Lecture 2: Data and Features

Machine Learning (BBWL)

Michael Mommert, University of St. Gallen

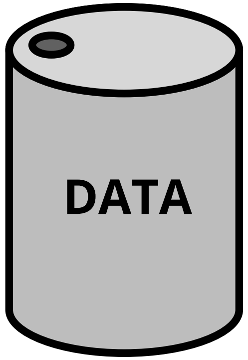
Today's lecture

Types of data

Features and feature engineering

Data scaling

Data



What is data?

Full Definition of *data*

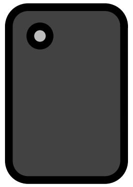
Merriam-Webster

What is data?

Full Definition of *data*

Merriam-Webster

: information output by a sensing device or organ that includes both useful and irrelevant or redundant information and must be processed to be meaningful



Data acquisition

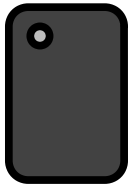
What is data?

Full Definition of *data*

Merriam-Webster

: information output by a sensing device or organ that includes both useful and irrelevant or redundant information and must be processed to be meaningful

: information in digital form that can be transmitted or processed



Data acquisition

What is data?

Merriam-Webster

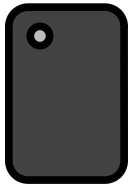
Full Definition of *data*

Merriam-Webster

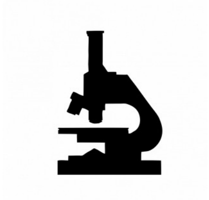
: information output by a sensing device or organ that includes both useful and irrelevant or redundant information and must be processed to be meaningful

: information in digital form that can be transmitted or processed

: factual information (such as measurements or statistics) used as a basis for reasoning, discussion, or calculation



Data acquisition



Data analysis

What is data?

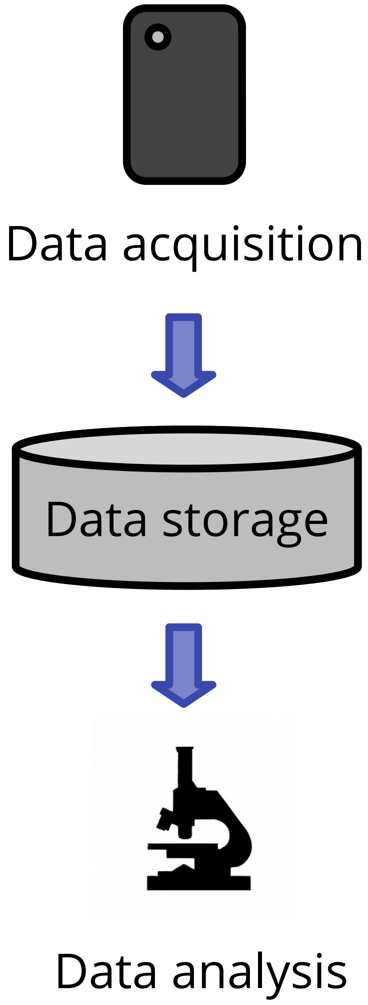
Full Definition of *data*

Merriam-Webster

: information output by a sensing device or organ that includes both useful and irrelevant or redundant information and must be processed to be meaningful

: information in digital form that can be transmitted or processed

: factual information (such as measurements or statistics) used as a basis for reasoning, discussion, or calculation



What is data?

Merriam-Webster

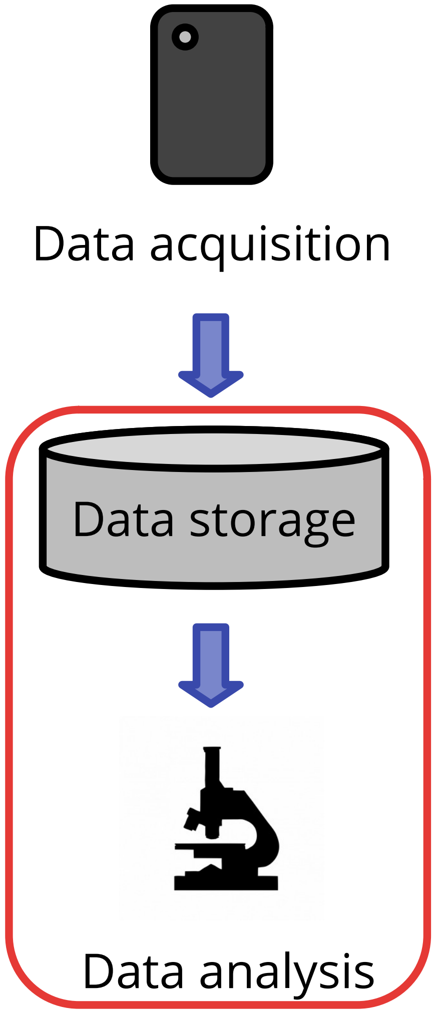
Full Definition of *data*

Merriam-Webster

: information output by a sensing device or organ that includes both useful and irrelevant or redundant information and must be processed to be meaningful

