Topics in Social Data Science

Week 8

Machine Learning 1

Agenda

Machine learning review

Model fitting

Model evaluation

Code example

Decision trees

Ensemble learning

Canonical example

Canonical example

Dog

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

Data point

Canonical example

Dog

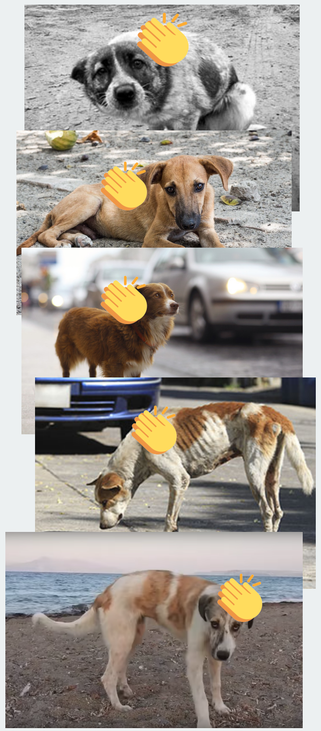
fluffy	sad looking	showing teeth	ears down	tail between legs	is retriever	growling
1	1	0	1	0	0	0

Data point

bites
0

Outcome

Canonical example



Experiments

$$
\begin{array}{c}
\begin{array}{ccccccc}
\textbf{fluffy} & \textbf{sad looking} & \textbf{showing teeth} & \textbf{ears down} & \textbf{tail between legs} & \textbf{is retriever} & \textbf{growling}
\end{array} \\
\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 1
\end{bmatrix}
\end{array}
\qquad
\begin{array}{c}
\textbf{bites} \\
\begin{bmatrix}
0 \\ 0 \\ 0 \\ 1 \\ 1
\end{bmatrix}
\end{array}
$$

