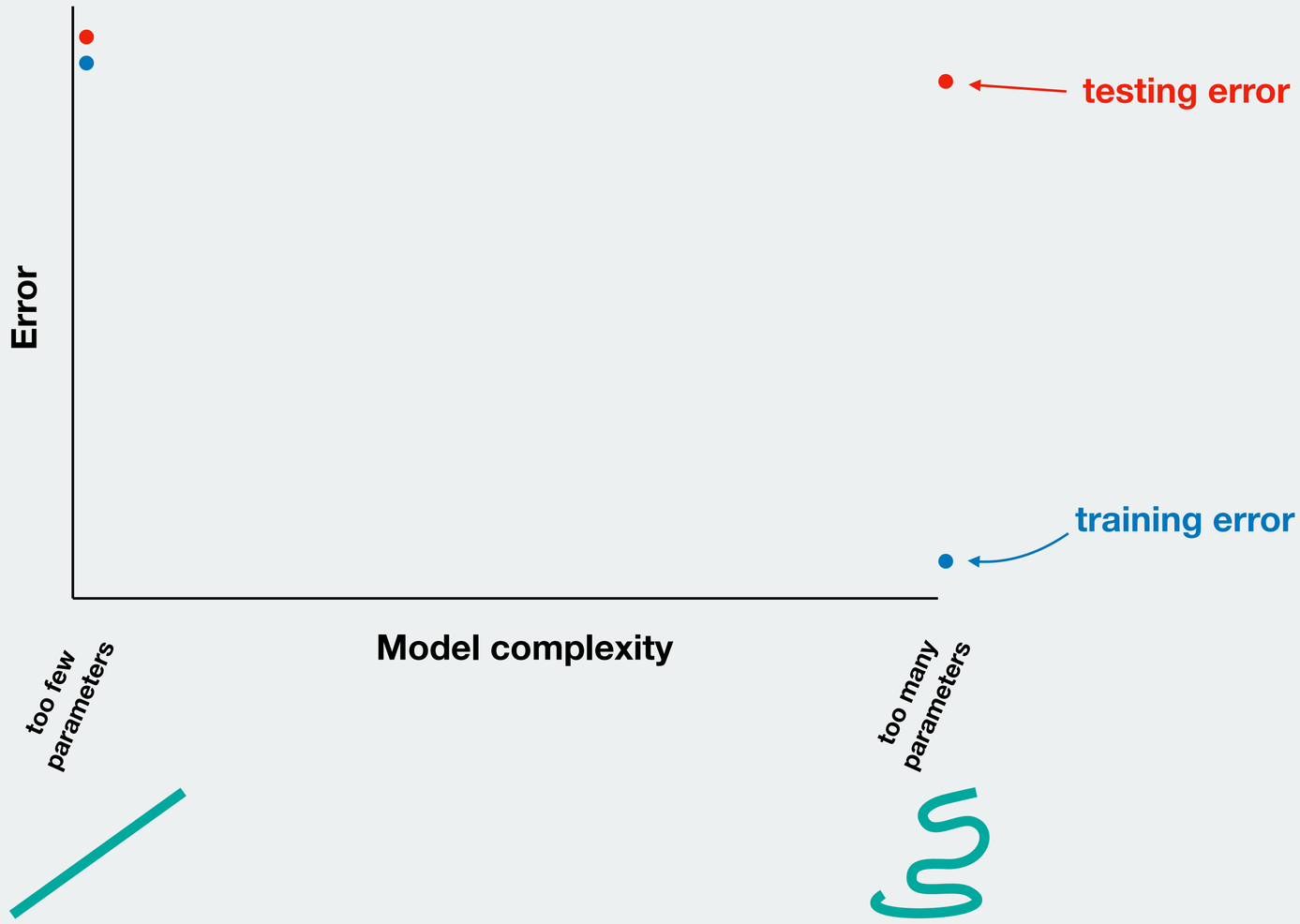
Avoiding underfitting and overfitting



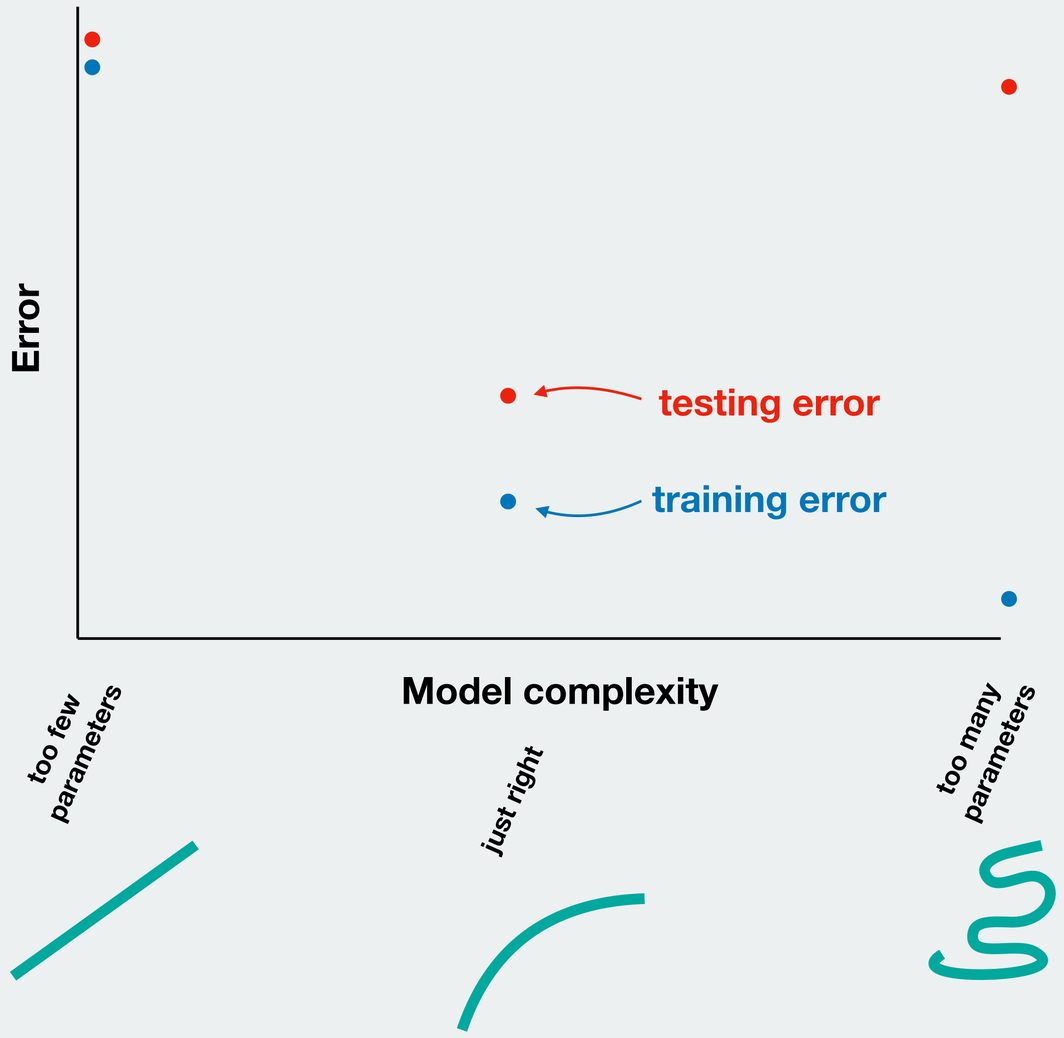
Avoiding underfitting and overfitting



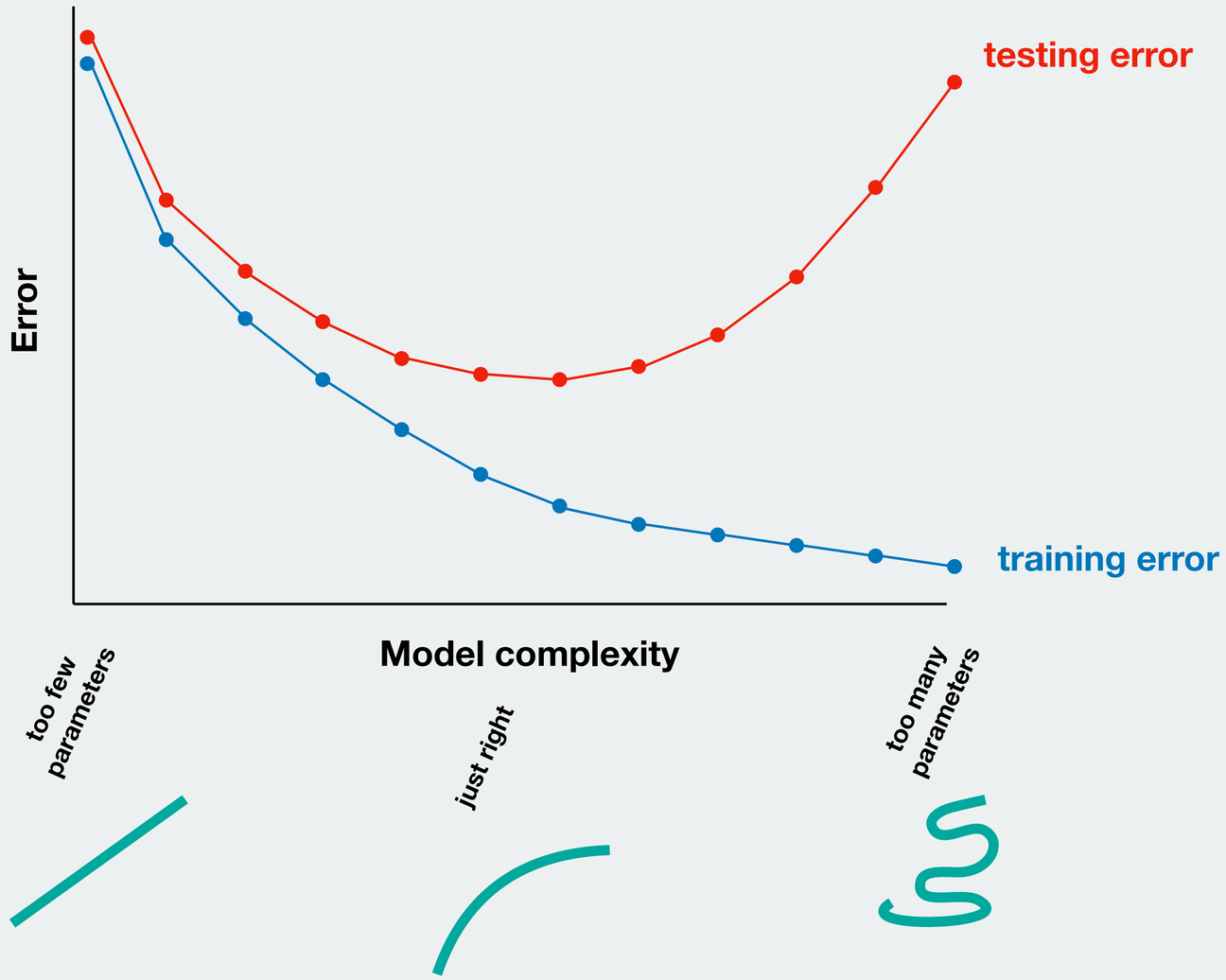