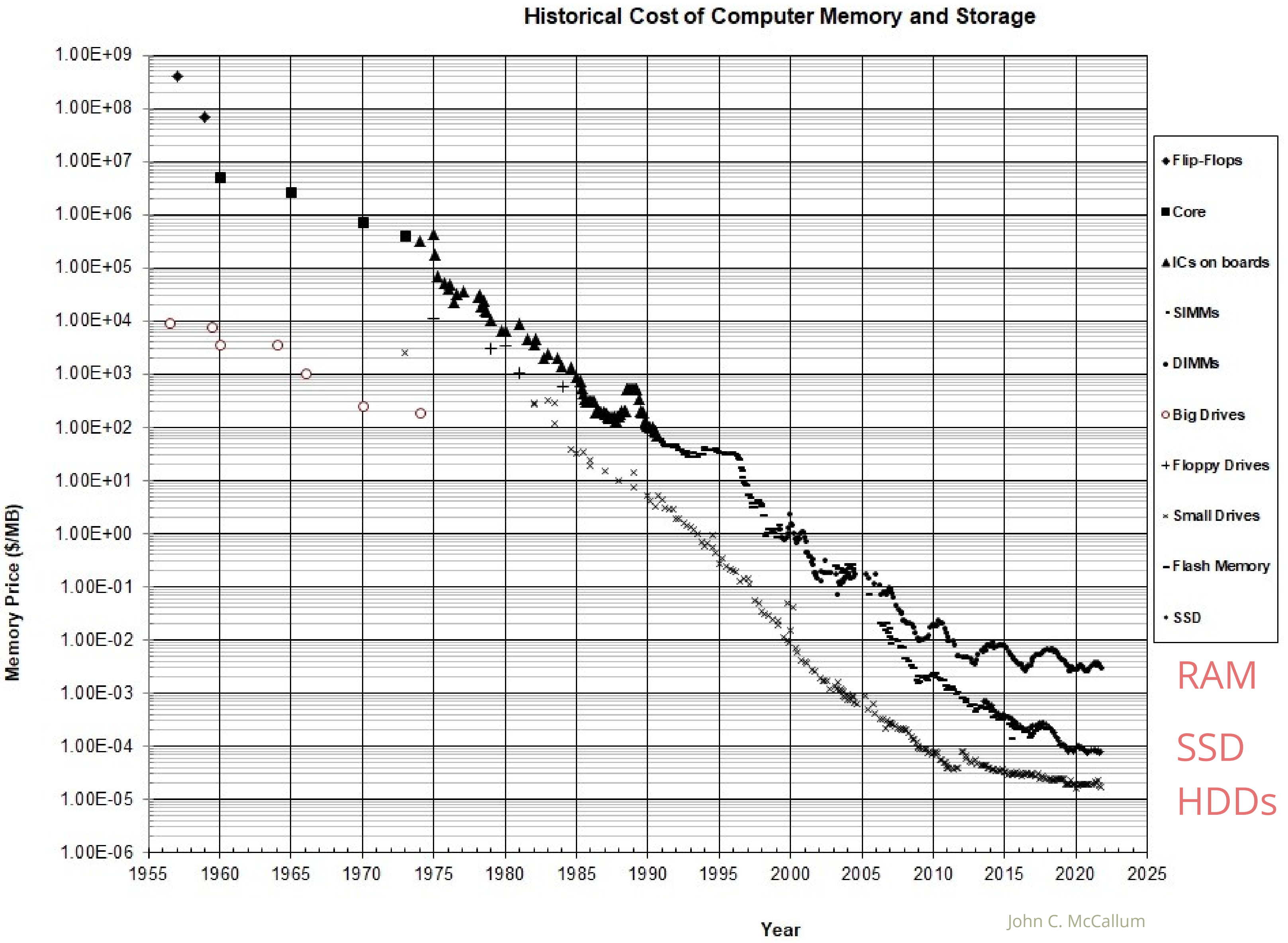
: information in digital form that can be transmitted or processed

: factual information (such as measurements or statistics) used as a basis for reasoning, discussion, or calculation



Data storage

- Data storage used to be a bottleneck – not anymore!



John C. McCallum

Data storage

- Data storage used to be a bottleneck – not anymore!

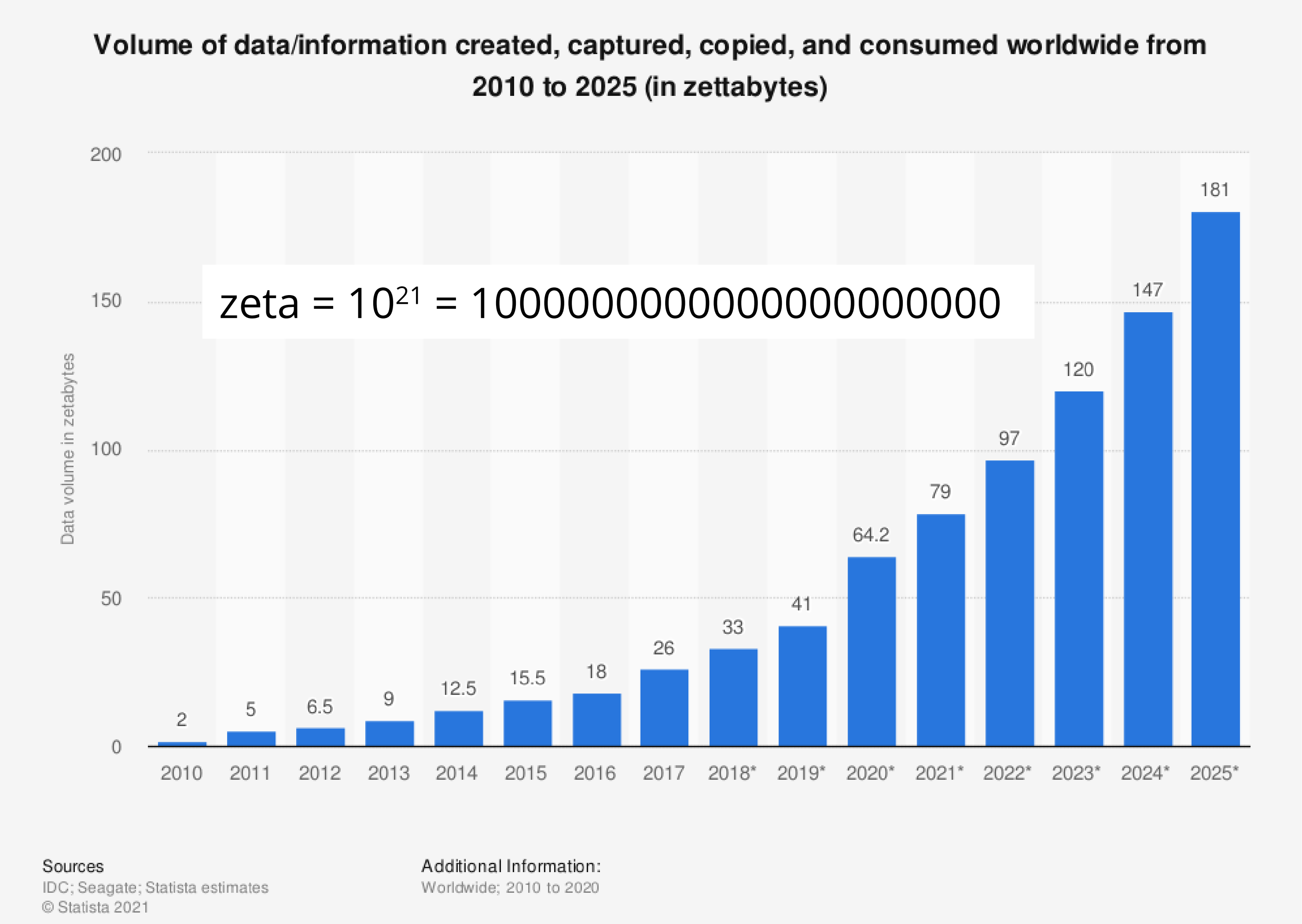
Data storage

- Data storage used to be a bottleneck – not anymore!