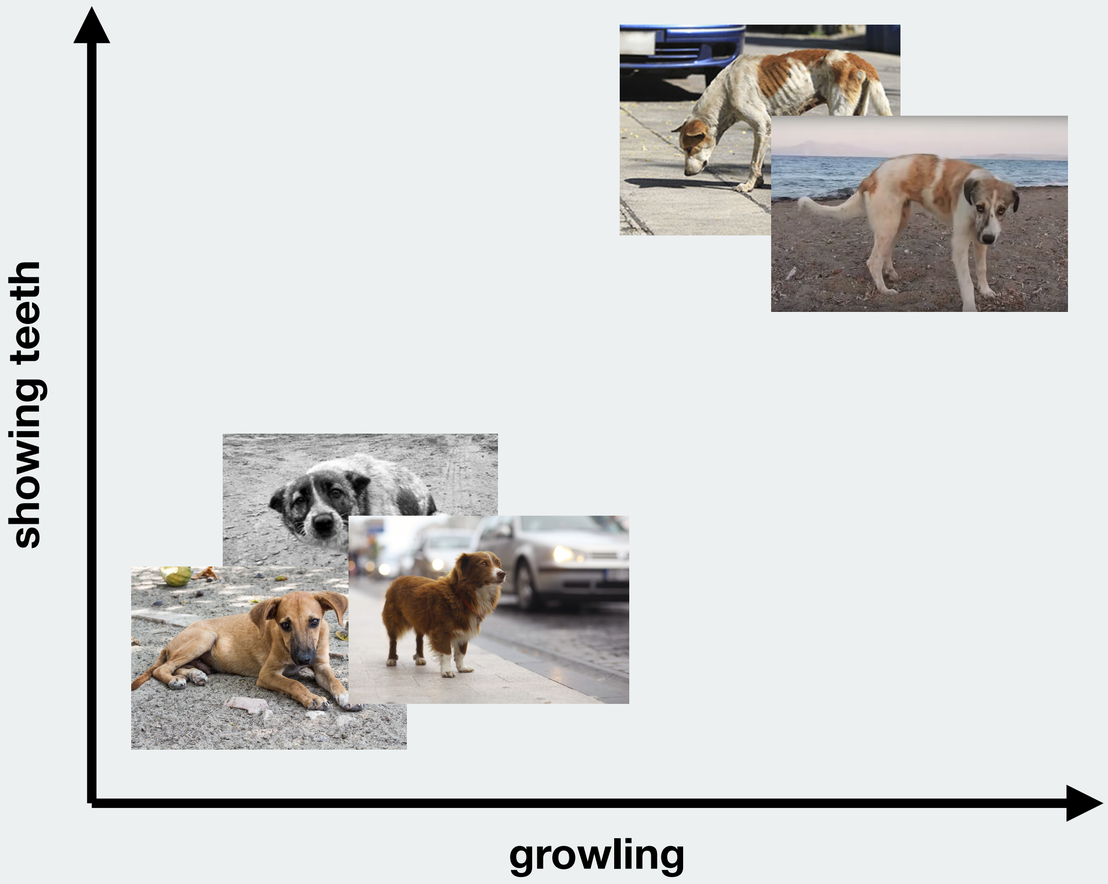
Data points

X

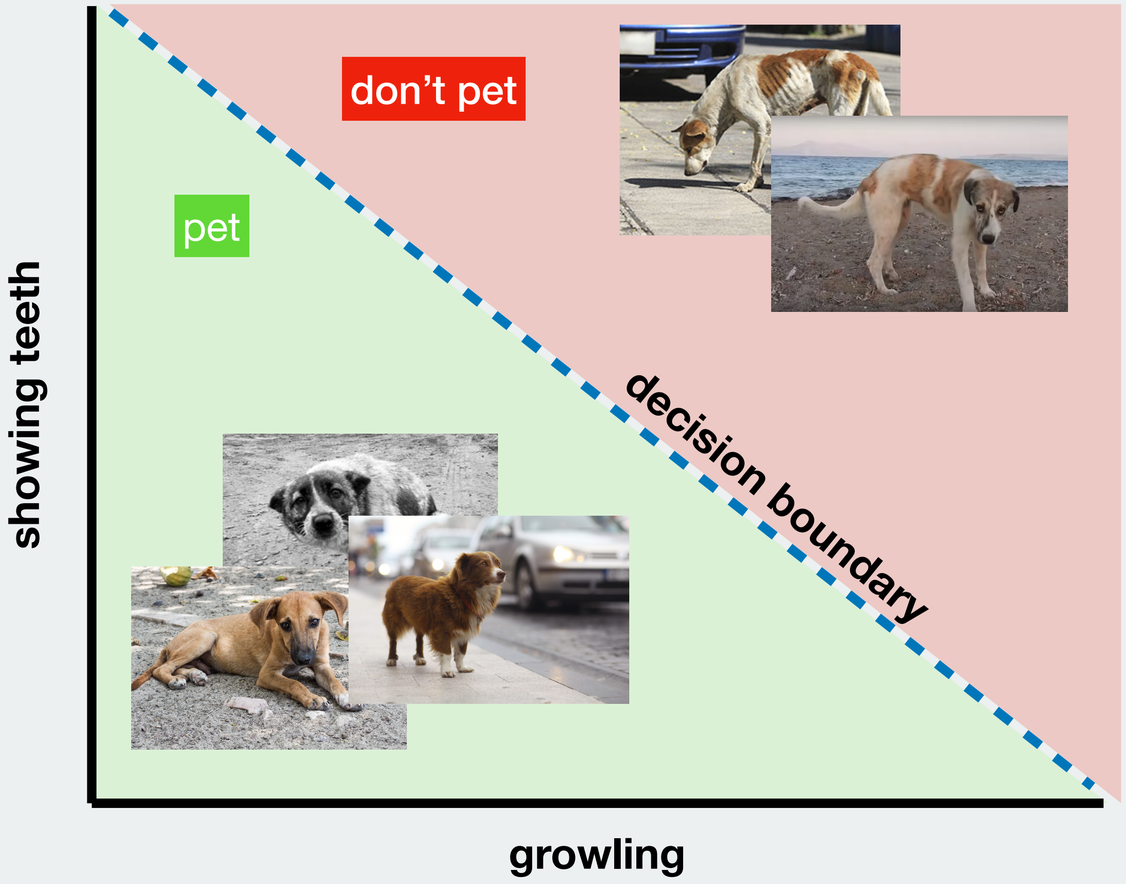
Outcomes

y

Canonical example



Canonical example



Types of machine learning

Types of machine learning

Supervised

When you have outcomes 👏

Unsupervised

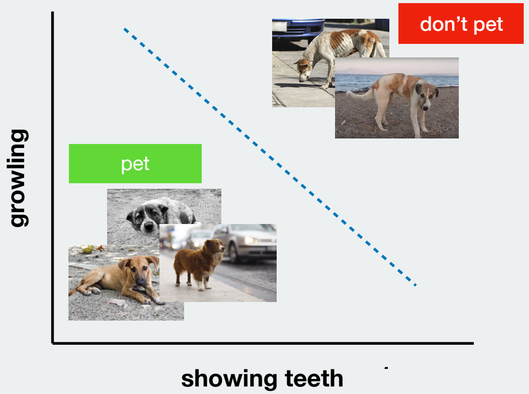
When you don't have outcomes 👏

Types of machine learning

Supervised

When you have outcomes 👏

- Classification



Unsupervised

When you don't have outcomes 🚫👏

Types of machine learning

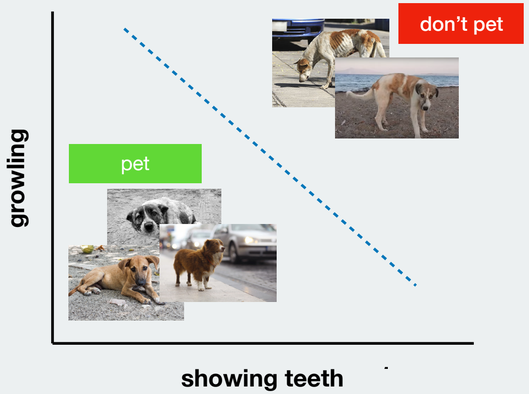
Supervised

When you have outcomes 👏

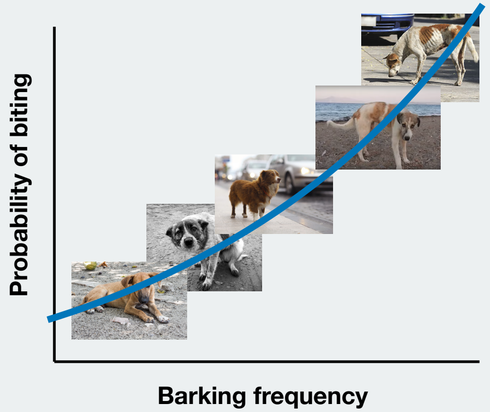
Unsupervised

When you don't have outcomes 🚫👏

- Classification



- Regression

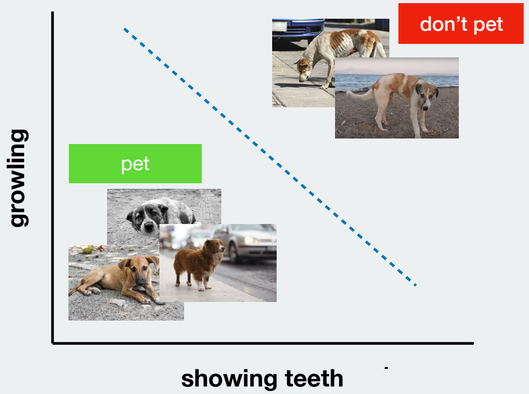


Types of machine learning

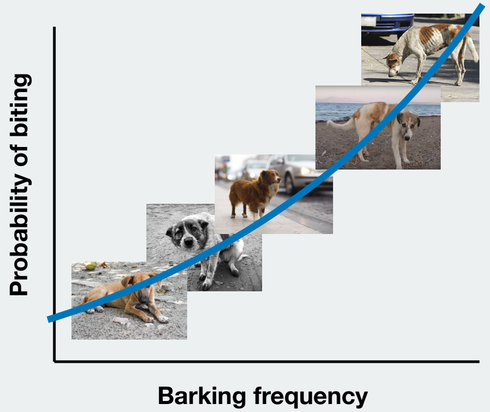
Supervised

When you have outcomes 👏

- Classification



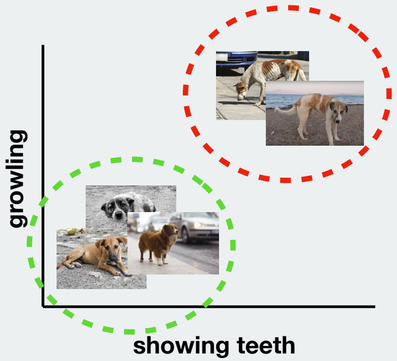
- Regression



Unsupervised

When you don't have outcomes 🚫👏

- Clustering

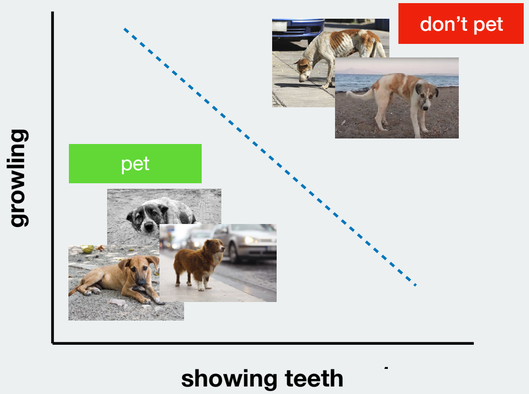


Types of machine learning

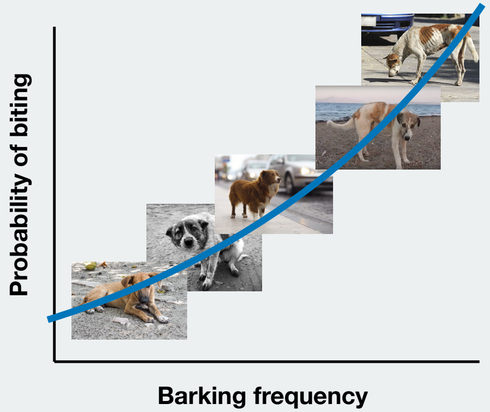
Supervised

When you have outcomes 👏

- Classification



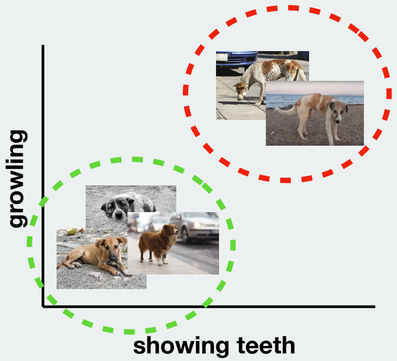
- Regression



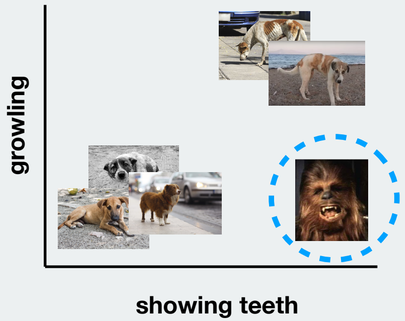
Unsupervised

When you don't have outcomes 🚫👏

- Clustering



- Outlier detection

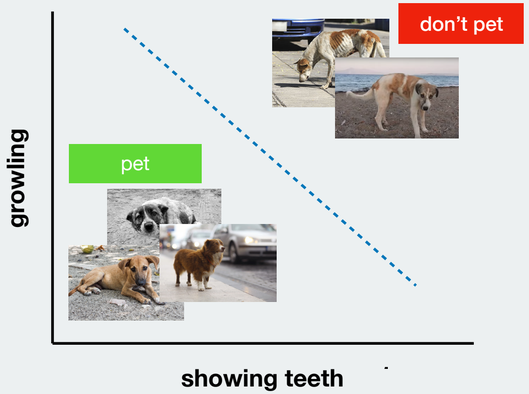


Types of machine learning

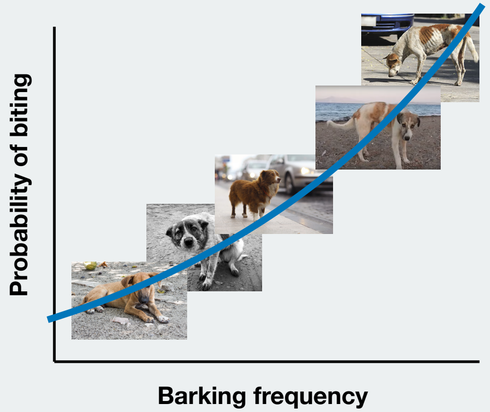
Supervised

When you have outcomes 👏

- Classification



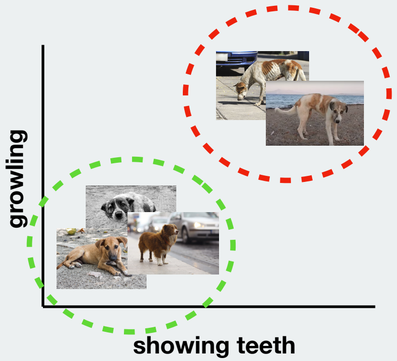
- Regression



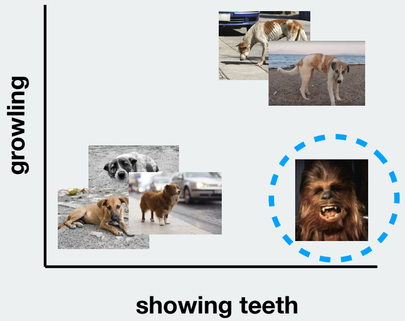