Avoiding underfitting and overfitting



Avoiding underfitting and overfitting

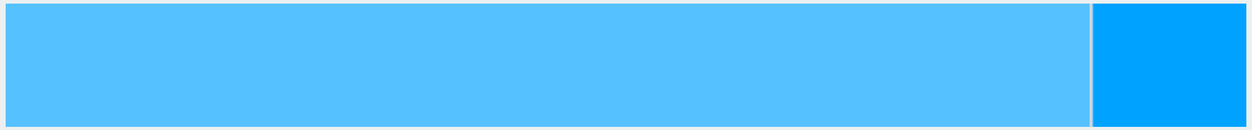


Avoiding underfitting and overfitting

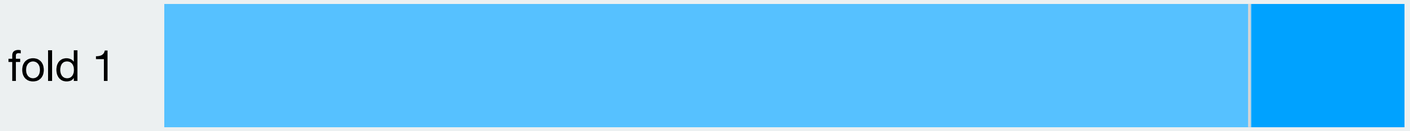


Cross validation

Cross validation

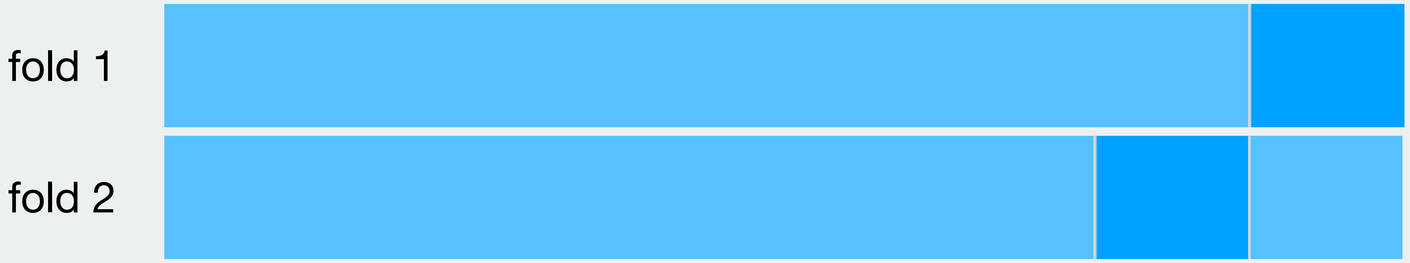


Cross validation



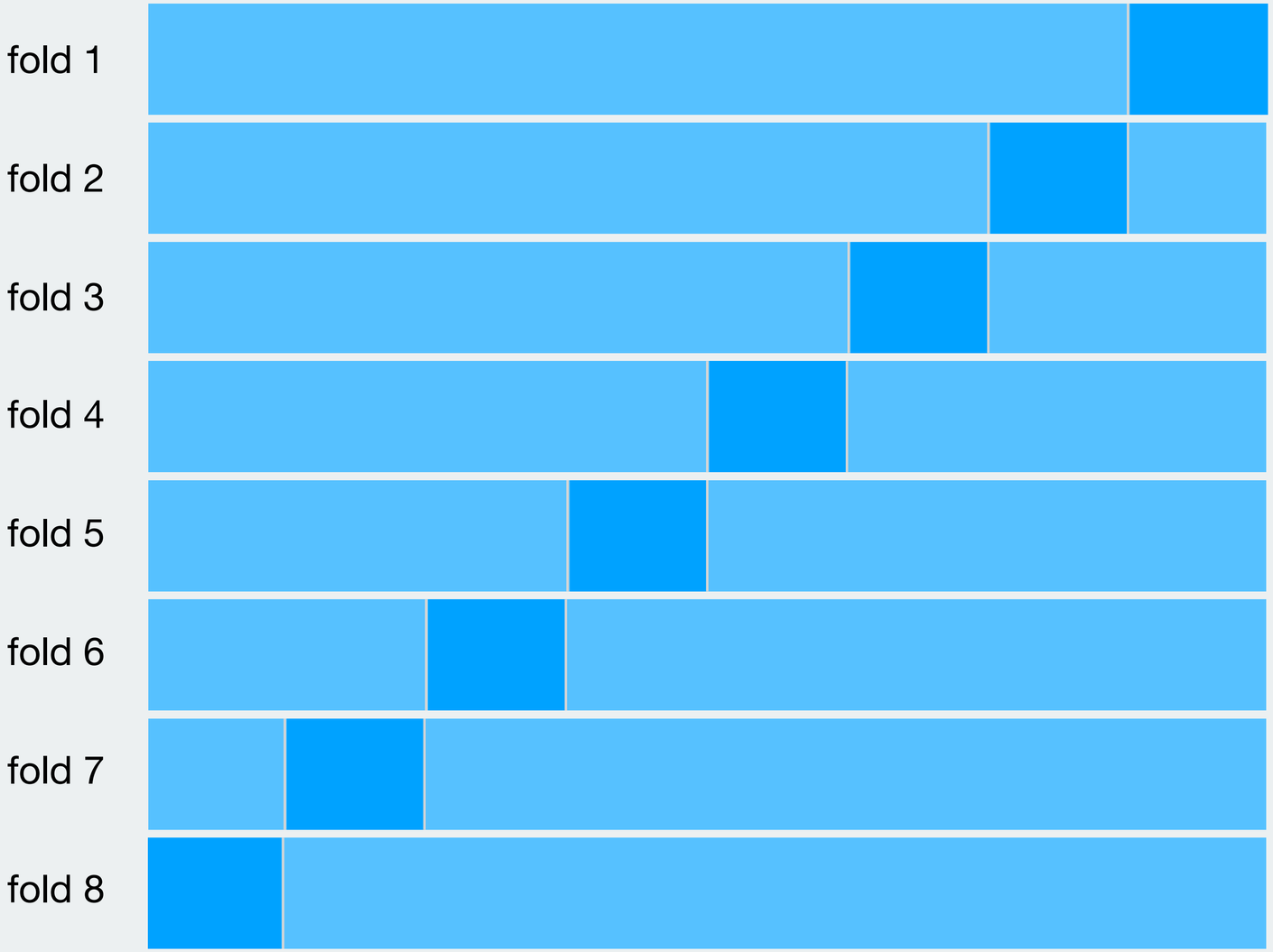