- Vast amounts of data can now be stored easily

Data storage

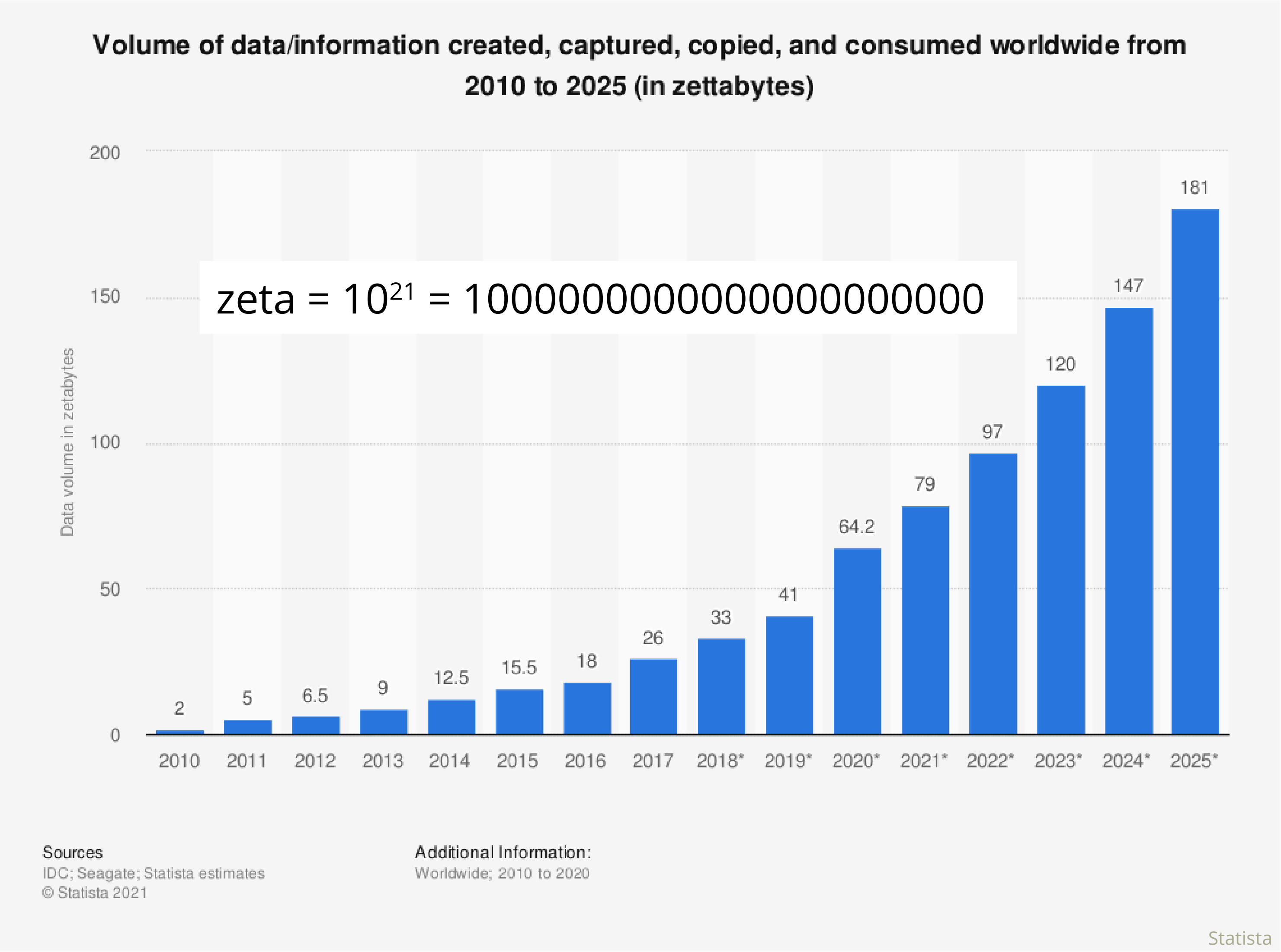
- Data storage used to be a bottleneck – not anymore!
- Vast amounts of data can now be stored easily



Statista

Data storage

- Data storage used to be a bottleneck – not anymore!
- Vast amounts of data can now be stored easily
- Is all this data technically accessible for analysis?
  (of course not, since most of it is privately owned, but…)



Structured vs unstructured data

Structured vs unstructured data

Structured data

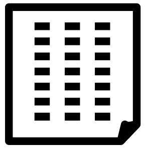
Preprocessed and formatted data that is easily queryable.

Structured vs unstructured data

Structured data

Preprocessed and formatted data that is easily queryable.

Quantitative data

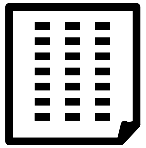


Structured vs unstructured data

Structured data

Preprocessed and formatted data that is easily queryable.

Quantitative data



Unstructured data

Unprocessed and unformatted data is not easily queryable.

Structured vs unstructured data

Structured data

Preprocessed and formatted data that is easily queryable.

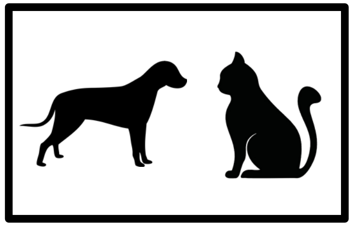
Quantitative data

Unstructured data

Unprocessed and unformatted data is not easily queryable.

Qualitative data

Image data



Video data

Textual data

Data stream

Audio data

Structured vs unstructured data

Structured data

Preprocessed and formatted data that is easily queryable.