Unsupervised

When you don't have outcomes 🙅

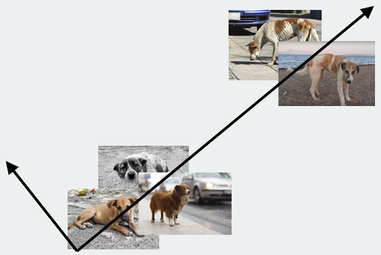
- Clustering



- Outlier detection



- Latent variable analysis

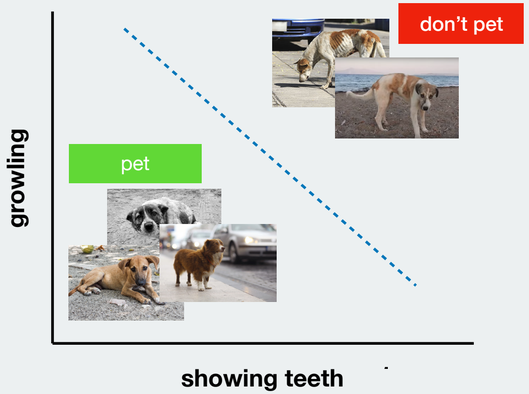


Types of machine learning

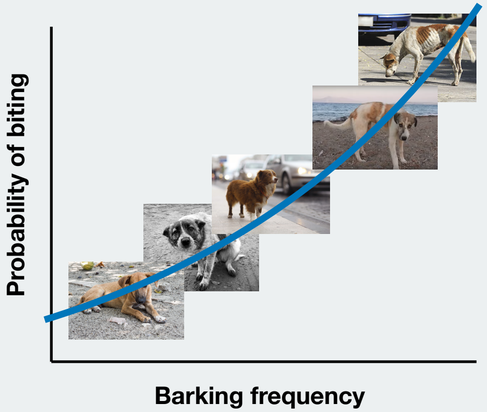
Supervised

When you have outcomes 👏

- Classification



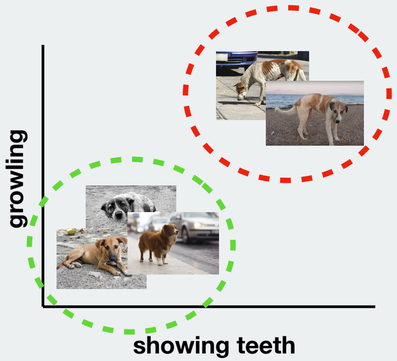
- Regression



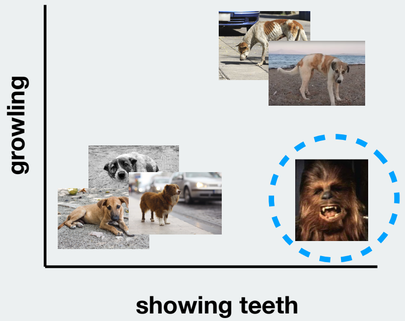
Unsupervised

When you don't have outcomes 🙅

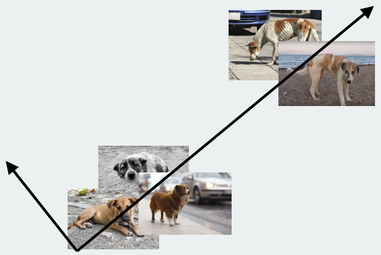
- Clustering



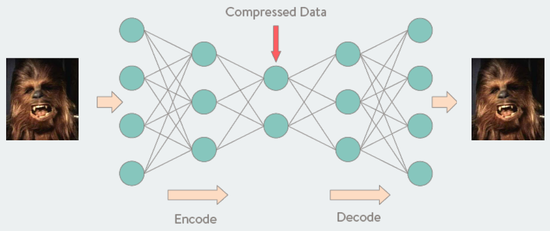
- Outlier detection



- Latent variable analysis

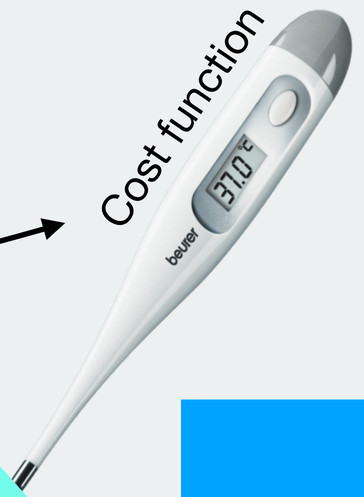


- Auto encoding



In a nutshell

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



Input → **model** → **Output**

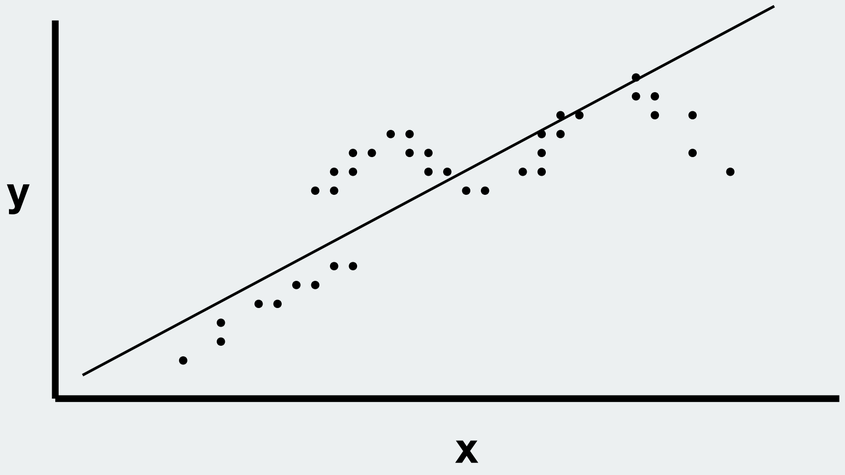
What is a model?

What is a model?

What is a model?

f(x) = a·x + b

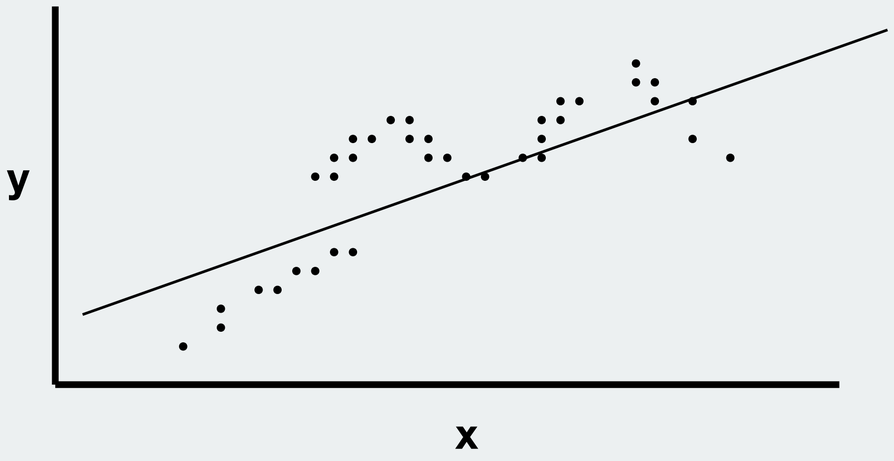
regression



What is a model?

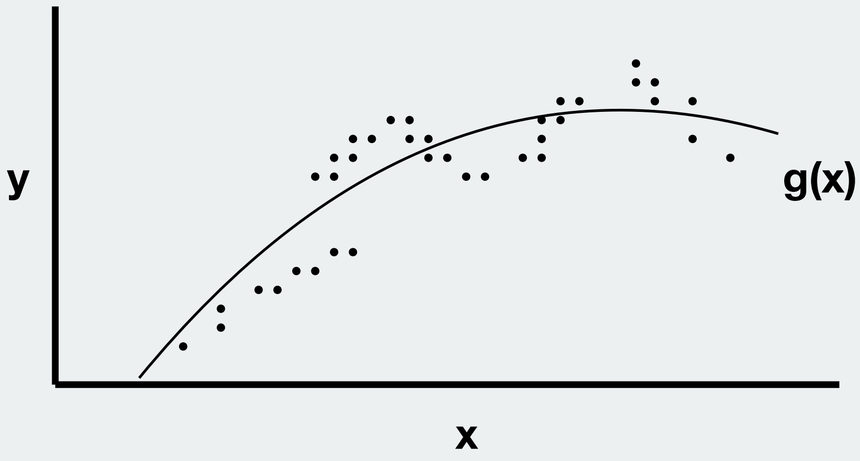
regression

$$f(x) = a \cdot x + b$$



What is a model?

regression

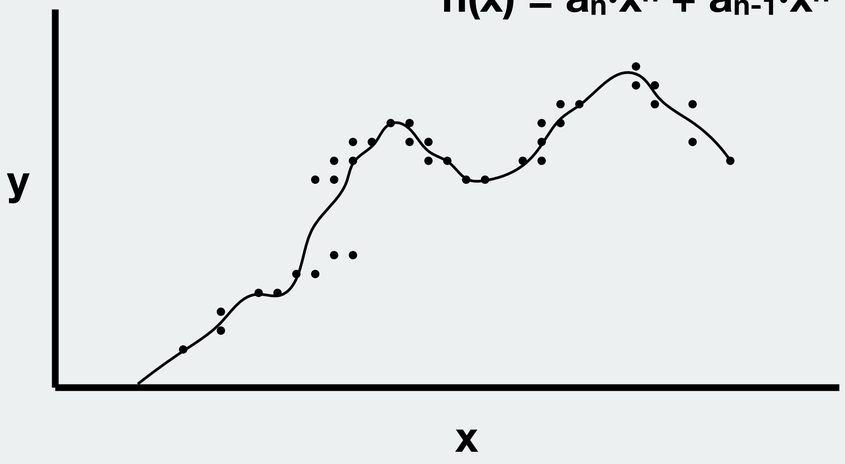


$g(x) = a \cdot x^2 + b \cdot x + c$

What is a model?

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

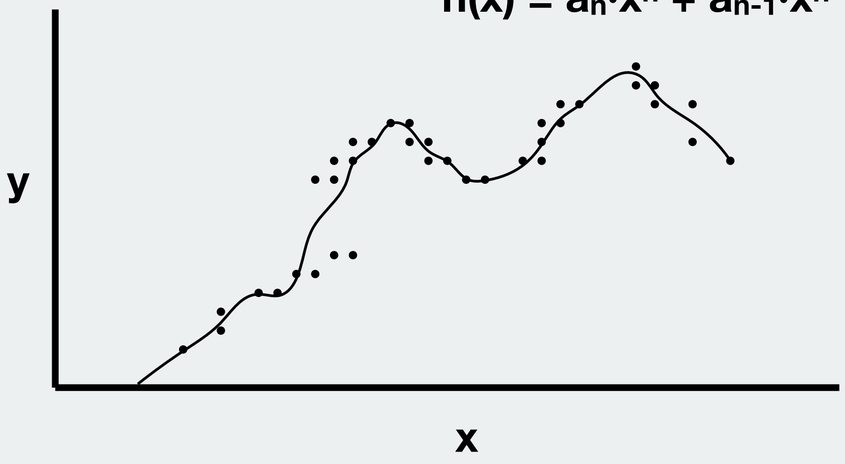
regression



What is a model?

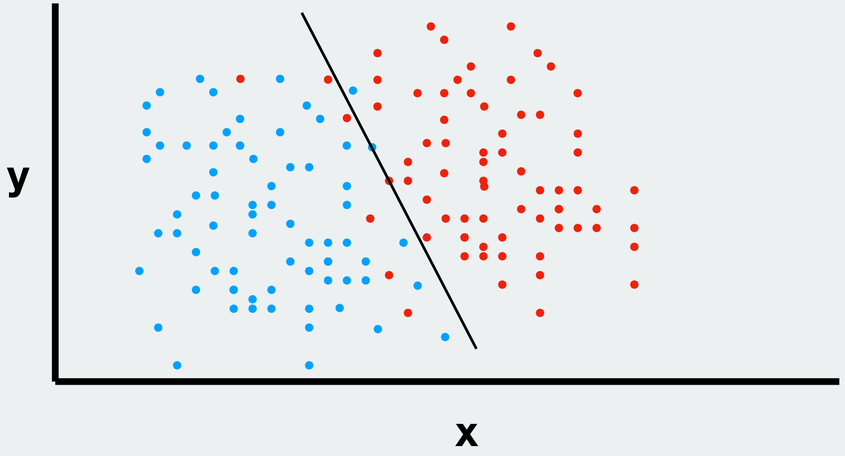
$$h(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_0 \cdot x^0$$

regression



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

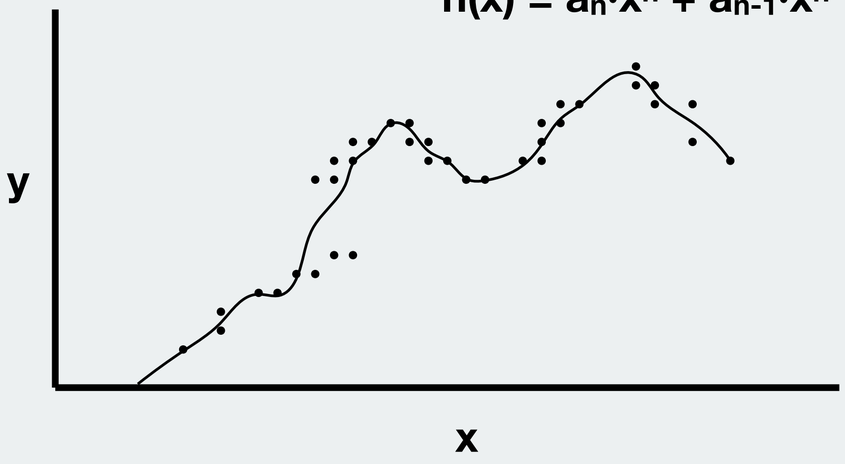
classification



What is a model?