Train err.	Test err.
0.07	0.11

Cross validation



	Train err.	Test err.
fold 1	0.07	0.11
fold 2	0.04	0.18

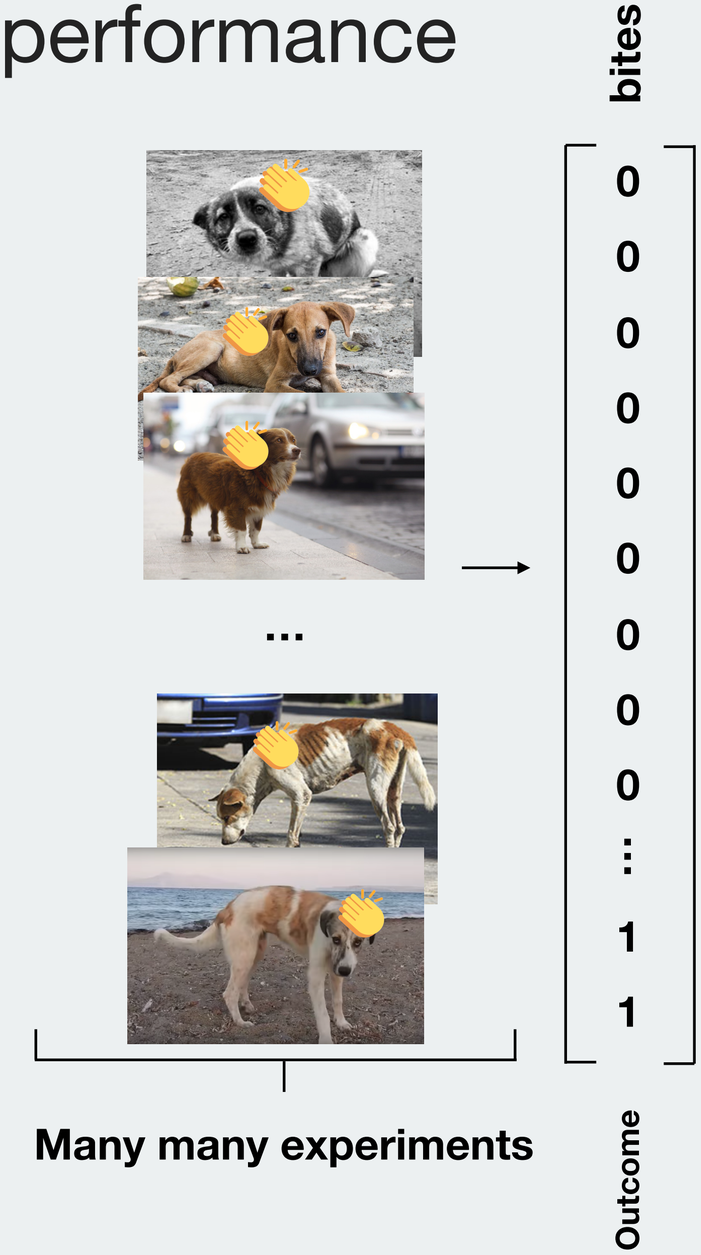
Cross validation



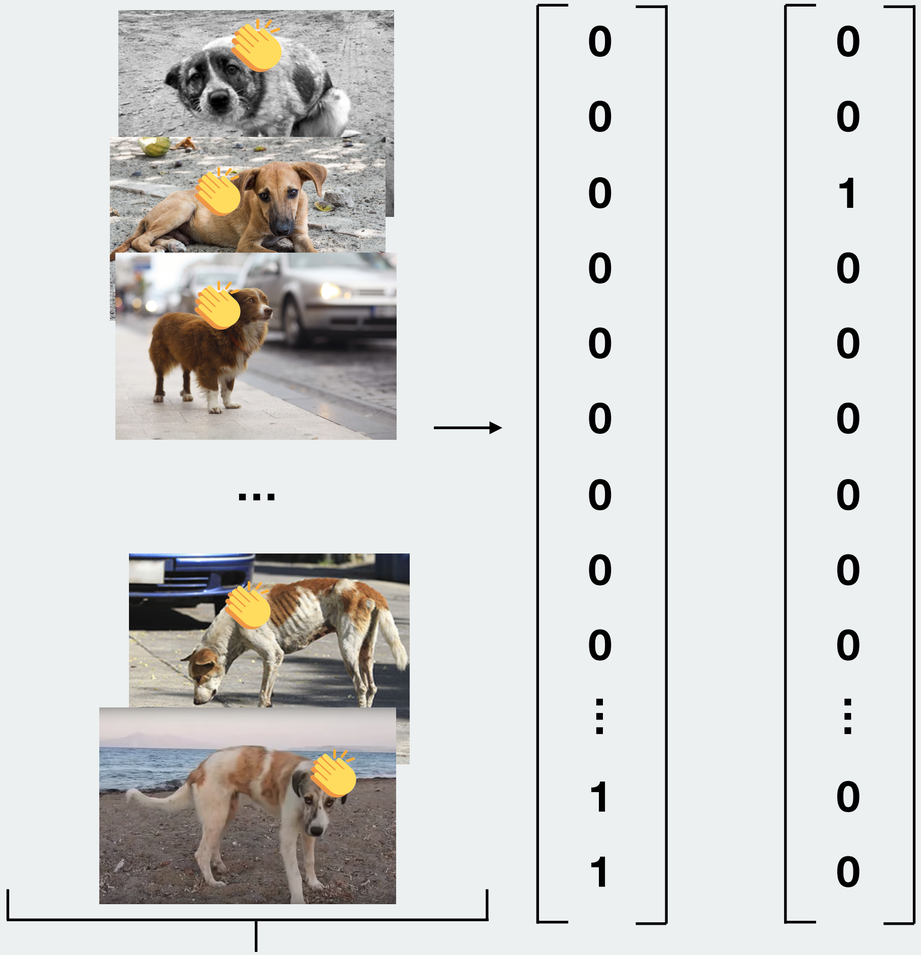
	Train err.	Test err.
fold 1	0.07	0.11
fold 2	0.04	0.18
fold 3	0.00	0.32
fold 4	0.11	0.11
fold 5	0.07	0.11
fold 6	0.02	0.14
fold 7	0.16	0.1
fold 8	0.07	0.11
	0.07	**0.15**