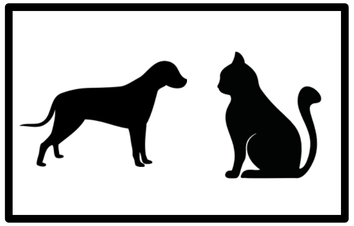
Quantitative data

Unstructured data

Unprocessed and unformatted data is not easily queryable.

Qualitative data

Image data



Video data

Data complexity

Textual data

Data stream

Audio data

Structured vs unstructured data

Structured data

Preprocessed and formatted data that is easily queryable.

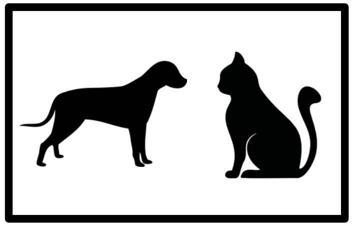
Quantitative data

Unstructured data

Unprocessed and unformatted data is not easily queryable.

Qualitative data

Image data



Video data

Data complexity

Textual data

Data stream

Audio data

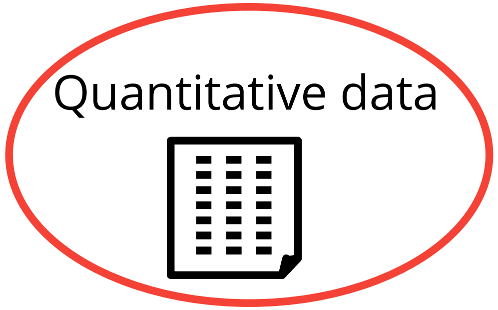
Most data analysis techniques require data to be available in a structured form for easier processing.

Structured data can always be represented in a database **schema** (e.g., a table in 2 dimensions).

Structured vs unstructured data

Structured data

Preprocessed and formatted data that is easily queryable.

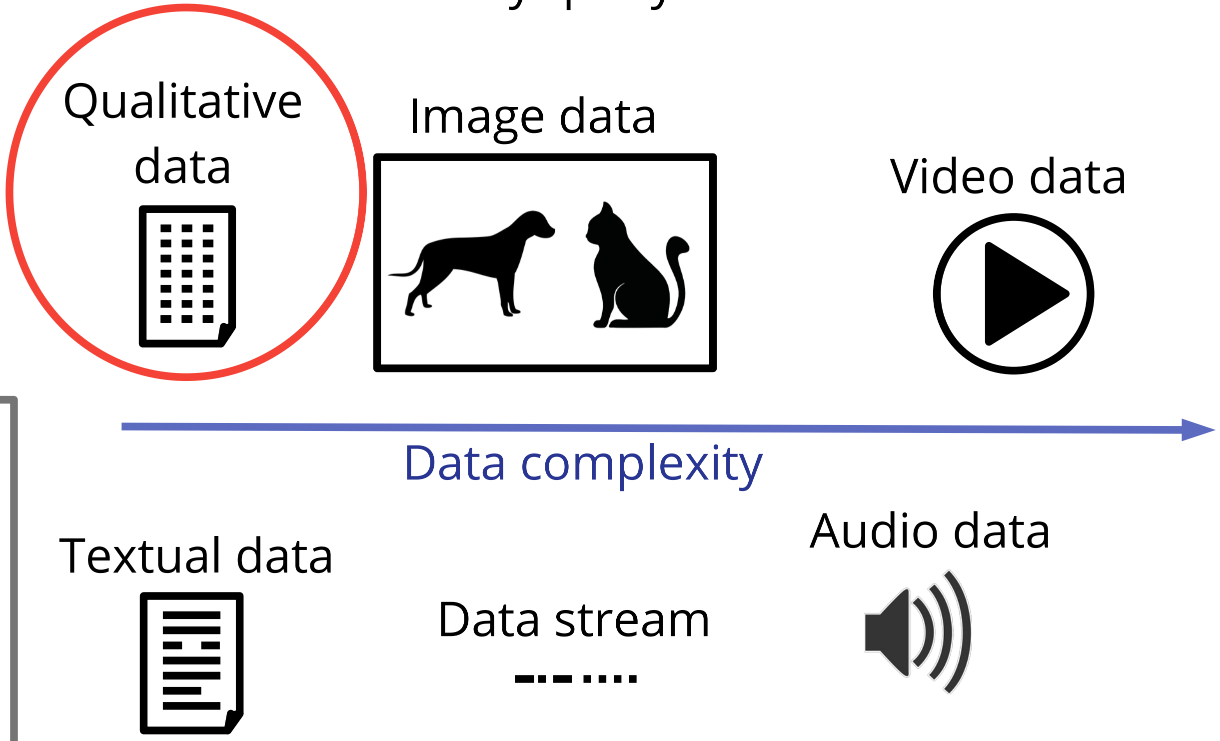


Most data analysis techniques require data to be available in a structured form for easier processing.

Structured data can always be represented in a database **schema** (e.g., a table in 2 dimensions).

Unstructured data

Unprocessed and unformatted data is not easily queryable.



Quantitative and qualitative data

Quantitative data (can be measured; distances can be defined)	**Qualitative (categorical) data** (cannot be measured; distances not defined)

Quantitative and qualitative data

Quantitative data (can be measured; distances can be defined)	**Qualitative (categorical) data** (cannot be measured; distances not defined)

Continuous data

Real-valued numbers; potentially within a given range

Examples:
- Temperatures
- A person’s height
- Prices

Quantitative and qualitative data

Quantitative data
(can be measured; distances can be defined)

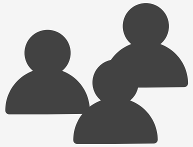
Continuous data

Real-valued numbers; potentially within a given range

Examples:
- Temperatures
- A person's height
- Prices

Discrete data

Discrete numbers; whole numbers or real numbers, potentially within a given range



Examples:
- Number of people in a room
- Inventory counts

Qualitative (categorical) data
(cannot be measured; distances not defined)

Quantitative and qualitative data

Quantitative data (can be measured; distances can be defined)		**Qualitative (categorical) data** (cannot be measured; distances not defined)
Continuous data	**Discrete data**	**Nominal data**