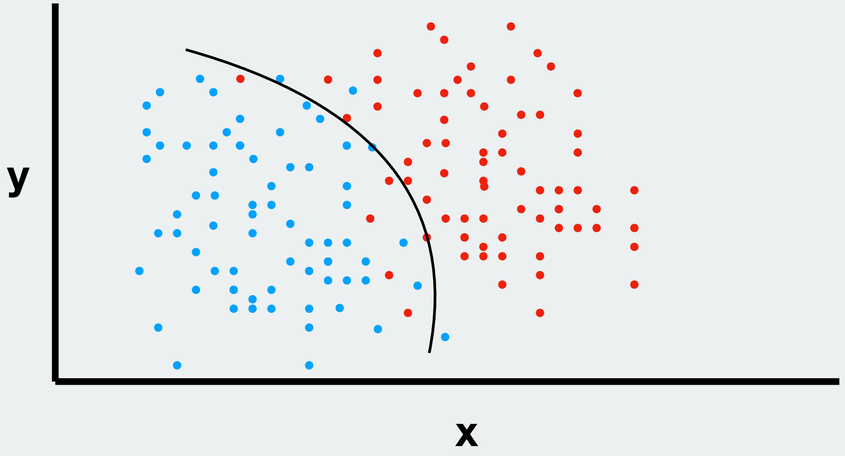
$$h(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_0 \cdot x^0$$

regression



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

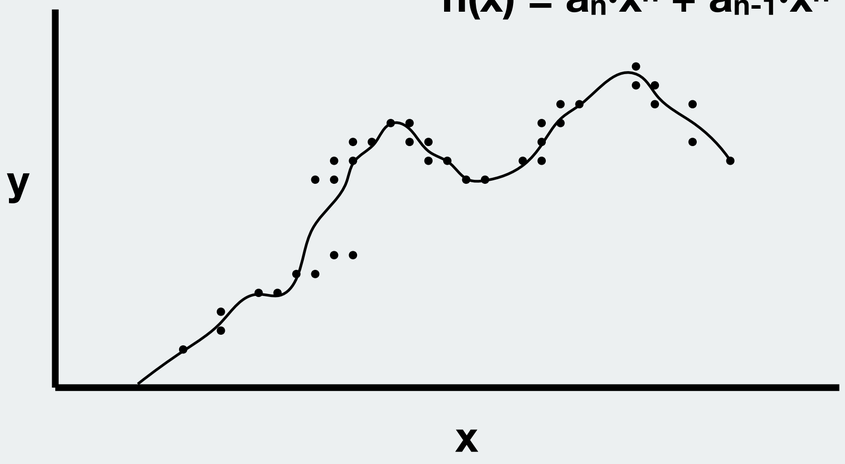
classification



What is a model?

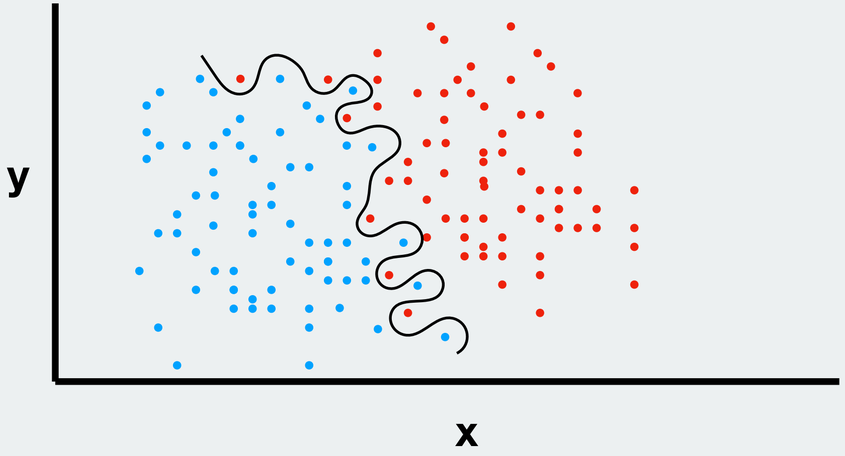
$$h(x) = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_0 \cdot x^0$$

regression



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

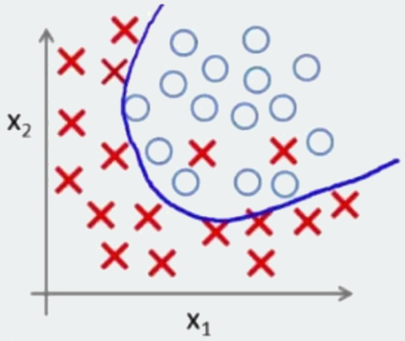
classification



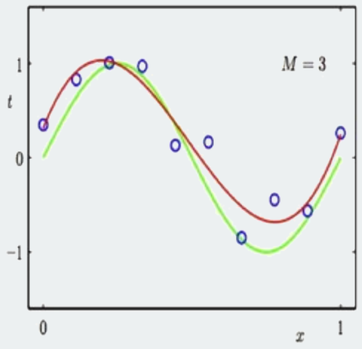
Fitting terminology

Fitting terminology

Classification



Regression



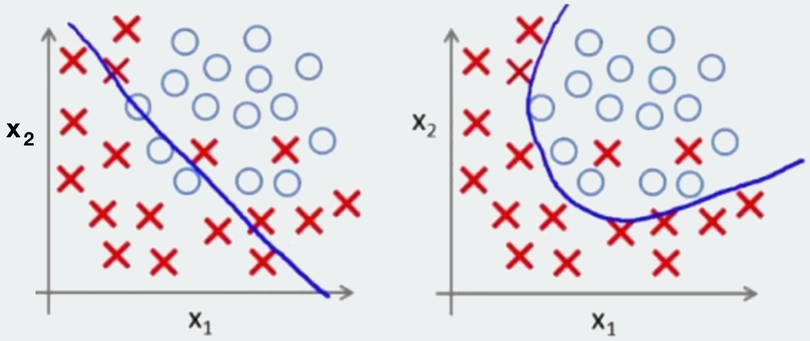
Acceptable train error

Acceptable test error

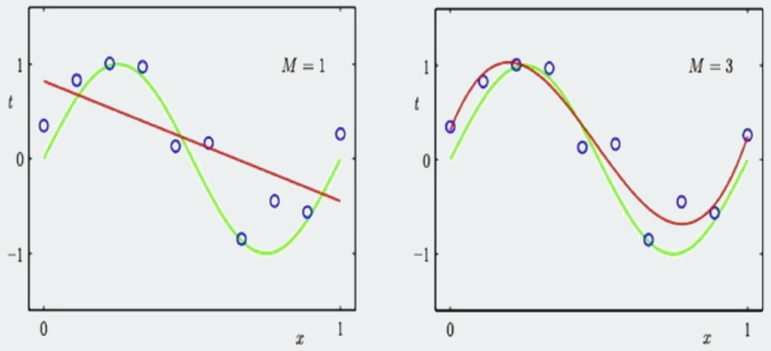
Fitting terminology

Underfitting

Classification



Regression



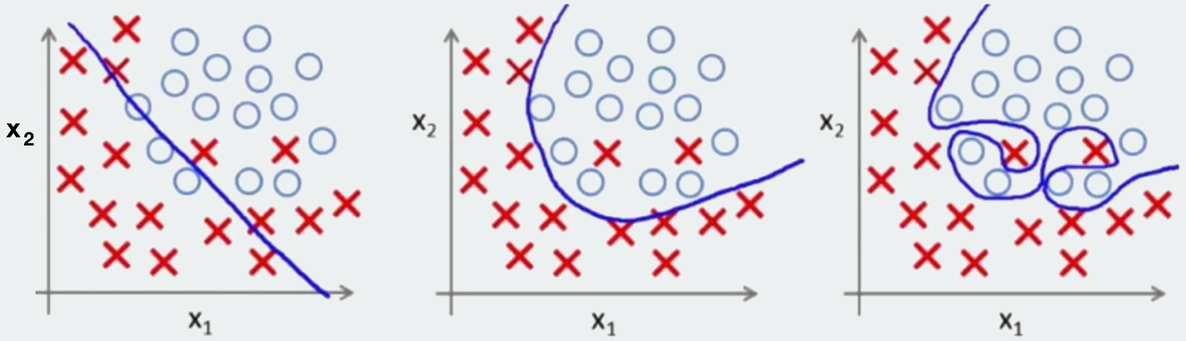
High train error **Acceptable train error**

High test error **Acceptable test error**

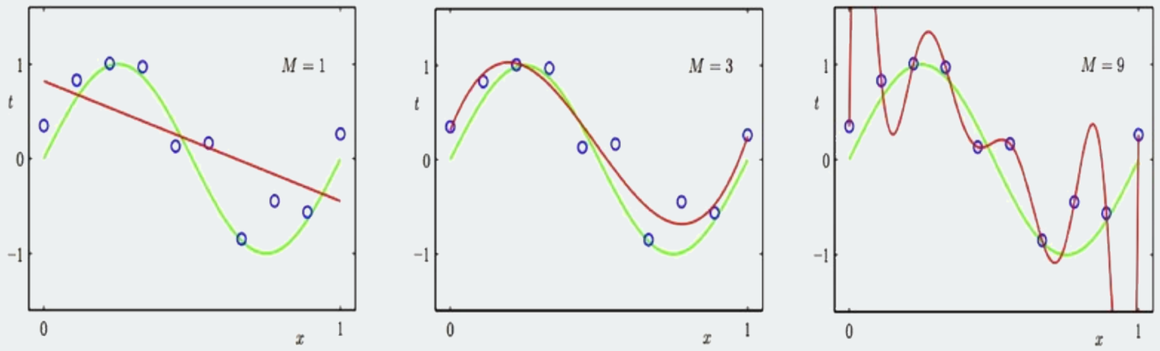
Fitting terminology

Underfitting **Overfitting**

Classification

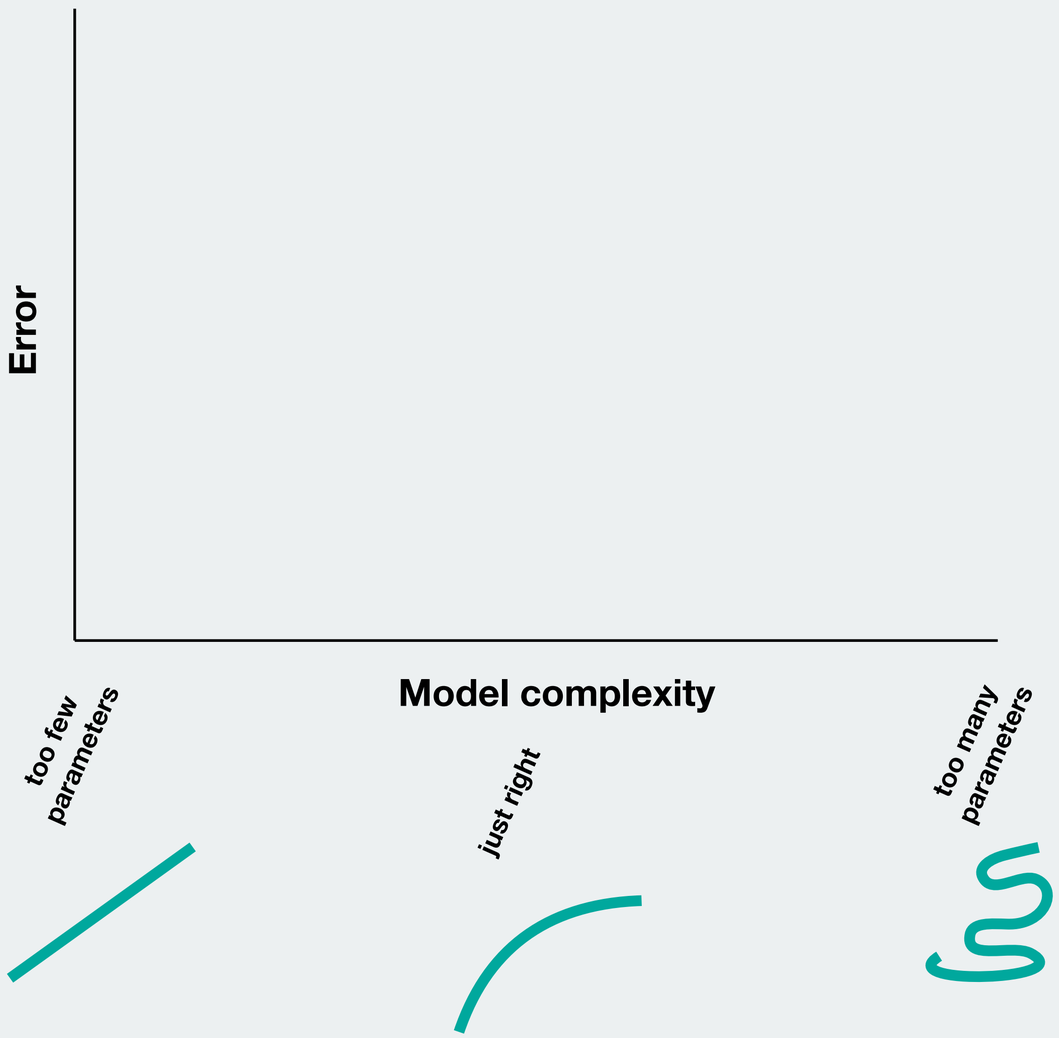


Regression

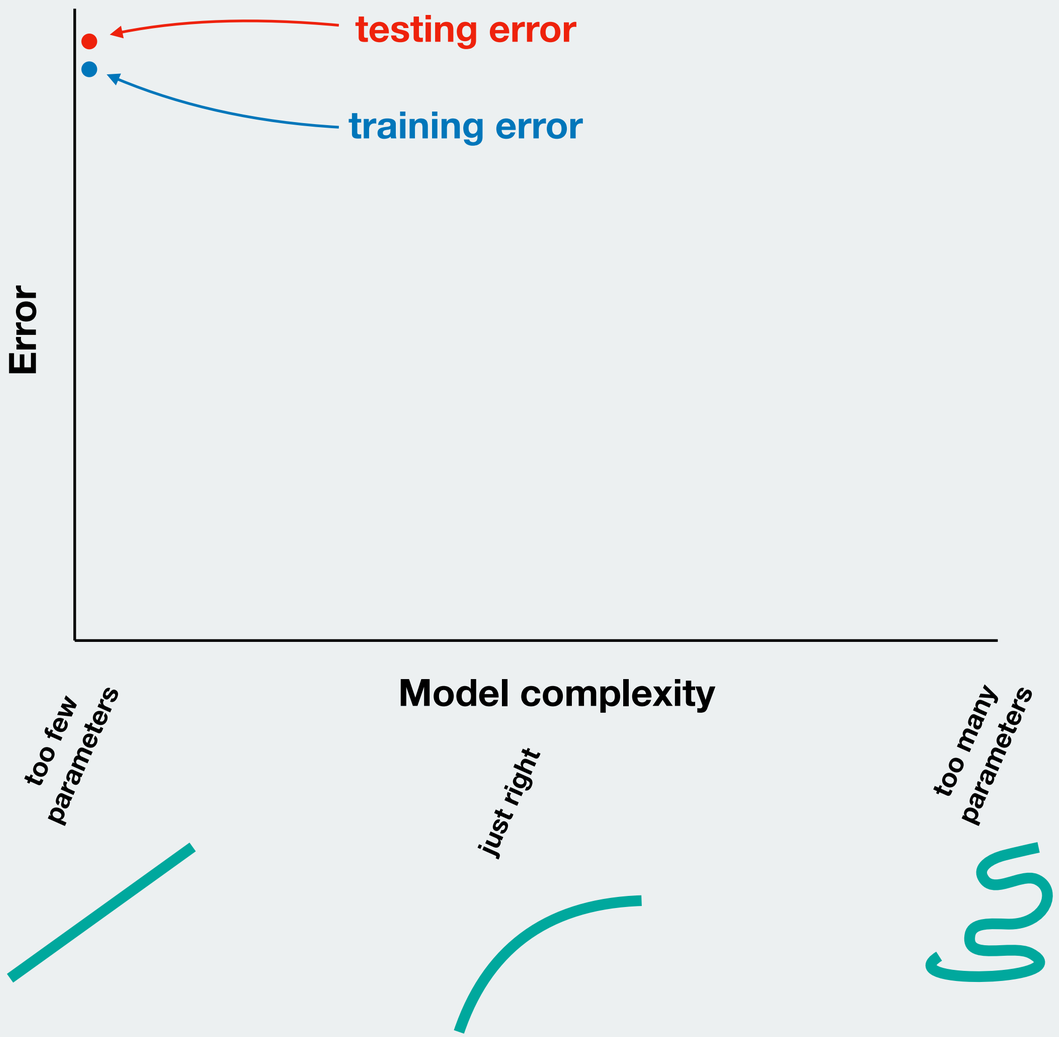


High train error	**Acceptable train error**	**Low train error**
High test error	**Acceptable test error**	**High test error**