Model performance

Model performance

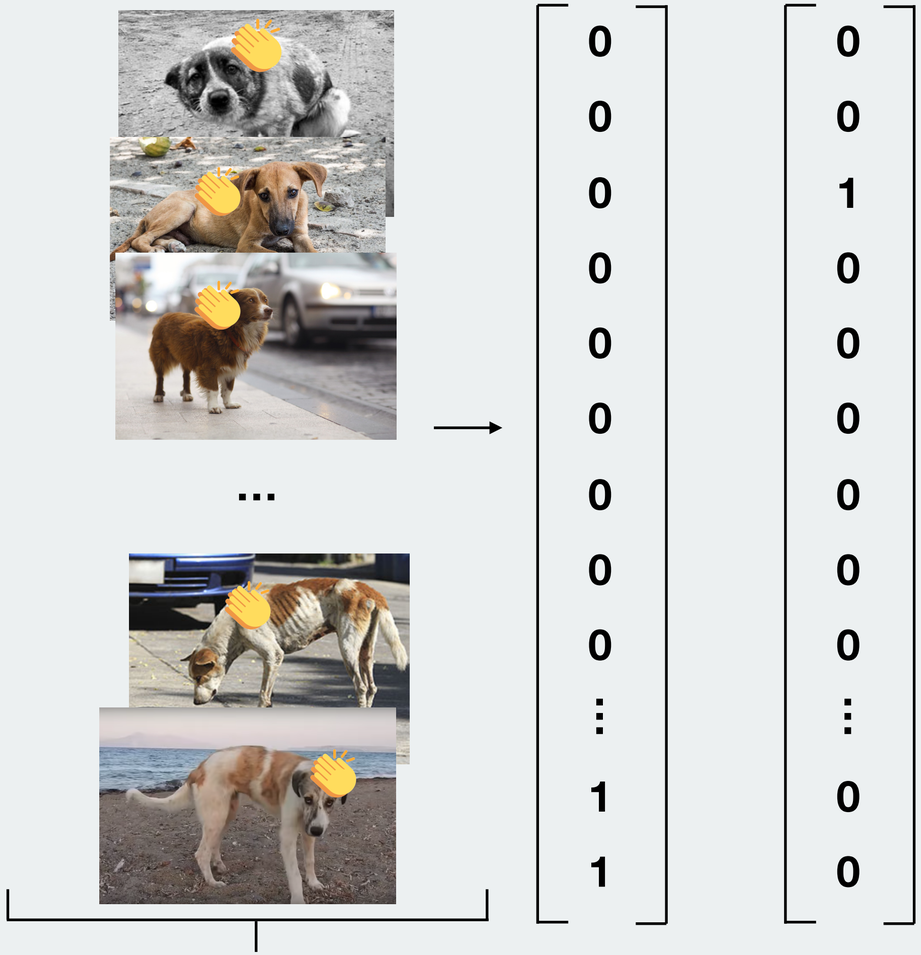


Model performance



Dumb model: *Only dogs named 'Frank' bite*

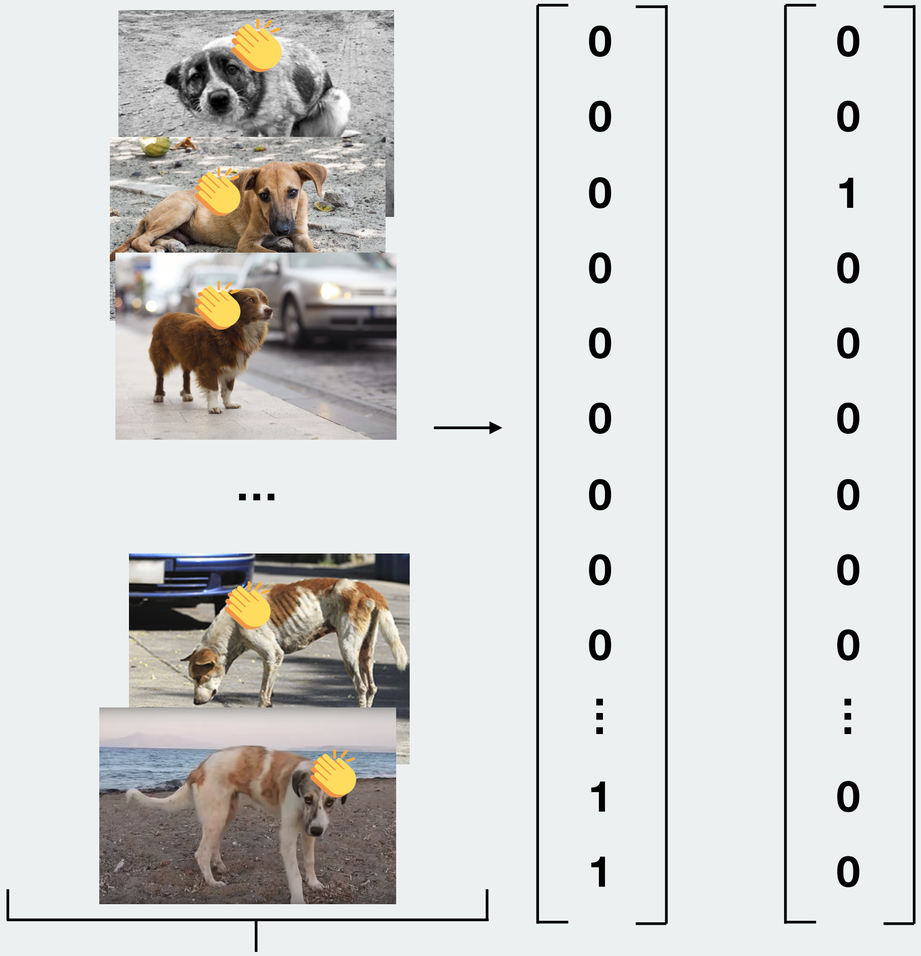
Model performance



Dumb model: *Only dogs named 'Frank' bite*

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

Model performance



Dumb model: *Only dogs named 'Frank' bite*

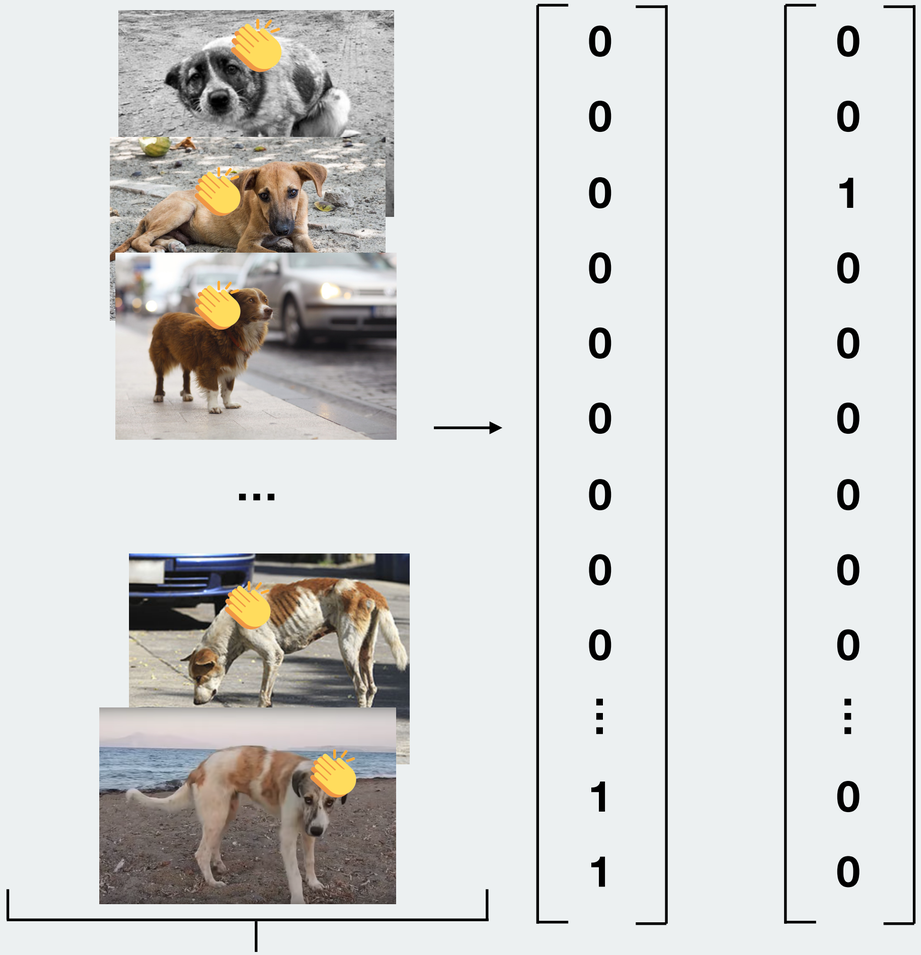
Dogs named Frank:	1%
Dogs that bite:	2%
Frank & bites:	0.02%

Accuracy:

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