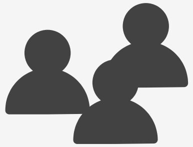
Continuous data

Real-valued numbers; potentially within a given range

Examples:
- Temperatures
- A person's height
- Prices

Discrete data

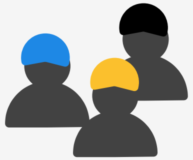
Discrete numbers; whole numbers or real numbers, potentially within a given range



Examples:
- Number of people in a room
- Inventory counts

Nominal data

Labels for different categories without ordering



Examples:
- Color of hair
- Names of persons
- Types of fruit

Quantitative and qualitative data

Quantitative data
(can be measured; distances can be defined)

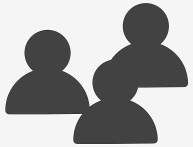
Continuous data

Real-valued numbers; potentially within a given range

Examples:
- Temperatures
- A person's height
- Prices

Discrete data

Discrete numbers; whole numbers or real numbers, potentially within a given range

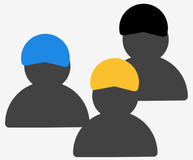


Examples:
- Number of people in a room
- Inventory counts

Qualitative (categorical) data
(cannot be measured; distances not defined)

Nominal data

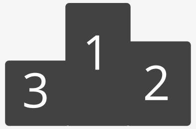
Labels for different categories without ordering



Examples:
- Color of hair
- Names of persons
- Types of fruit

Ordinal data

Labels for different categories following an inherent ranking scheme.



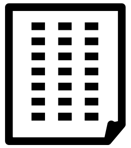
Examples:
- Rank in a competition
- Grades
- Day of the week

Turning unstructured data into structured data

Structured data

Preprocessed and formatted data that is easily queryable.

Quantitative



Unstructured data

Unprocessed and unformatted data is not easily queryable.

Qualitative data

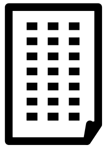
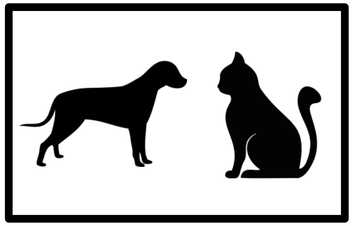
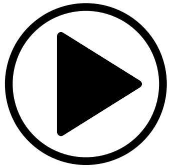


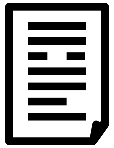
Image data



Video data



Textual data



Data stream

Audio data

Turning unstructured data into structured data

Structured data

Preprocessed and formatted data that is easily queryable.

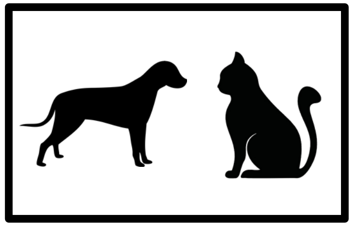
Quantitative

Unstructured data

Unprocessed and unformatted data is not easily queryable.

Qualitative data

Image data



Video data

Textual data

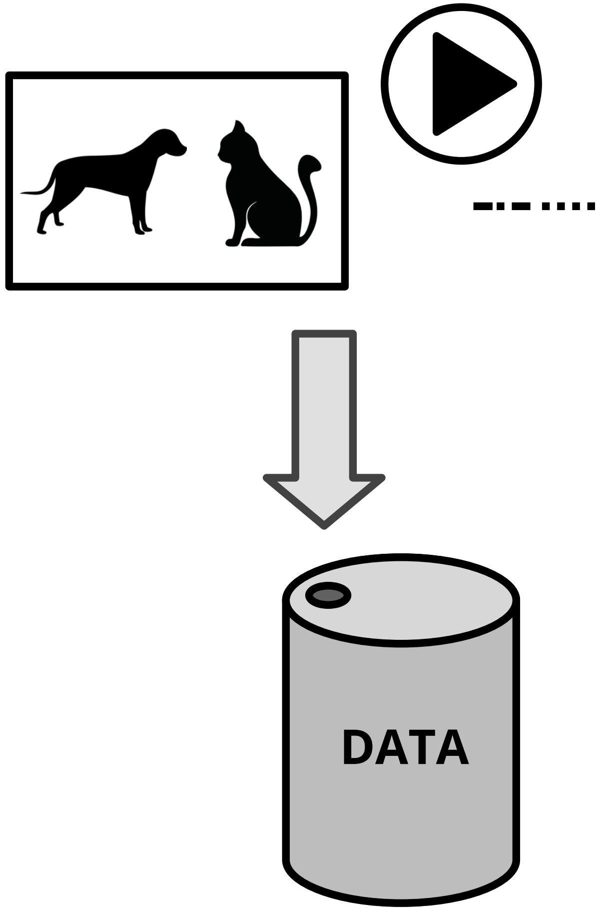
Data stream

Audio data

Before ML methods can be applied to unstructured data, we have to process those and extract useful features from them.

This process is called **feature engineering**.

Features and Feature Engineering

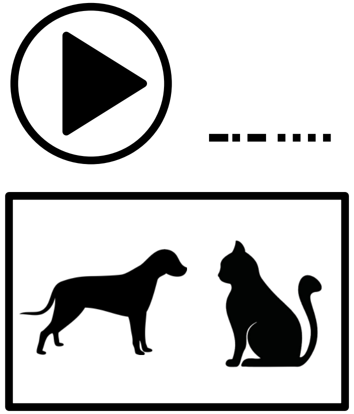


What are features?

Features are quantitative and independent variables based on which our ML models learn.

What are features?