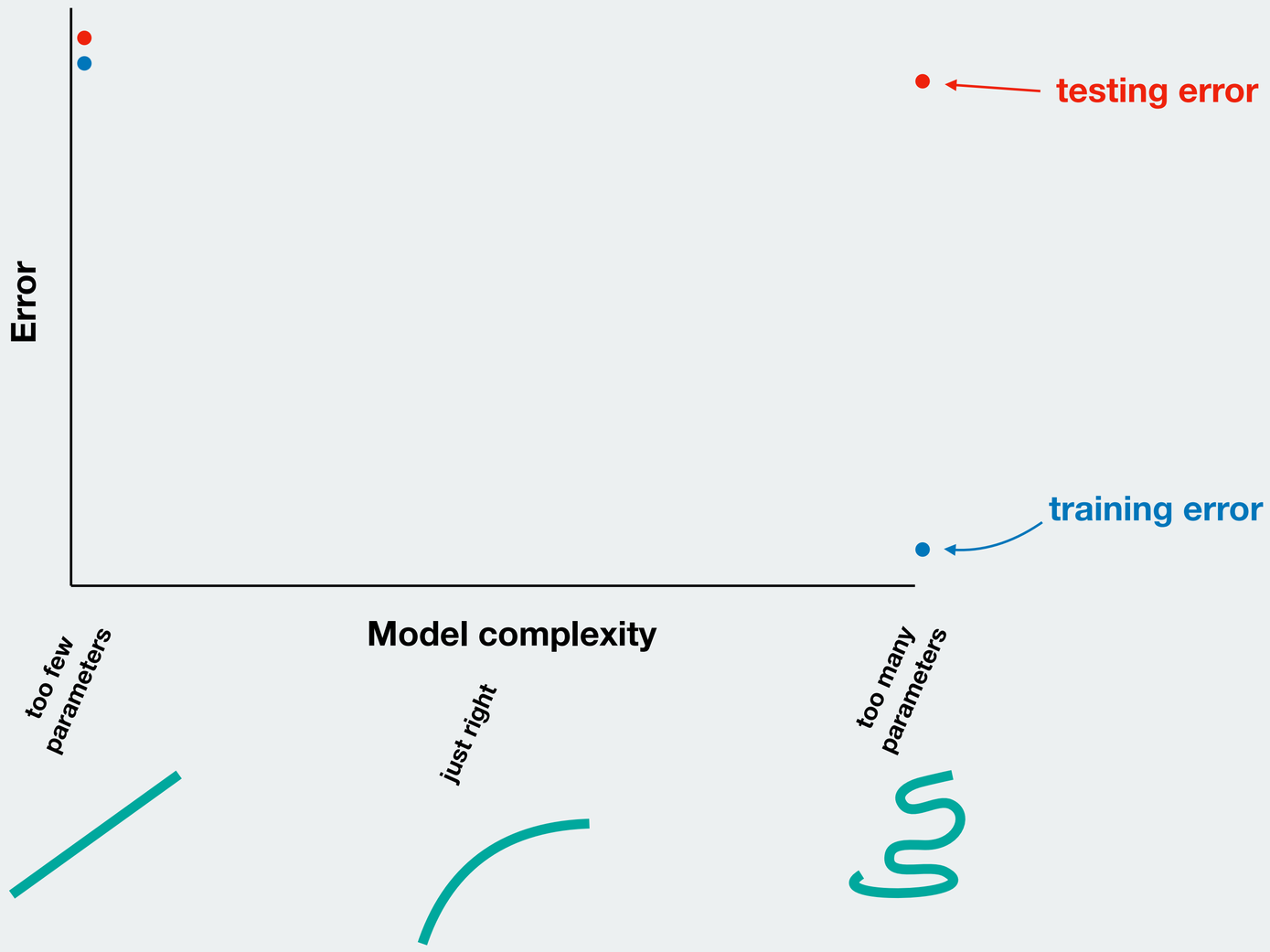
Training- vs. testing error



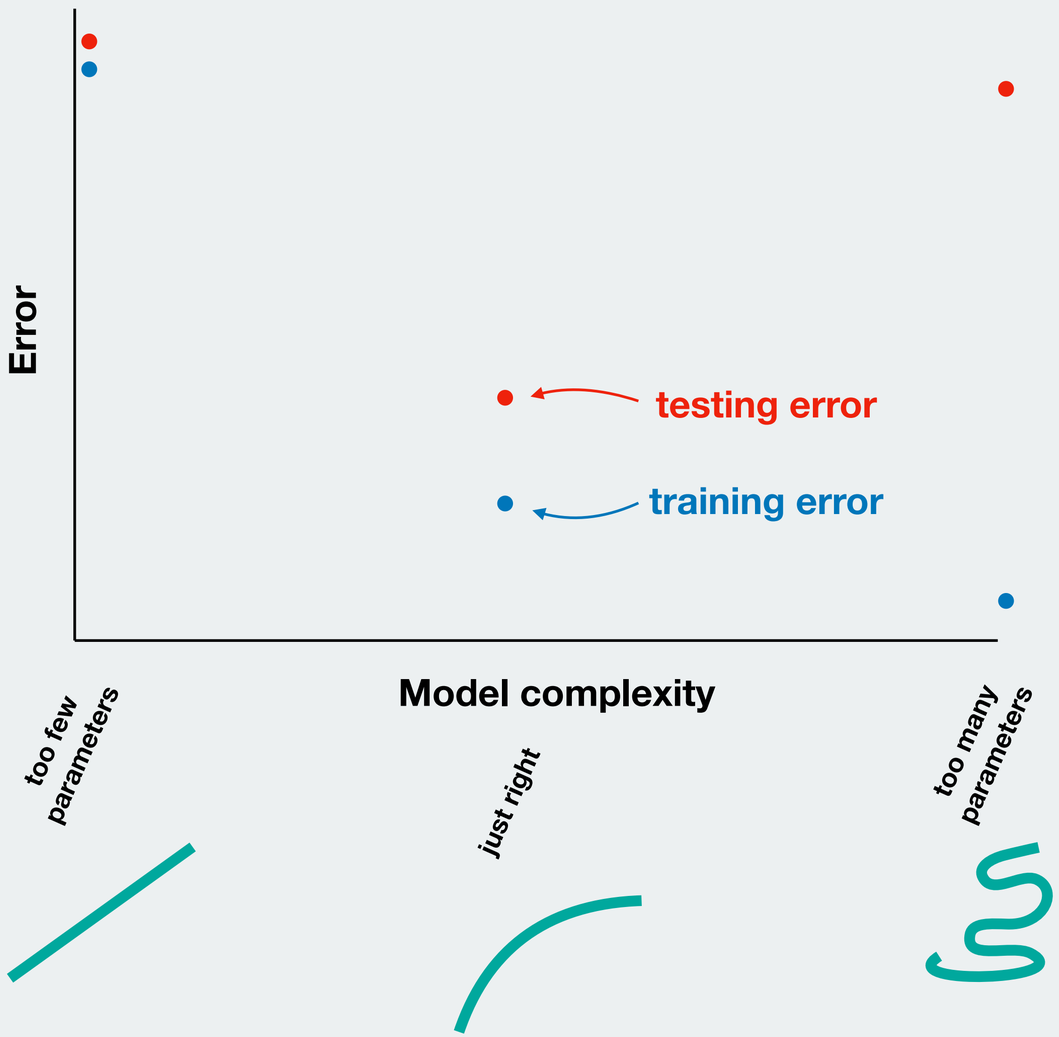
Training- vs. testing error



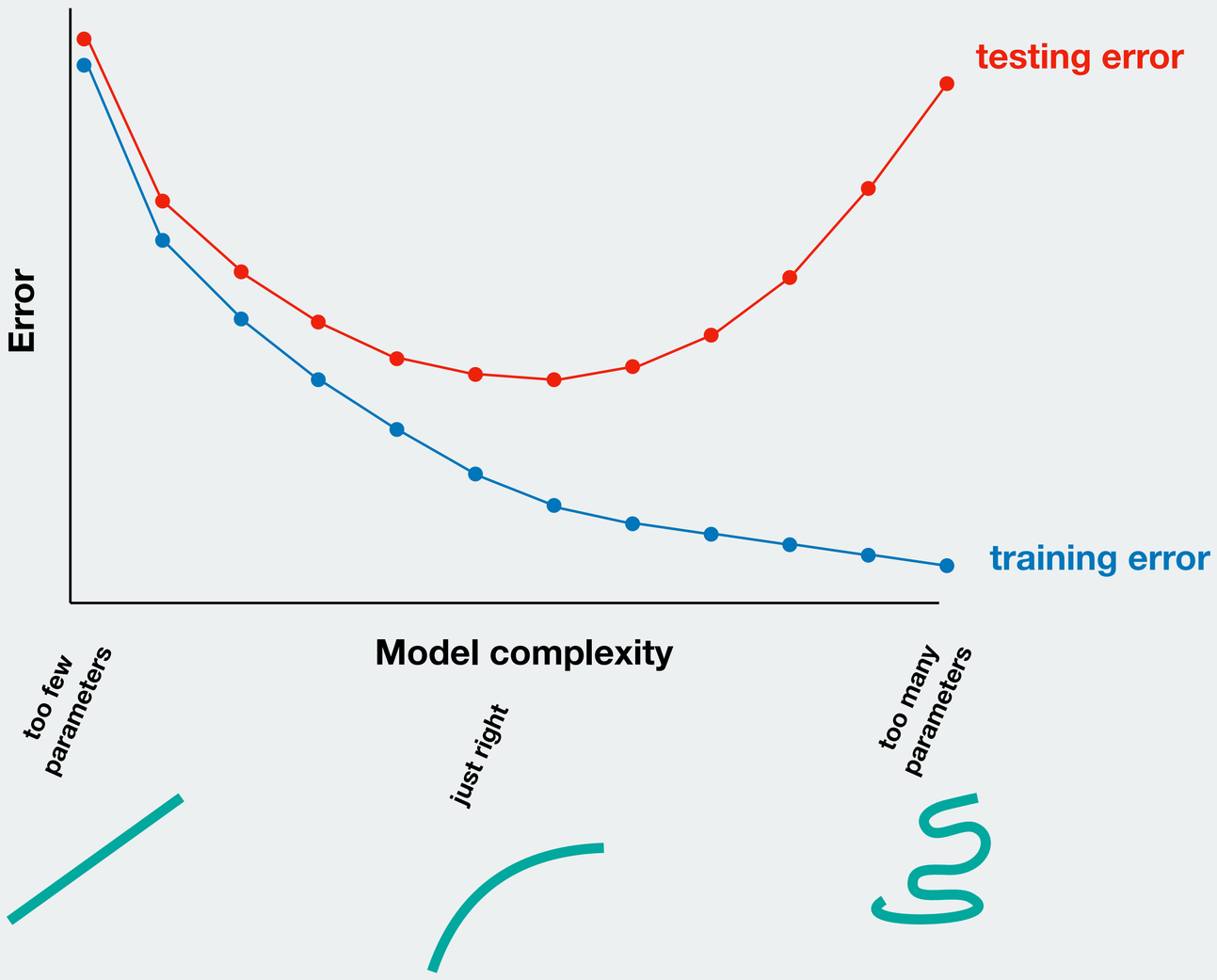
Training- vs. testing error



Training- vs. testing error



Training- vs. testing error

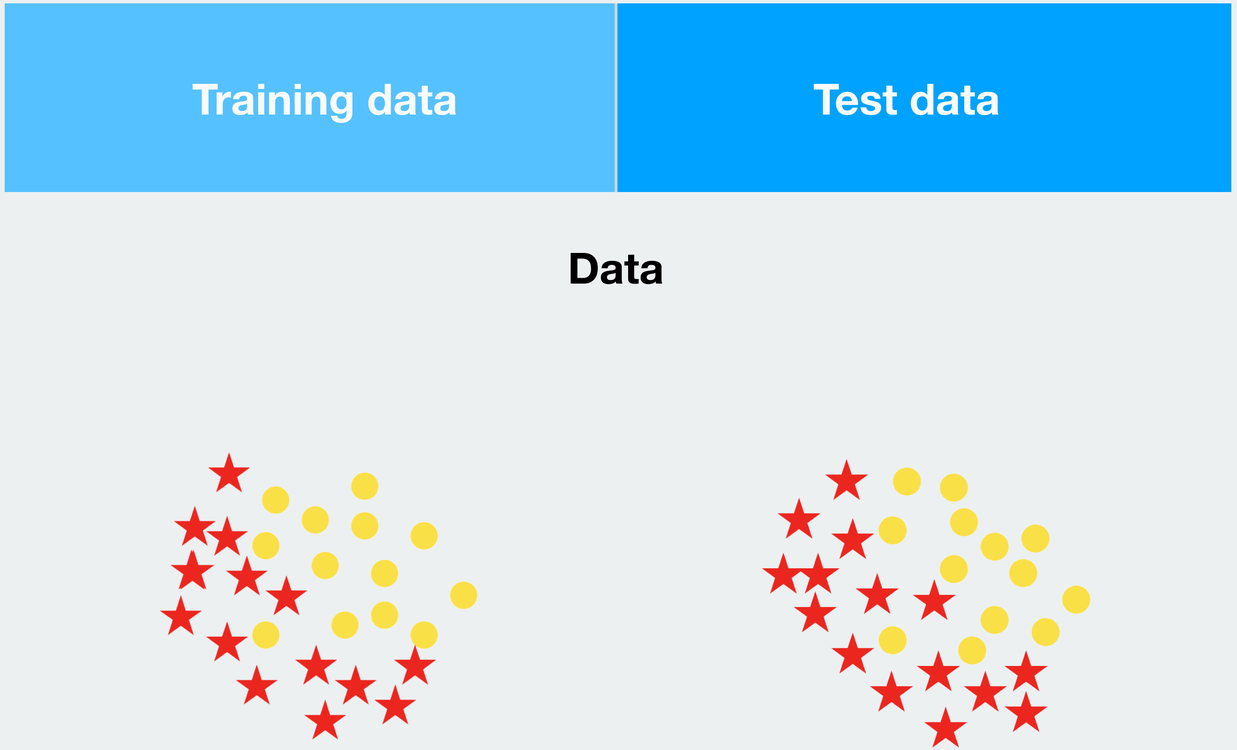


Model evaluation

Model evaluation

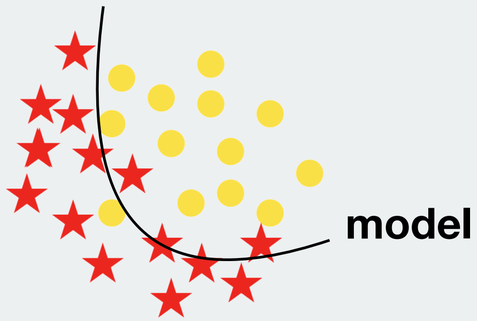
Data

Model evaluation



Model evaluation

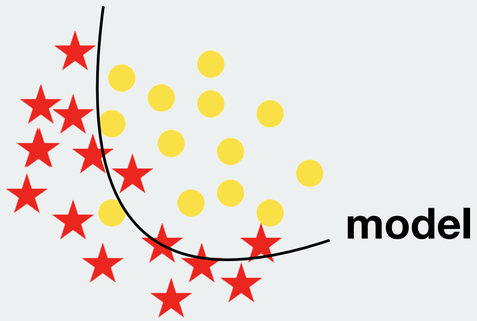
Data



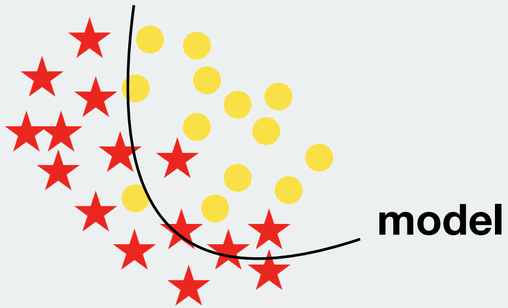
training error: 2 / 27 = **0.07**

Model evaluation

Data



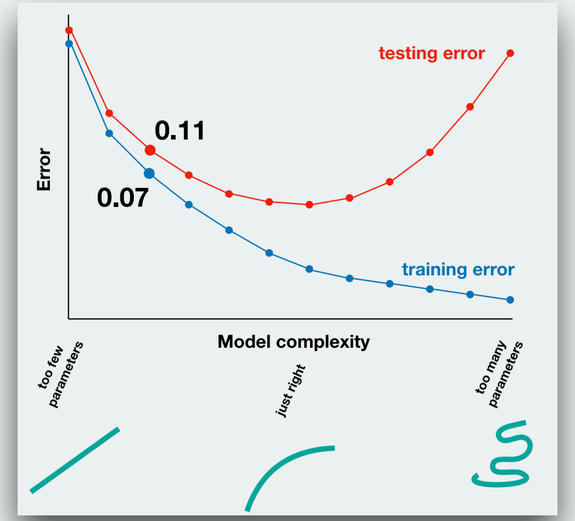
training error: 2 / 27 = **0.07**

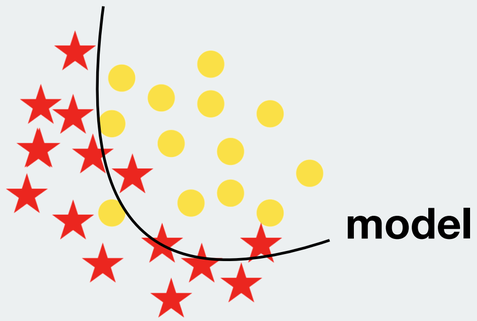


test error: 3 / 27 = **0.11**

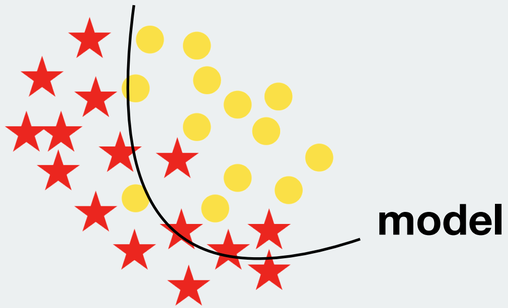
Model evaluation

Data





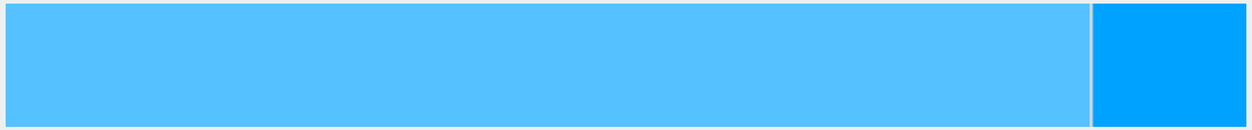
training error: 2 / 27 = **0.07**



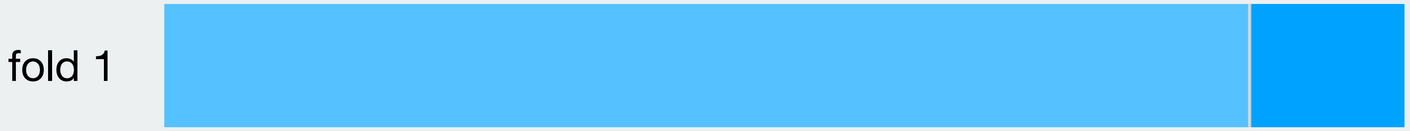
test error: 3 / 27 = **0.11**

Cross validation

Cross validation

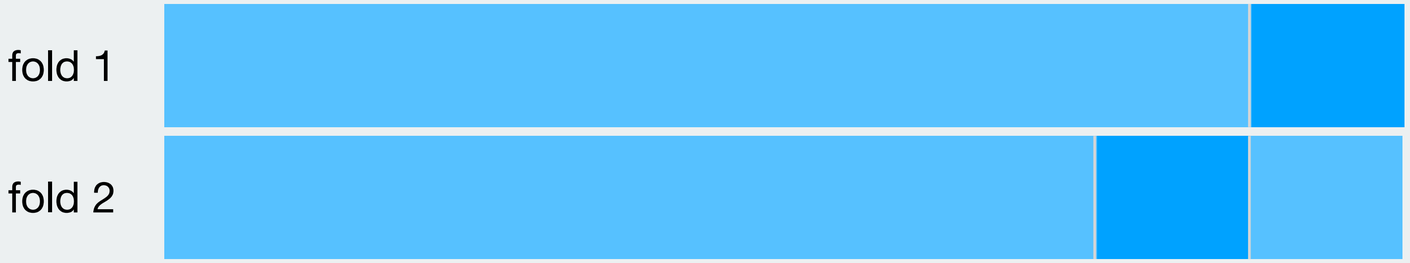


Cross validation



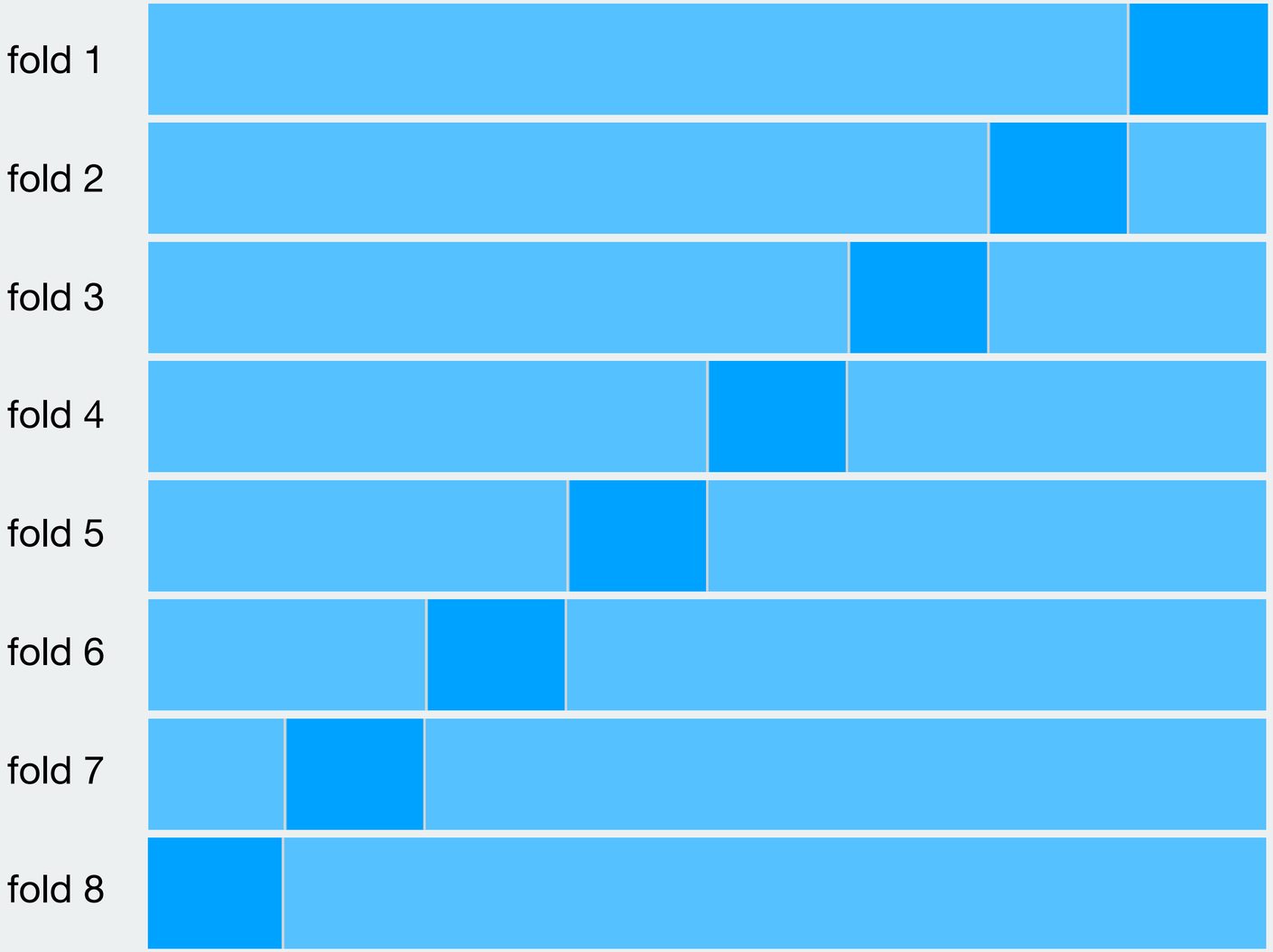
Train err.	Test err.
0.07	0.11

Cross validation



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

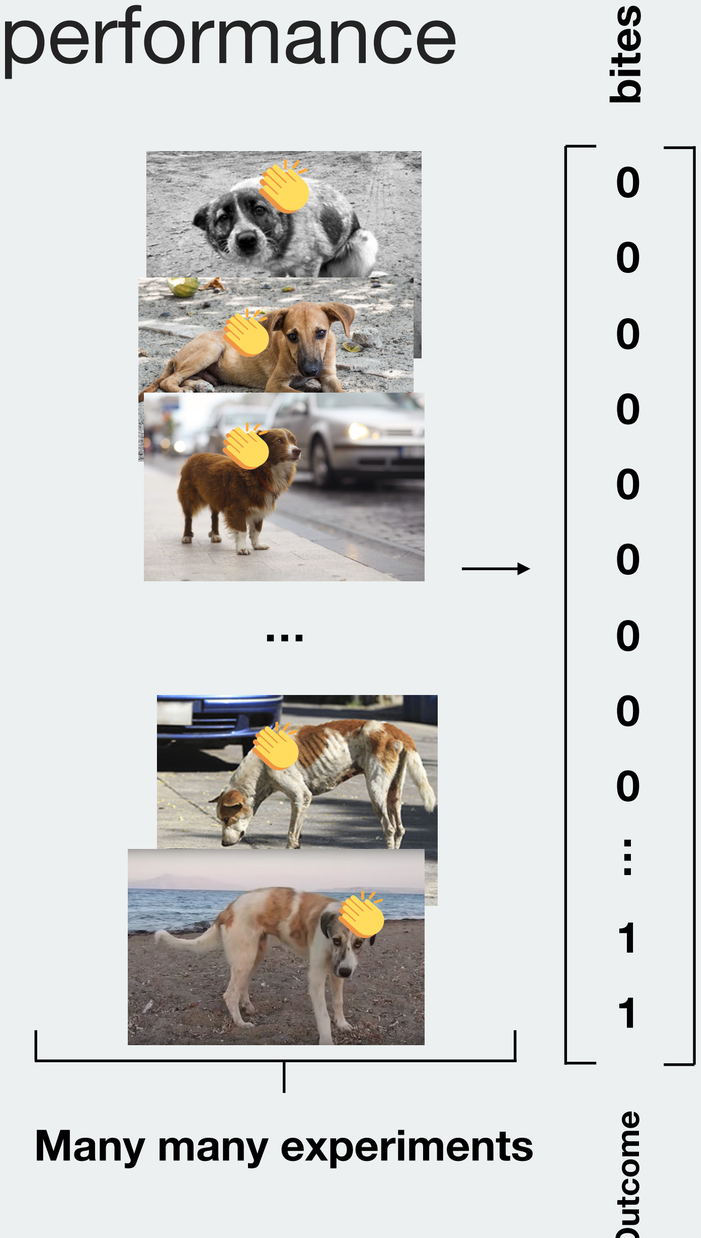
Cross validation



	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

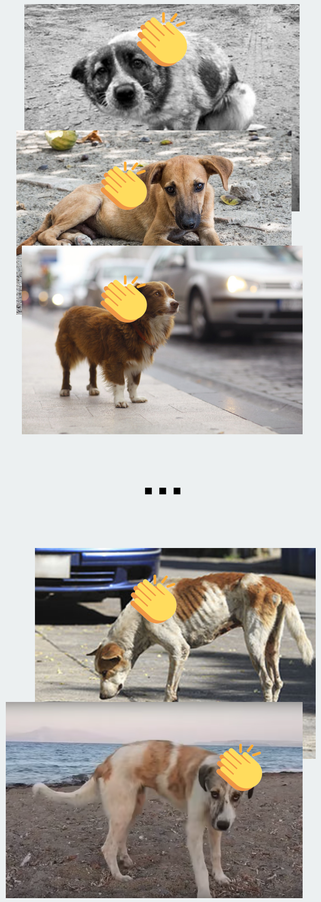
Model performance



Model performance

accuracy = 1 - error

Model performance



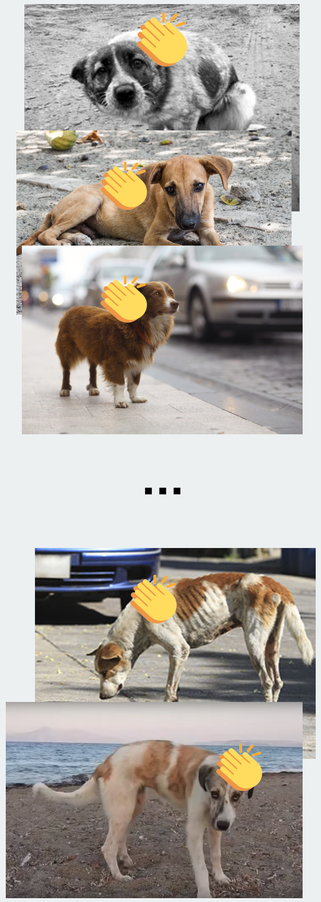
...