high!

Model performance



Dumb model: *Only dogs named 'Frank' bite*

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

Accuracy:

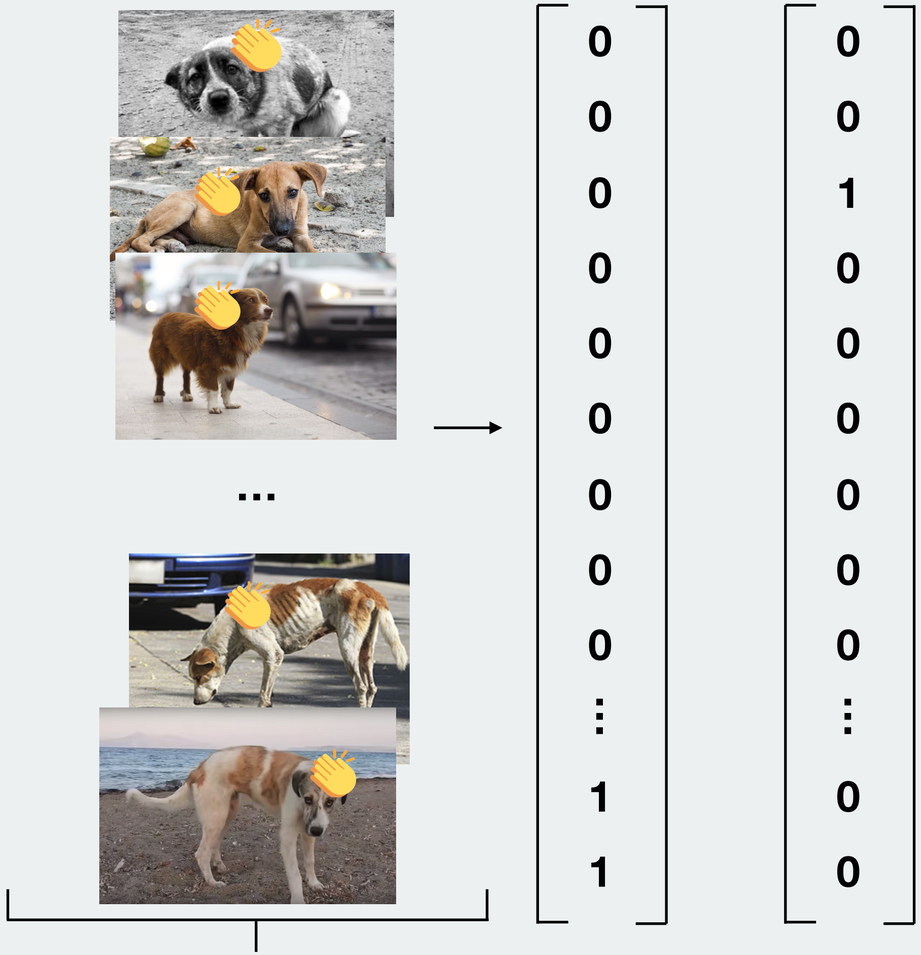
$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{200 + 970200}{1000000} = 97\%$$

high!

Confusion matrix:

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

Model performance



Dumb model: *Only dogs named 'Frank' bite*

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

Accuracy:

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

high!