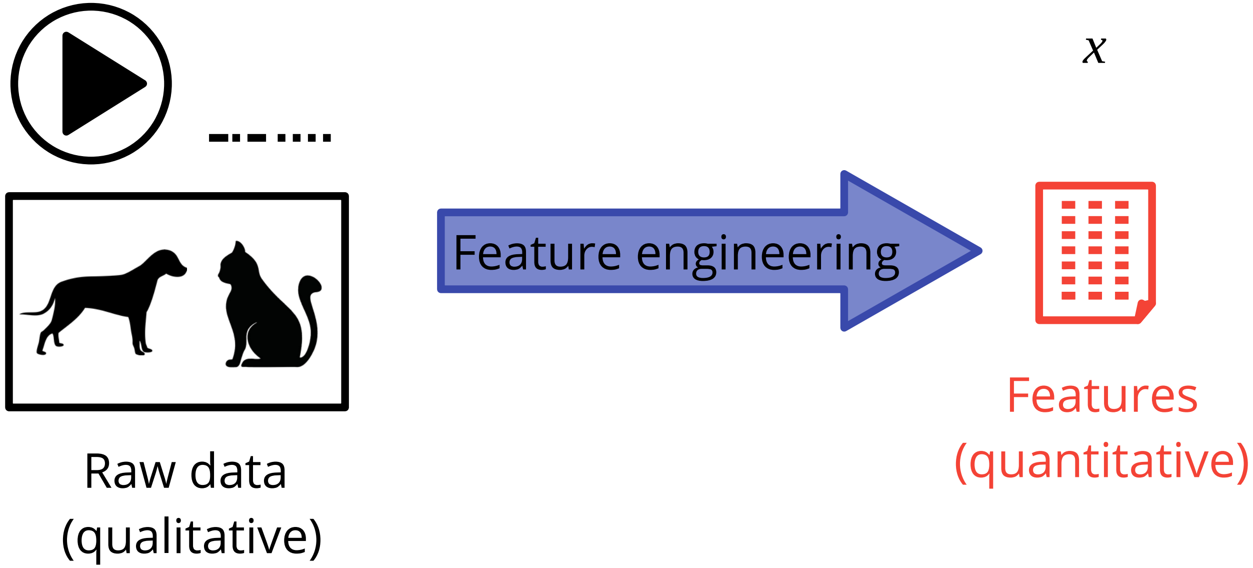
Features are quantitative and independent variables based on which our ML models learn.



Raw data
(qualitative)

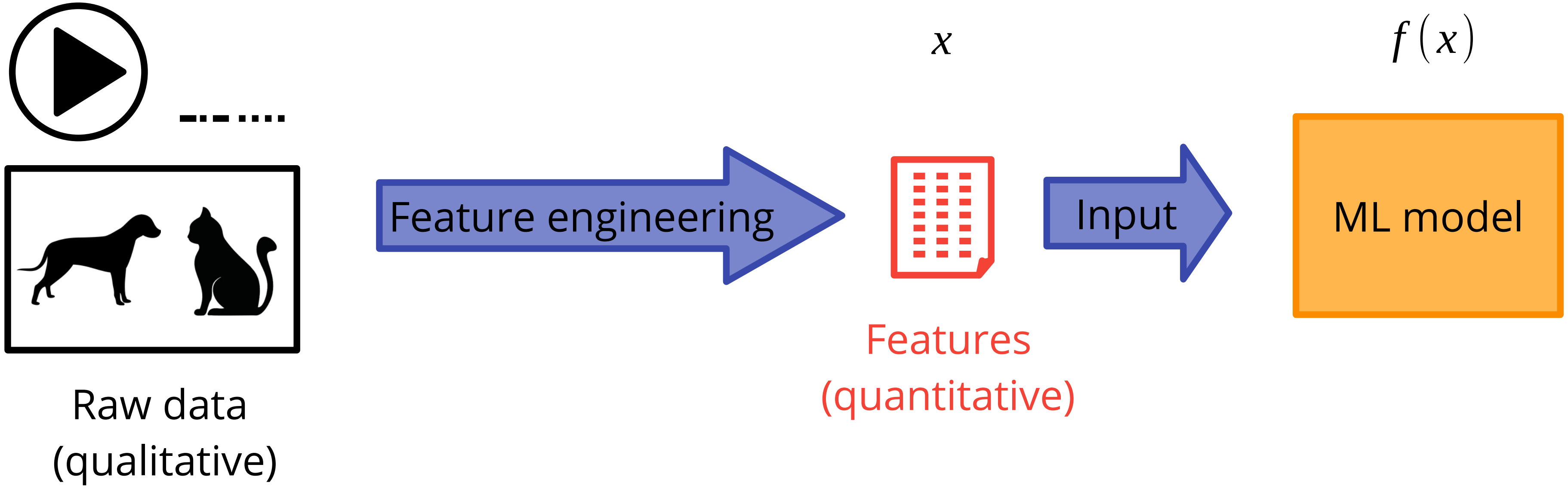
What are features?

Features are quantitative and independent variables based on which our ML models learn.



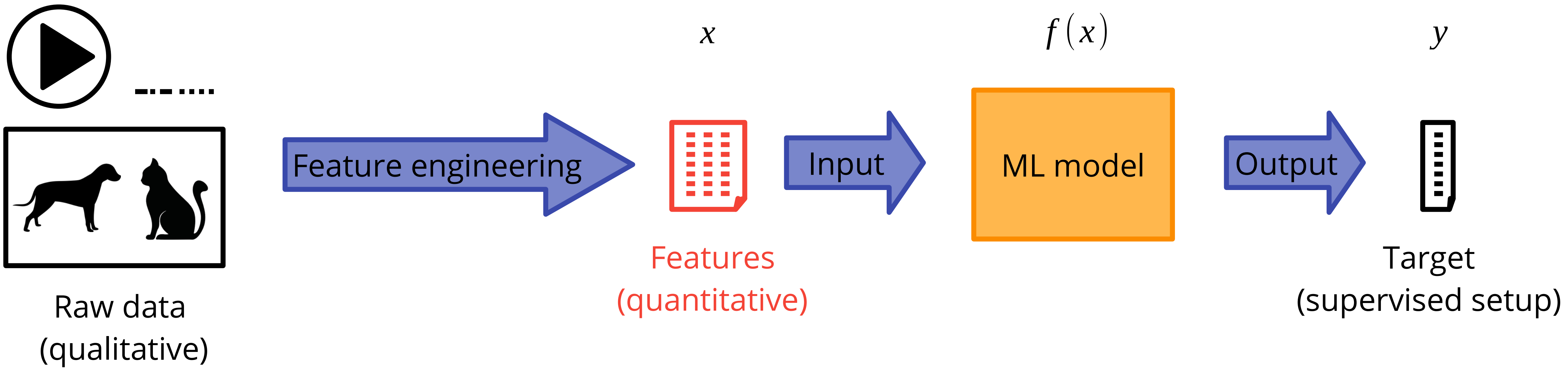
What are features?