Many many experiments

bites (Outcome)	bites (Prediction)
0	0
0	0
0	1
0	0
0	0
0	0
0	0
0	0
0	0
...	...
1	0
1	0

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

Model performance



...

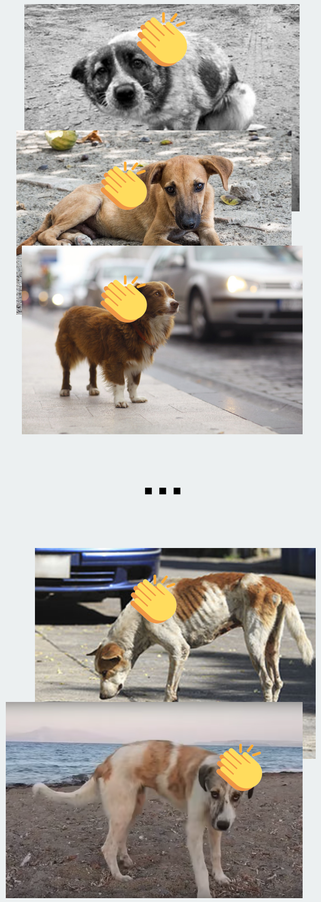
Many many experiments

bites (Outcome)	bites (Prediction)
0	0
0	0
0	1
0	0
0	0
0	0
0	0
0	0
0	0
...	...
1	0
1	0

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

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

Model performance



...

Many many experiments

bites (Outcome)	bites (Prediction)
0	0
0	0
0	1
0	0
0	0
0	0
0	0
0	0
0	0
...	...
1	0
1	0

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

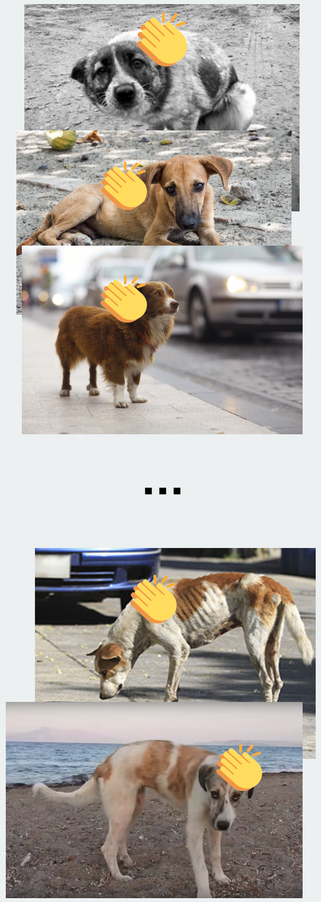
Dogs named Coco:	1%
Dogs that bite:	2%
Coco & bites:	0.02%

Accuracy:

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

high!

Model performance



...

Many many experiments

bites (Outcome)	bites (Prediction)
0	0
0	0
0	1
0	0
0	0
0	0
0	0
0	0
0	0
...	...
1	0
1	0

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

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

Accuracy:

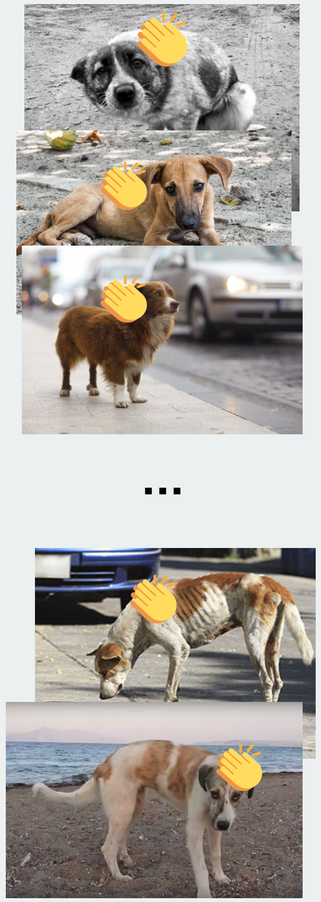
$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{200 + 970200}{1000000} = 97\%$$

high!

Confusion matrix:

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

Model performance



...

Many many experiments

bites (Outcome)	bites (Prediction)
0	0
0	0
0	1
0	0
0	0
0	0
0	0
0	0
0	0
⋮	⋮
1	0
1	0

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

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

Accuracy:

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

high!

Confusion matrix:

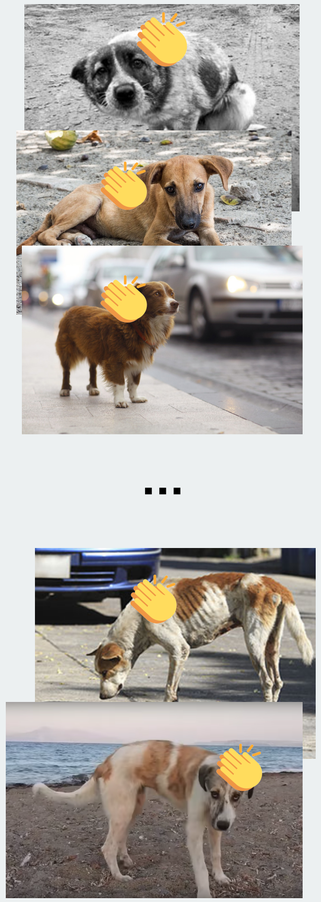
Prediction \ Outcome	Bite	No bite
Bite	TP: 200	FP: 9800
No bite	FN: 19800	TN: 970200

Precision & recall

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

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

Model performance



...

Many many experiments

bites (Outcome)	bites (Prediction)
0	0
0	0
0	1
0	0
0	0
0	0
0	0
0	0
0	0
⋮	⋮
1	0
1	0

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

Dogs named Coco:	1%
Dogs that bite:	2%
Coco & 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

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

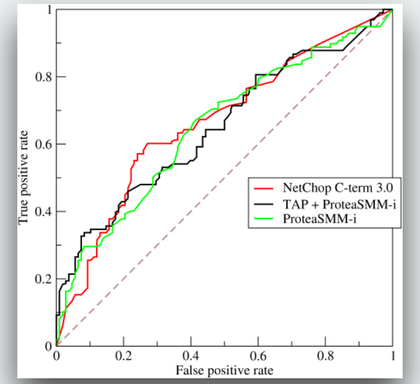
$$recall = \frac{TP}{TP + FN} = \frac{200}{200 + 19800} = 1\%$$

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],
```

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],
```

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],
```

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],
```

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],
```

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],
                 [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],
```

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],
```

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],
```

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],
```

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],
                 [
```

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],
                 [
```

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],
```

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],
                 [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],
                 [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],
                 [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],
```

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],
```

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

```
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],
```