Confusion matrix:

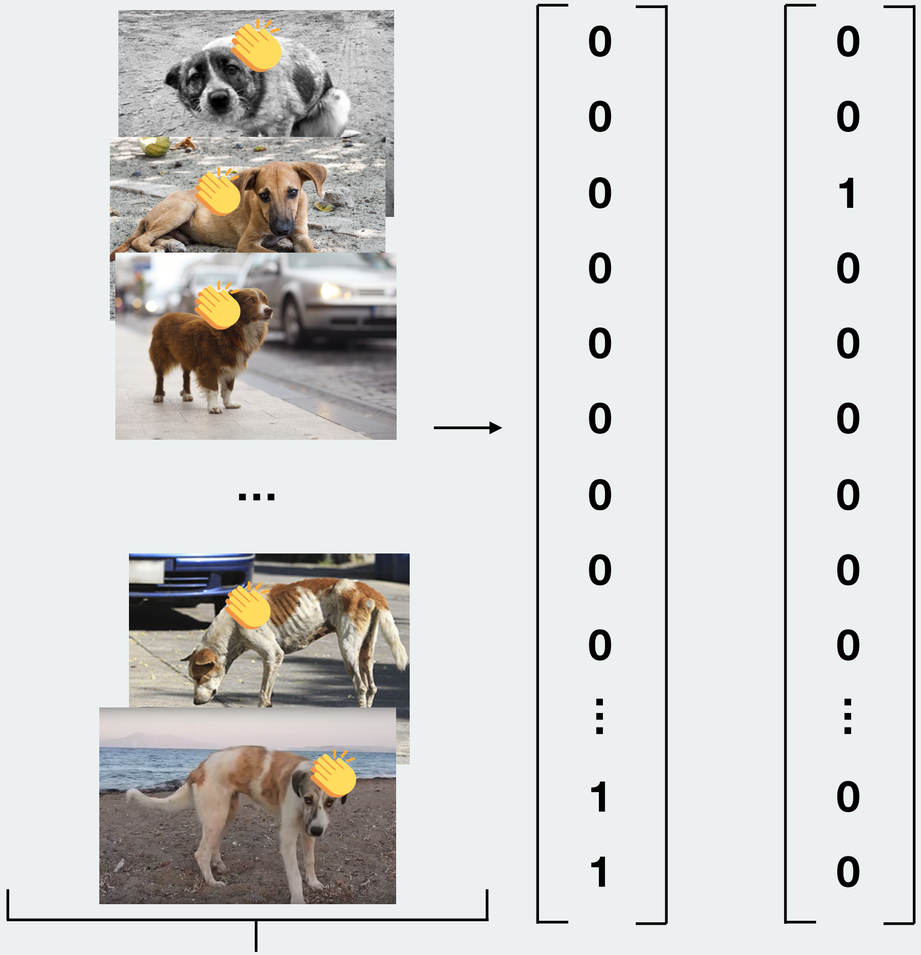
Prediction \ Outcome	Bite	No bite
Bite	TP: 200	FP: 9800
No bite	FN: 19800	TN: 970200

Precision & recall

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

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

Model performance



Dumb model: *Only dogs named 'Frank' bite*

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

Accuracy:

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

high!

Confusion matrix:

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

Precision & recall

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

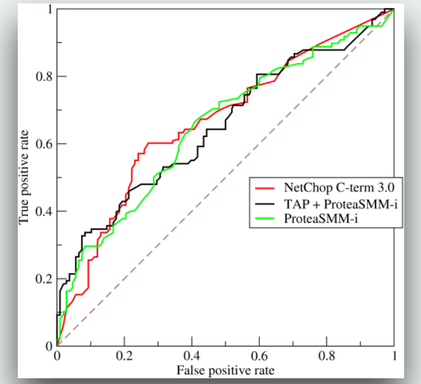
$$recall = \frac{TP}{TP + FN} = \frac{200}{200 + 19800} = 1\%$$

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

Pretty easy to understand

“Honest” measure of performance

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

```
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],
```

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],
```

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

```
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],
```

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

```
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],
```

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

```
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],
```

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

```
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],
```

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

```
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],
```

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

```
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],
```

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

```
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],
```

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

```
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],
```

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

```
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],
```

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 [162]: y_true
Last executed 2018-02-11 22:47:23 in 5ms
Out[162]: 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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
                 2, 2, 2, 2, 2, 2])
```

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],
```

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 [162]:   1 y_true
          Last executed 2018-02-11 22:47:23 in 5ms

Out[162]: 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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
                 2, 2, 2, 2, 2, 2])
```

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],
```

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 [162]: 1 y_true
          Last executed 2018-02-11 22:47:23 in 5ms

Out[162]: 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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
                 2, 2, 2, 2, 2, 2])
```

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],
```

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 [162]: y_true
          Last executed 2018-02-11 22:47:23 in 5ms

Out[162]: 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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
                 2, 2, 2, 2, 2, 2])
```

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],
```

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 [163]: 1 y_pred
          Last executed 2018-02-11 22:47:43 in 5ms

Out[163]: 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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1,
                 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2,
                 2, 2, 2, 2, 2, 2])
```

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],
```

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

```
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],
```

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],
```

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

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

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],
```

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]: sum(y_pred == y_true)
          Last executed 2018-02-11 22:58:06 in 8ms
Out[165]: 72

In [166]: len(y_true)
          Last executed 2018-02-11
Out[166]: 75
```

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],
```

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
```