Features are quantitative and independent variables based on which our ML models learn.



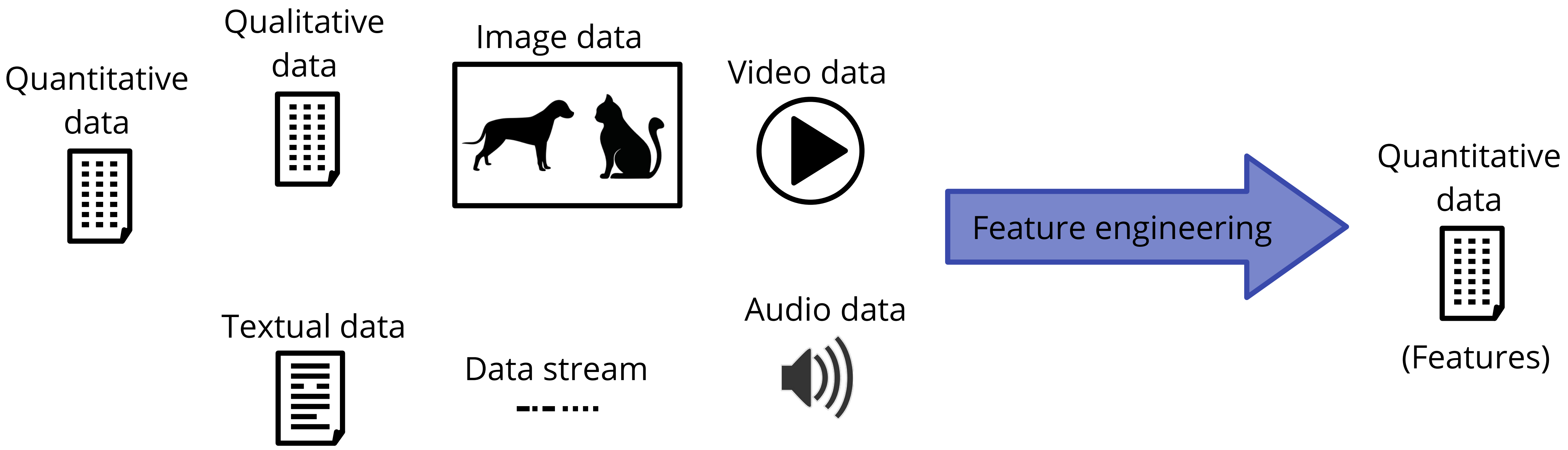
What are features?

Features are quantitative and independent variables based on which our ML models learn.



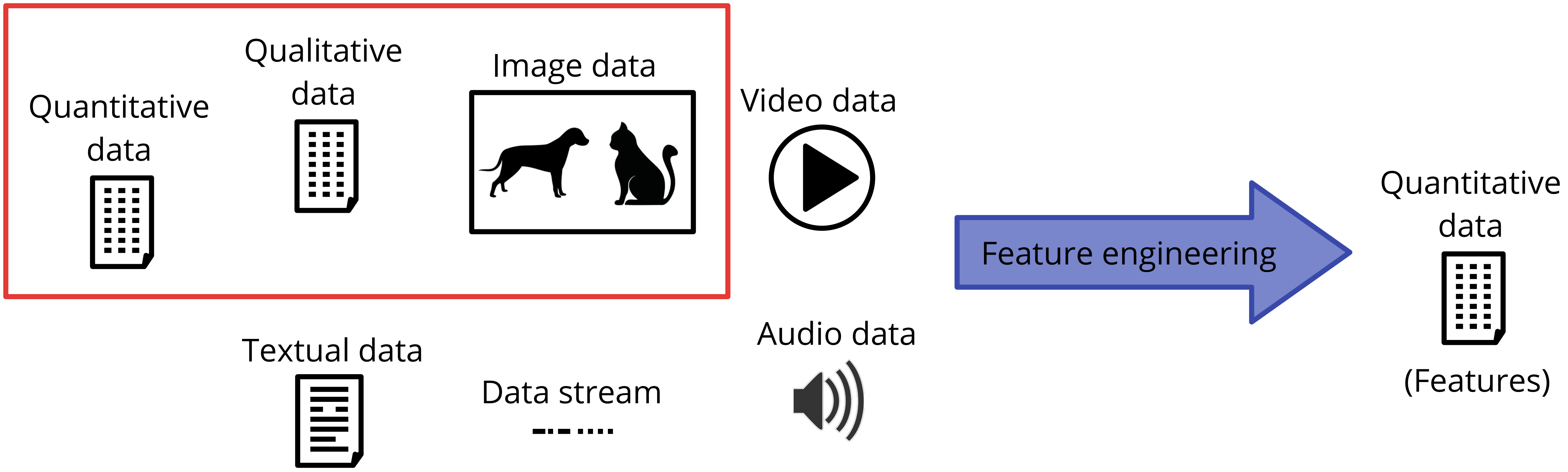
Feature engineering

Extract or create features that may provide a ML model with rich information on its task based on **domain knowledge**. Feature engineering can be applied to raw data, resulting in quantitative data that can be directly fed into the ML model (features).



Feature engineering

Extract or create features that may provide a ML model with rich information on its task based on **domain knowledge**. Feature engineering can be applied to raw data, resulting in quantitative data that can be directly fed into the ML model (features).



Feature engineering – quantitative data

Create meaningful features through mathematical transformations.

Examples:

Feature engineering – quantitative data

Create meaningful features through mathematical transformations.

Examples:

Arithmetic

Situation: You have two variables, x_1 and x_2 , but you are more interested in their difference, δ .

Transformation:

$$\delta = x_1 - x_2$$

Feature engineering – quantitative data

Create meaningful features through mathematical transformations.

Examples:

Arithmetic	**Aggregation of Features**
Situation: You have two variables, x_1 and x_2, but you are more interested in their difference, δ.	*Situation*: You have results from different business units, $\boldsymbol{x_i}$, but your ML model should not consider the results separately, but as an aggregated overall result, $\boldsymbol{x}$.
Transformation: $\delta = x_1 - x_2$	*Transformation*: $\boldsymbol{x} = \sum_i \boldsymbol{x_i}$

Feature engineering – quantitative data

Create meaningful features through mathematical transformations.