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
```

```
In [166]: 1 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],
                 [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
```

Decision trees

Decision trees

Lays eggs	Cold blooded	Mammal
0	0	1
0	0	1
0	0	1
0	0	1
0	0	1
1	0	0
1	0	0
1	1	0
1	1	0
1	1	0
1	1	0
1	1	0
1	1	0
1	1	0
1	1	0

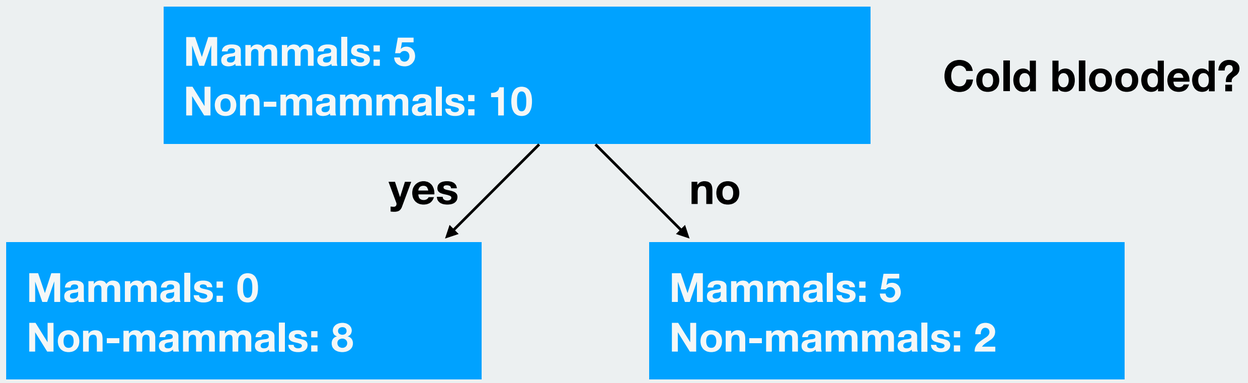
features **target**

Mammals: 5
Non-mammals: 10

Decision trees

Lays eggs	Cold blooded	Mammal
0	0	1
0	0	1
0	0	1
0	0	1
0	0	1
1	0	0
1	0	0
1	1	0
1	1	0
1	1	0
1	1	0
1	1	0
1	1	0
1	1	0
1	1	0
1	1	0

features **target**

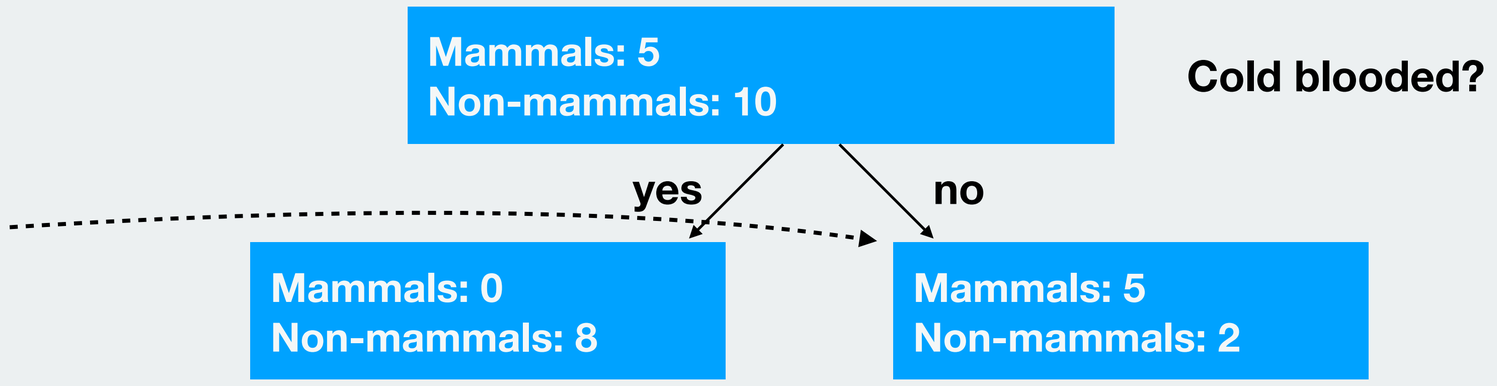


Decision trees

	Lays eggs	Cold blooded	Mammal
no	0	0	1
	0	0	1
	0	0	1
	0	0	1
	0	0	1
	1	0	0
	1	0	0
yes	1	1	0
	1	1	0
	1	1	0
	1	1	0
	1	1	0
	1	1	0
	1	1	0
	1	1	0

features

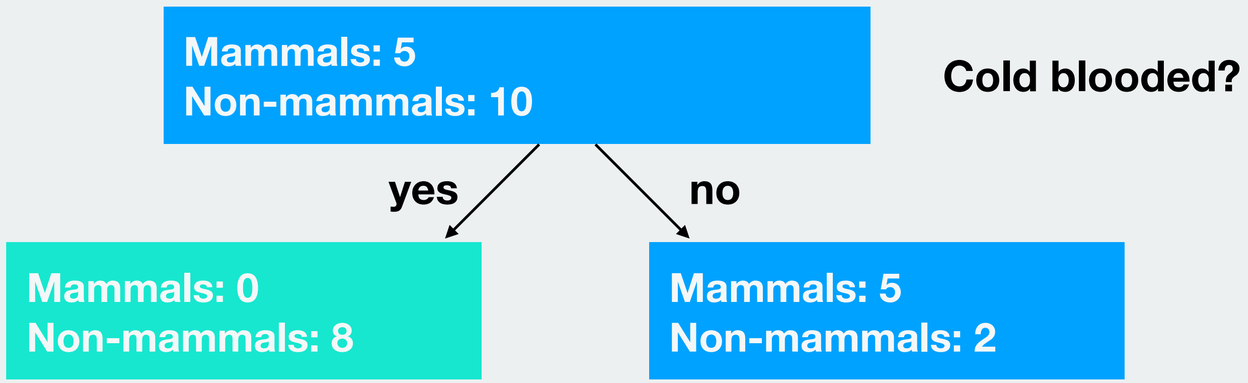
target



Decision trees

	Lays eggs	Cold blooded	Mammal
no	0	0	1
	0	0	1
	0	0	1
	0	0	1
	0	0	1
	1	0	0
	1	0	0
yes	1	1	0
	1	1	0
	1	1	0
	1	1	0
	1	1	0
	1	1	0
	1	1	0
	1	1	0

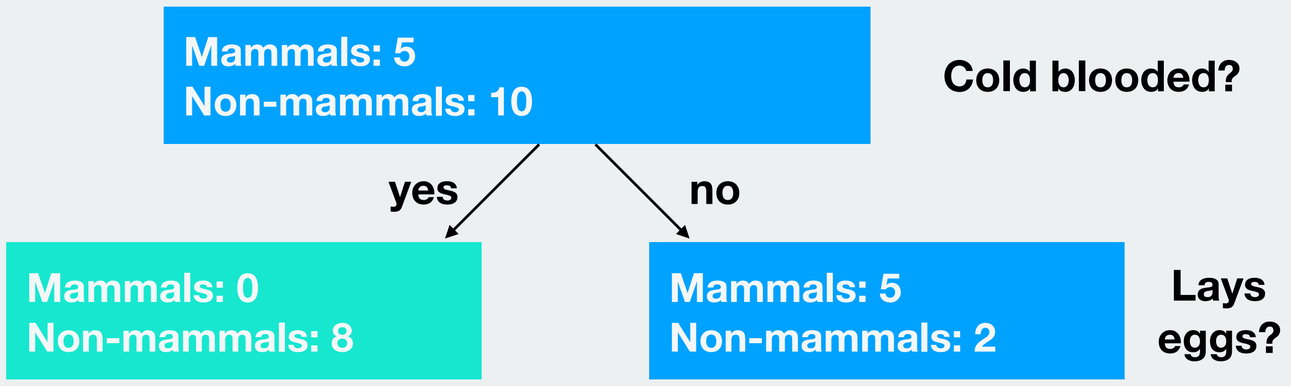
features **target**



Decision trees

	Lays eggs	Cold blooded	Mammal
no	0	0	1
	0	0	1
	0	0	1
	0	0	1
	0	0	1
	1	0	0
	1	0	0
yes	1	1	0
	1	1	0
	1	1	0
	1	1	0
	1	1	0
	1	1	0
	1	1	0
	1	1	0

features **target**

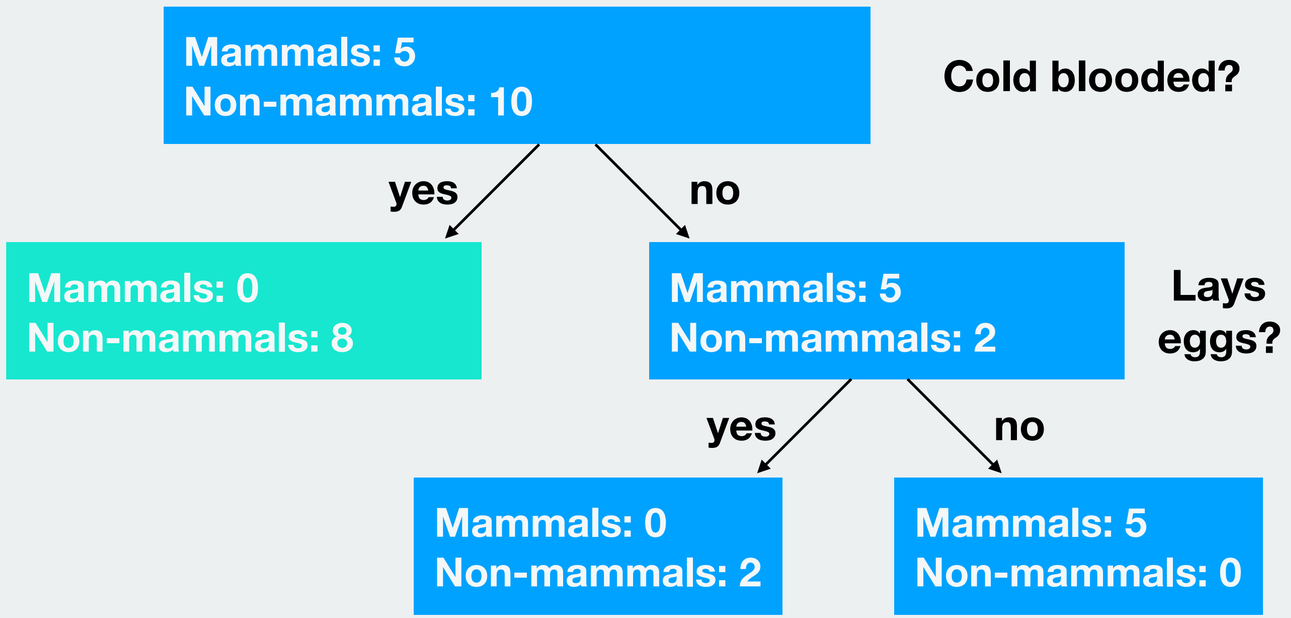


Decision trees

	Lays eggs	Cold blooded	Mammal
no	0	0	1
no	0	0	1
no	0	0	1
no	0	0	1
no	0	0	1
yes	1	0	0
yes	1	0	0
yes	1	1	0
yes	1	1	0
yes	1	1	0
yes	1	1	0
yes	1	1	0
yes	1	1	0
yes	1	1	0
yes	1	1	0

features

target

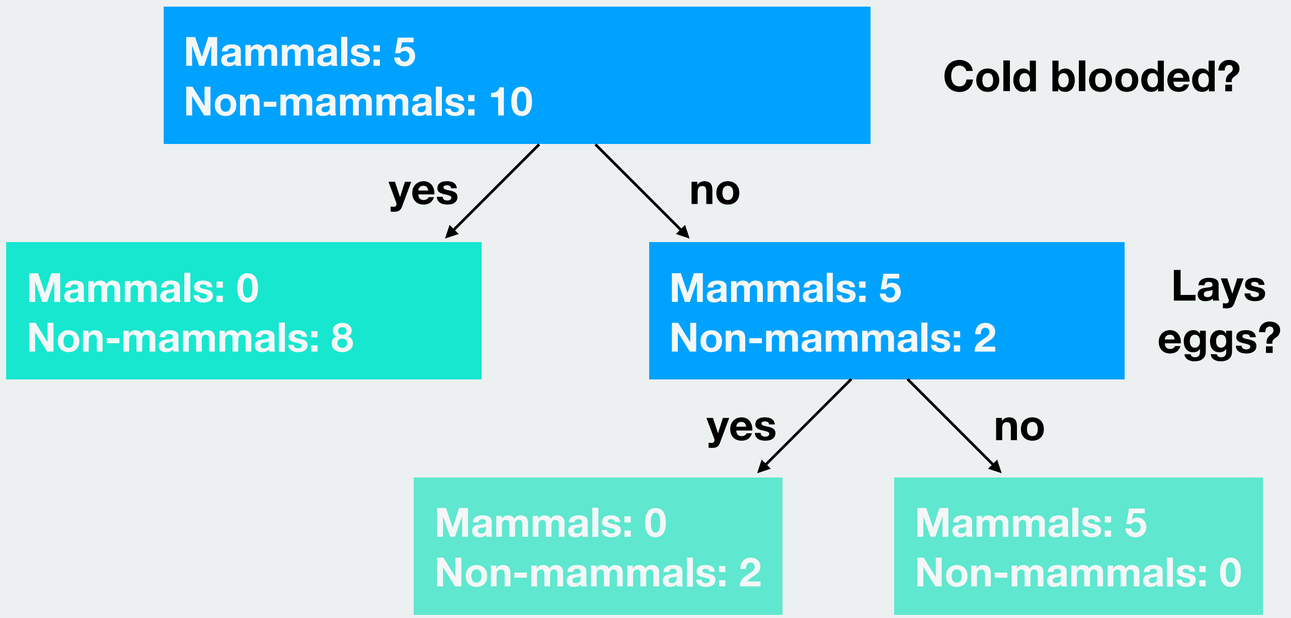


Decision trees

	Lays eggs	Cold blooded	Mammal
no	0	0	1
no	0	0	1
no	0	0	1
no	0	0	1
no	0	0	1
yes	1	0	0
yes	1	0	0
yes	1	1	0
yes	1	1	0
yes	1	1	0
yes	1	1	0
yes	1	1	0
yes	1	1	0
yes	1	1	0
yes	1	1	0

features

target



Decision trees

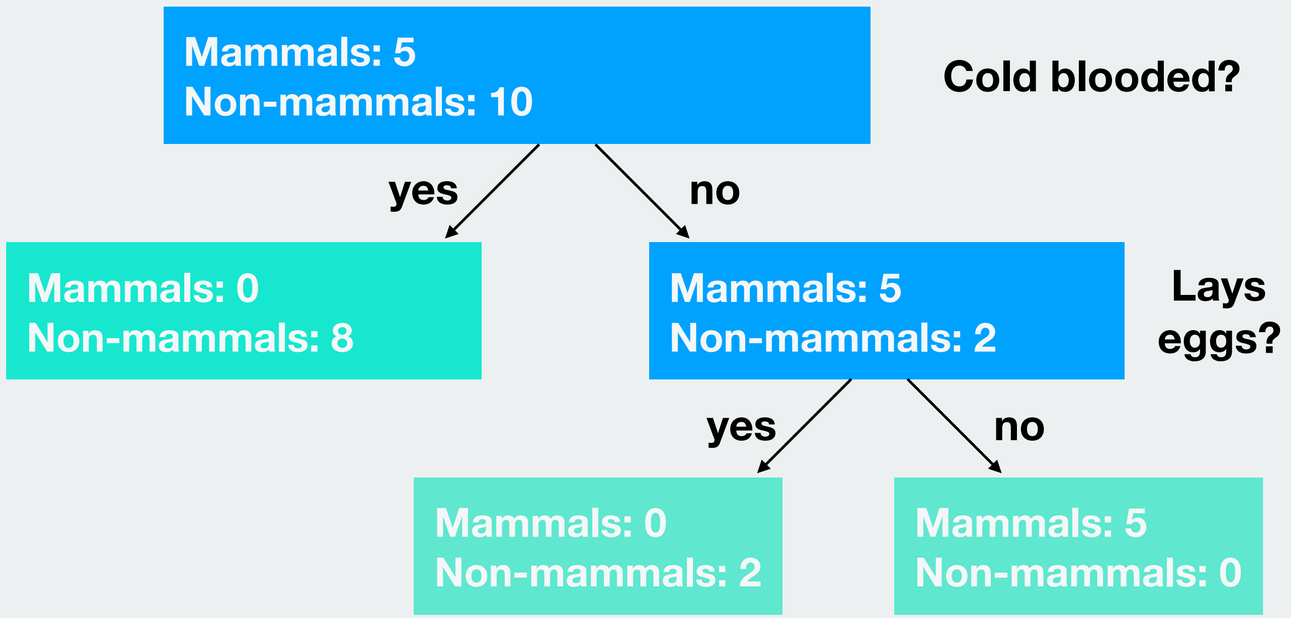
Could we have asked better questions?

Decision trees

	Lays eggs	Cold blooded	Mammal
no	0	0	1
no	0	0	1
no	0	0	1
no	0	0	1
no	0	0	1
yes	1	0	0
yes	1	0	0
yes	1	1	0
yes	1	1	0
yes	1	1	0
yes	1	1	0
yes	1	1	0
yes	1	1	0
yes	1	1	0
yes	1	1	0

features

target

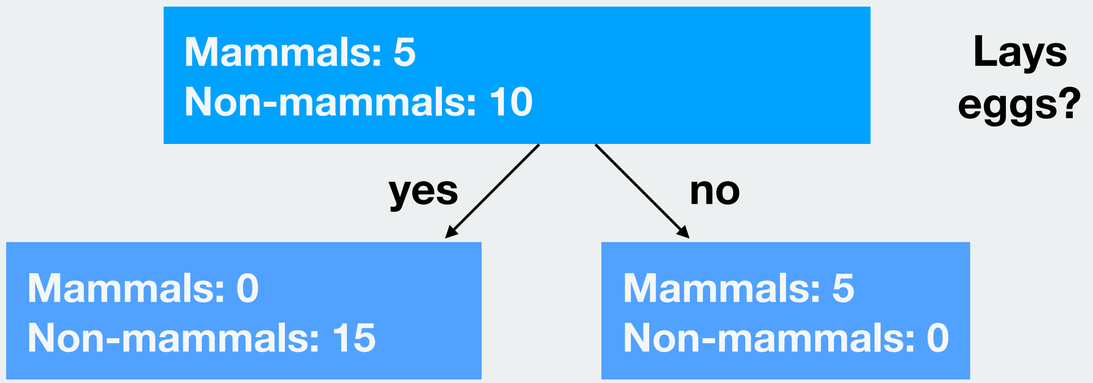


Decision trees

	Lays eggs	Cold blooded	Mammal
no	0	0	1
	0	0	1
	0	0	1
	0	0	1
	0	0	1
yes	1	0	0
	1	0	0
	1	1	0
	1	1	0
	1	1	0
	1	1	0
	1	1	0
	1	1	0
	1	1	0
	1	1	0

features

target

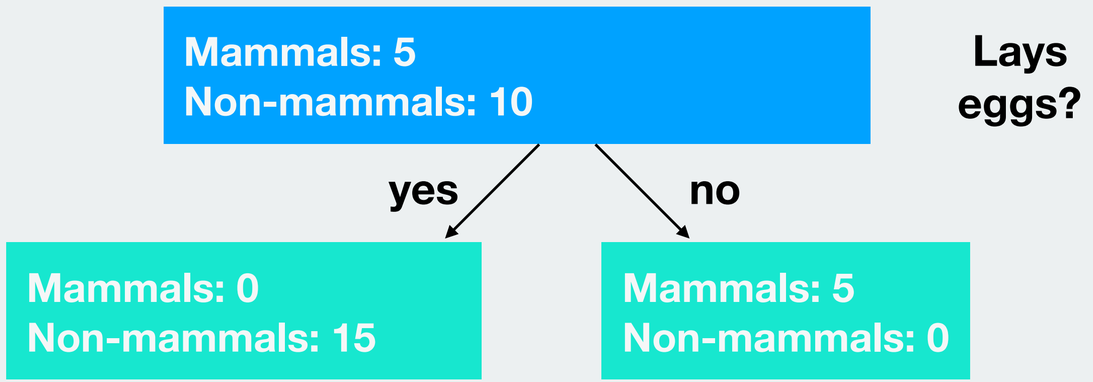


Decision trees

	Lays eggs	Cold blooded	Mammal
no	0	0	1
	0	0	1
	0	0	1
	0	0	1
	0	0	1
yes	1	0	0
	1	0	0
	1	1	0
	1	1	0
	1	1	0
	1	1	0
	1	1	0
	1	1	0
	1	1	0
	1	1	0

features

target

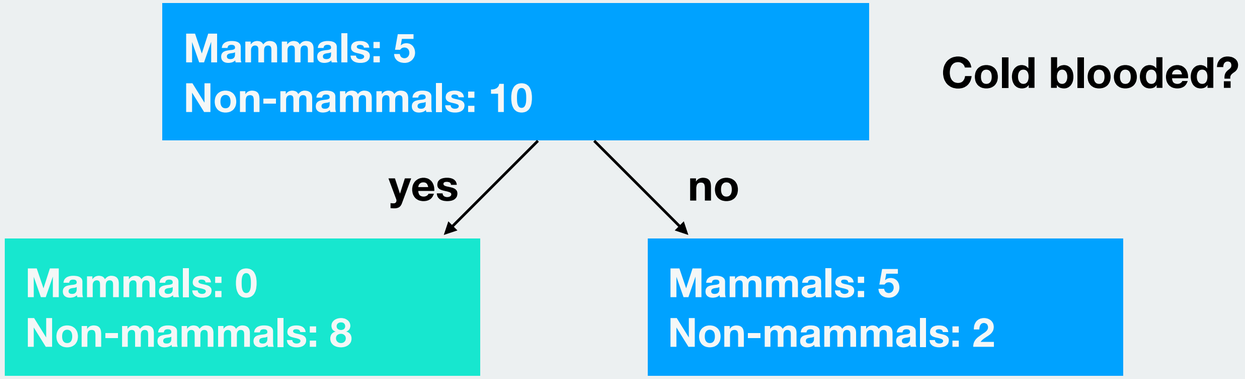


Decision trees

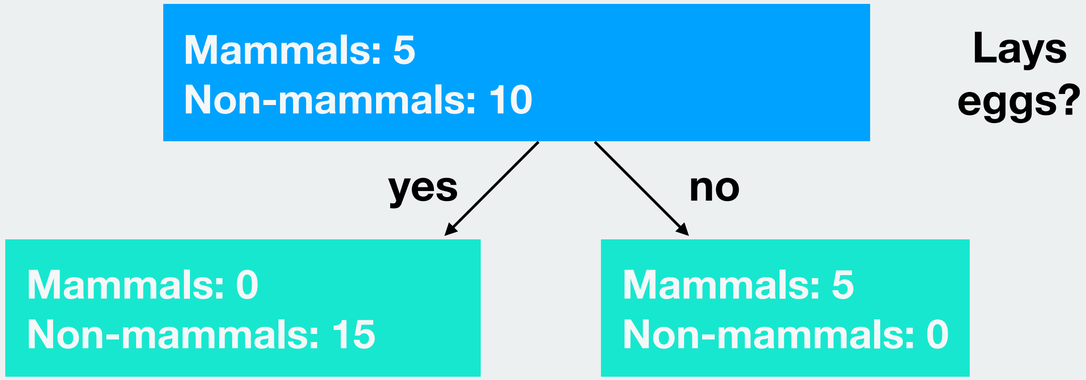
What do we do when we have lots of features and datapoints?

Automatic split selection

Split 1:



Split 2:



Automatic split selection

(Shannon) $$Entropy = -\sum_i p(i) \log_2 p(i)$$

Input: Probability vector (a list of values between 0 and 1, which sums to 1)

Output: Entropy (a measure of how "spread out" the probability distribution is)

Automatic split selection

$$Entropy = -\sum_i p(i) \log_2 p(i)$$

Mammals: 0
Non-mammals: 8

p = [1, 0]

Automatic split selection

$$Entropy = -\sum_i p(i) \log_2 p(i)$$

Mammals: 0
Non-mammals: 8

$p = [1, 0]$

$Entropy = -(1 \cdot \log_2(1) + 0 \cdot \log_2(0)) = 0$

Automatic split selection

$$Entropy = -\sum_i p(i) \log_2 p(i)$$

Mammals: 0
Non-mammals: 8

p = [1, 0]

Entropy = - (1 · $\log_2(1)$ + 0 · $\log_2(0)$) = 0

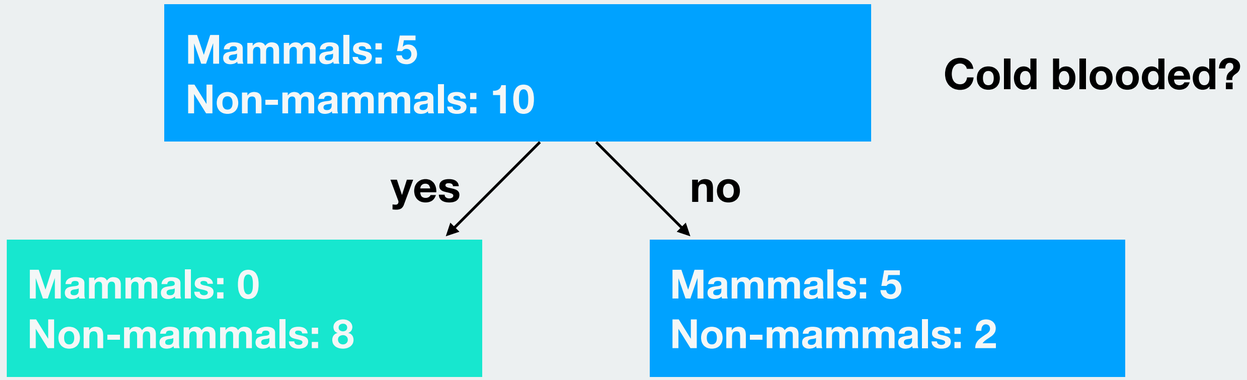
Mammals: 5
Non-mammals: 2

p = [2/7, 5/7]

Entropy = - (2/7 · $\log_2(2/7)$ + 5/7 · $\log_2(5/7)$) = 0.86

Automatic split selection

<u>Split 1:</u>



split entropy = 8 / 15 · 0 + 7 / 15 · 0.86 = 0.40

Automatic split selection

<u>Split 1:</u>

Cold blooded?

Mammals: 5
Non-mammals: 10

yes → **Mammals: 0**, **Non-mammals: 8**

no → **Mammals: 5**, **Non-mammals: 2**

split entropy = 8 / 15 · 0 + 7 / 15 · 0.86 = 0.40

<u>Split 2:</u>

Lays eggs?

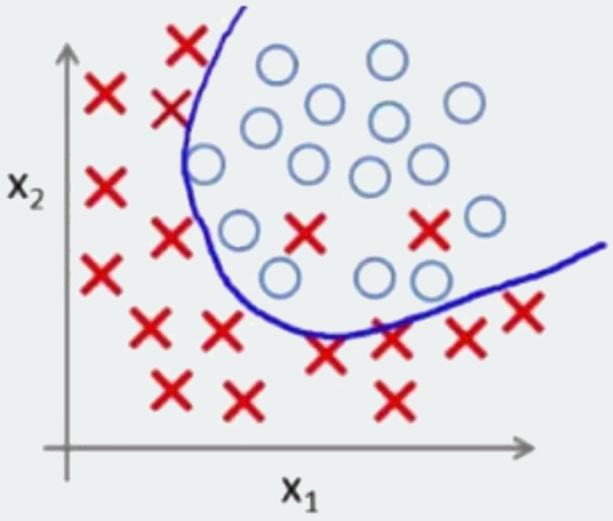
Mammals: 5
Non-mammals: 10

yes → **Mammals: 0**, **Non-mammals: 15**

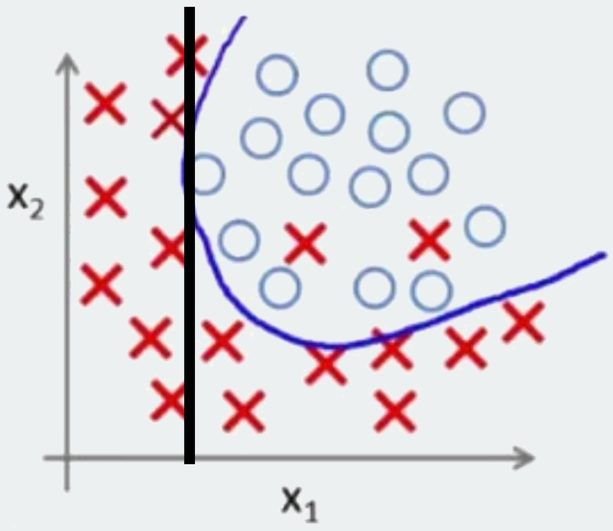
no → **Mammals: 5**, **Non-mammals: 0**

split entropy = 8 / 15 · 0 + 7 / 15 · 0 = 0

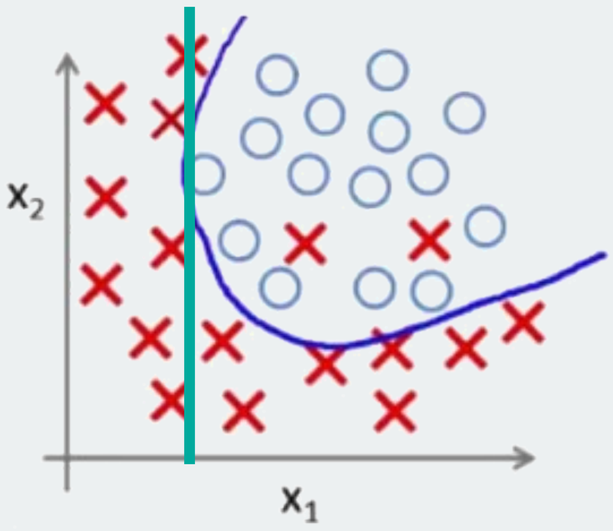
Automatic split selection



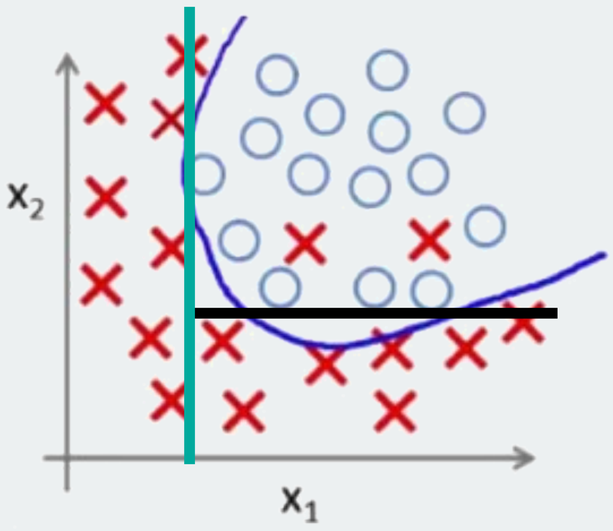
Automatic split selection



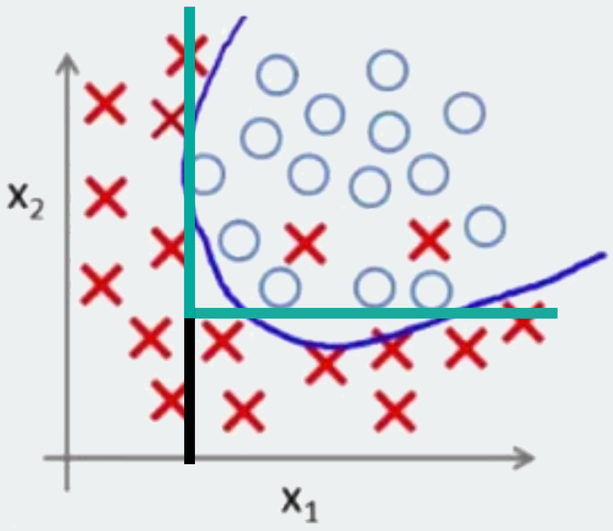
Automatic split selection



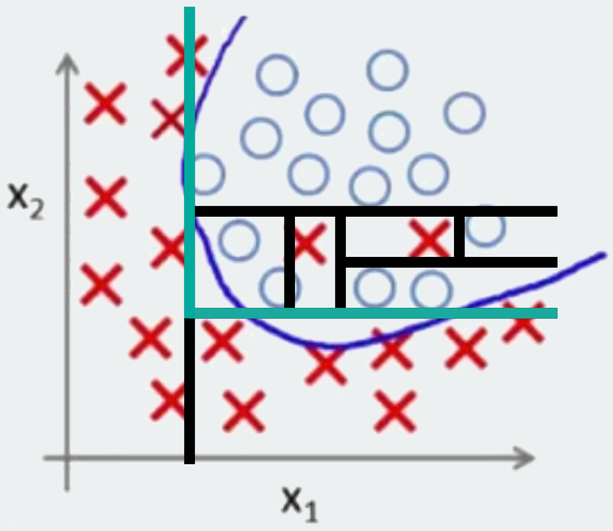
Automatic split selection



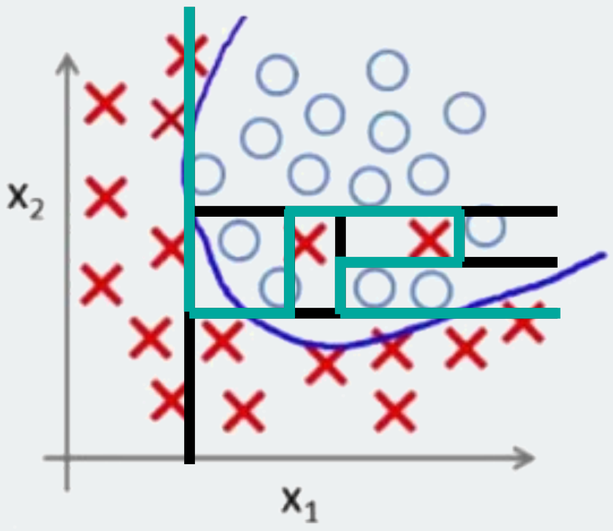
Automatic split selection



Automatic split selection



Automatic split selection



Ensemble learning

Ensemble Learning

- Create and train **many** classification **models**
- Treat each model as a "**voter**"
- Classify each data point according to what **most models** predicts it to be

model1(x) = 1
model2(x) = 1
model3(x) = 0
model4(x) = 1
model5(x) = 1
model6(x) = 1
model7(x) = 0
model8(x) = 1
...
modeln(x) = 1

average = 0.84 ≈ 1

Random Forest

model1

	Pclass1	Pclass2	Pclass3	Sexfemale	Sexmale	Embarkednan	EmbarkedC	EmbarkedQ	EmbarkedS	CabinFalse	CabinTrue	PassengerId	Age	SibSp	Parch	Fare	Survived
0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	1	22.0	1	0	7.2500	0
1	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	2	38.0	1	0	71.2833	1
2	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	3	26.0	0	0	7.9250	1
3	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	4	35.0	1	0	53.1000	1
4	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	5	35.0	0	0	8.0500	0
5	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	6	NaN	0	0	8.4583	0
6	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	7	54.0	0	0	51.8625	0
7	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	8	2.0	3	1	21.0750	0
8	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	9	27.0	0	2	11.1333	1
9	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	10	14.0	1	0	30.0708	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
881	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	882	33.0	0	0	7.8958	0
882	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	883	22.0	0	0	10.5167	0
883	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	884	28.0	0	0	10.5000	0
884	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	885	25.0	0	0	7.0500	0
885	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	886	39.0	0	5	29.1250	0
886	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	887	27.0	0	0	13.0000	0
887	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	888	19.0	0	0	30.0000	1
888	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	889	NaN	1	2	23.4500	0
889	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	890	26.0	0	0	30.0000	1
890	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	891	32.0	0	0	7.7500	0

Random Forest

model2

	Pclass1	Pclass2	Pclass3	Sexfemale	Sexmale	Embarkednan	EmbarkedC	EmbarkedQ	EmbarkedS	CabinFalse	CabinTrue	PassengerId	Age	SibSp	Parch	Fare	Survived
0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	1	22.0	1	0	7.2500	0
1	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	2	38.0	1	0	71.2833	1
2	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	3	26.0	0	0	7.9250	1
3	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	4	35.0	1	0	53.1000	1
4	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	5	35.0	0	0	8.0500	0
5	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	6	NaN	0	0	8.4583	0
6	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	7	54.0	0	0	51.8625	0
7	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	8	2.0	3	1	21.0750	0
8	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	9	27.0	0	2	11.1333	1
9	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	10	14.0	1	0	30.0708	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
881	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	882	33.0	0	0	7.8958	0
882	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	883	22.0	0	0	10.5167	0
883	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	884	28.0	0	0	10.5000	0
884	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	885	25.0	0	0	7.0500	0
885	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	886	39.0	0	5	29.1250	0
886	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	887	27.0	0	0	13.0000	0
887	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	888	19.0	0	0	30.0000	1
888	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	889	NaN	1	2	23.4500	0
889	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	890	26.0	0	0	30.0000	1
890	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	891	32.0	0	0	7.7500	0

Random Forest

model3

	Pclass1	Pclass2	Pclass3	Sexfemale	Sexmale	Embarkednan	EmbarkedC	EmbarkedQ	EmbarkedS	CabinFalse	CabinTrue	PassengerId	Age	SibSp	Parch	Fare	Survived
0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	1	22.0	1	0	7.2500	0
1	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	2	38.0	1	0	71.2833	1
2	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	3	26.0	0	0	7.9250	1
3	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	4	35.0	1	0	53.1000	1
4	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	5	35.0	0	0	8.0500	0
5	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	6	NaN	0	0	8.4583	0
6	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	7	54.0	0	0	51.8625	0
7	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	8	2.0	3	1	21.0750	0
8	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	9	27.0	0	2	11.1333	1
9	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	10	14.0	1	0	30.0708	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
881	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	882	33.0	0	0	7.8958	0
882	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	883	22.0	0	0	10.5167	0
883	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	884	28.0	0	0	10.5000	0
884	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	885	25.0	0	0	7.0500	0
885	0.0	0.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	886	39.0	0	5	29.1250	0
886	0.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	887	27.0	0	0	13.0000	0
887	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	888	19.0	0	0	30.0000	1
888	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	889	NaN	1	2	23.4500	0
889	1.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0	890	26.0	0	0	30.0000	1
890	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0	891	32.0	0	0	7.7500	0