Examples:

Arithmetic

Situation: You have two variables, x_1 and x_2, but you are more interested in their difference, δ.

Transformation:

$$\delta = x_1 - x_2$$

Aggregation of Features

Situation: You have results from different business units, $\boldsymbol{x_i}$, but your ML model should not consider the results separately, but as an aggregated overall result, $\boldsymbol{x}$.

Transformation:

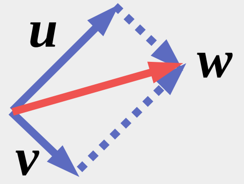
$$\boldsymbol{x} = \sum_i \boldsymbol{x_i}$$

Geometric Transformations

Situation: To identify common wind speed patterns, you have measurements of two orthogonal wind speed components, $\boldsymbol{u}$ and $\boldsymbol{v}$. Since only the magnitude of the resulting wind vector, $\boldsymbol{w}$, matters, you can utilize its magnitude, $|\boldsymbol{w}|$.

Transformation:

$$|\boldsymbol{w}| = \sqrt{\boldsymbol{u}^2 + \boldsymbol{v}^2}$$



Feature engineering – qualitative data

Feature engineering – qualitative data

Qualitative (categorical) data cannot be fed into ML models directly, they have to be turned into quantitative data first. There are two common methods available, depending on the data type:

Feature engineering – qualitative data

Qualitative (categorical) data cannot be fed into ML models directly, they have to be turned into quantitative data first. There are two common methods available, depending on the data type:

- **Label encoding**: *ordinal (ranked) data → discrete quantitative data*
  The intuition is that the ranking/order of the classes is conserved in a discrete numerical schema and a "distance" can be defined.

  Examples:

  - Competition ranks: [1st, 2nd, 3rd, 4th, 5th] → [1, 2, 3, 4, 5]
  - Cloudiness scale: [clear, mostly clear, partly cloudy, mostly cloudy] → [0, 1, 2, 3]
  - Quality scale: [very good, good, satisfying, sufficient, insufficient] → [0, 1, 2, 3, 4]
  - Days of the week: [Mon, Tue, Wed, Thu, Fri, Sat, Sun] → [1, 2, 3, 4, 5, 6, 7]

Feature engineering – qualitative data

Qualitative (categorical) data cannot be fed into ML models directly, they have to be turned into quantitative data first. There are two common methods available, depending on the data type:

- **Label encoding**: *ordinal (ranked) data → discrete quantitative data*
  The intuition is that the ranking/order of the classes is conserved in a discrete numerical schema and a “distance” can be defined.

  Examples:

  - Competition ranks: [1st, 2nd, 3rd, 4th, 5th] → [1, 2, 3, 4, 5]
  - Cloudiness scale: [clear, mostly clear, partly cloudy, mostly cloudy] → [0, 1, 2, 3]
  - Quality scale: [very good, good, satisfying, sufficient, insufficient] → [0, 1, 2, 3, 4]
  - Days of the week: [Mon, Tue, Wed, Thu, Fri, Sat, Sun] → [1, 2, 3, 4, 5, 6, 7]

← be careful: day of week is cyclical!

Feature engineering – qualitative data

Qualitative (categorical) data cannot be fed into ML models directly, they have to be turned into quantitative data first. There are two common methods available, depending on the data type:

- **Label encoding**: *ordinal (ranked) data → discrete quantitative data*
  The intuition is that the ranking/order of the classes is conserved in a discrete numerical schema and a "distance" can be defined.

  Examples:

  - Competition ranks: [1st, 2nd, 3rd, 4th, 5th] → [1, 2, 3, 4, 5]
  - Cloudiness scale: [clear, mostly clear, partly cloudy, mostly cloudy] → [0, 1, 2, 3]
  - Quality scale: [very good, good, satisfying, sufficient, insufficient] → [0, 1, 2, 3, 4]
  - Days of the week: [Mon, Tue, Wed, Thu, Fri, Sat, Sun] → [1, 2, 3, 4, 5, 6, 7]

← be careful: day of week is cyclical!

(Caveat: Label encoding can also be used if a large number of classes is present)

Feature engineering – qualitative data

Qualitative (categorical) data cannot be fed into ML models directly, they have to be turned into quantitative data first. There are two common methods available, depending on the data type:

- **Label encoding**: *ordinal (ranked) data → discrete quantitative data*

Feature engineering – qualitative data

Qualitative (categorical) data cannot be fed into ML models directly, they have to be turned into quantitative data first. There are two common methods available, depending on the data type:

- **Label encoding**: *ordinal (ranked) data → discrete quantitative data*
- **One-hot encoding**: *nominal (unranked) data → binary coding of labels*
  For each possible class in a feature, a binary feature is introduced; for each sample, all one-hot features are zero, only those that match have a value of one.
  Examples:
  - House properties: [balcony, cellar, fireplace, jacuzzi] →
    samples: house 1: “balcony” →
    house 2: ”fireplace” →
    house 3: ”balcony and jacuzzi” →
    house 4: “cellar, fireplace and jacuzzi” →

Feature engineering – qualitative data

Qualitative (categorical) data cannot be fed into ML models directly, they have to be turned into quantitative data first. There are two common methods available, depending on the data type:

- **Label encoding**: *ordinal (ranked) data → discrete quantitative data*
- **One-hot encoding**: *nominal (unranked) data → binary coding of labels*
  For each possible class in a feature, a binary feature is introduced; for each sample, all one-hot features are zero, only those that match have a value of one.
  Examples:
  - House properties: [balcony, cellar, fireplace, jacuzzi] →
    samples: house 1: "balcony" →
    house 2: "fireplace" →
    house 3: "balcony and jacuzzi" →
    house 4: "cellar, fireplace and jacuzzi" →

balcony	cellar	fireplace	jacuzzi
1	0	0	0
0	0	1	0
1	0	0	1
0	1	1	1

Feature engineering – qualitative data

Qualitative (categorical) data cannot be fed into ML models directly, they have to be turned into quantitative data first. There are two common methods available, depending on the data type:

- **Label encoding**: *ordinal (ranked) data → discrete quantitative data*
- **One-hot encoding**: *nominal (unranked) data → binary coding of labels*
  For each possible class in a feature, a binary feature is introduced; for each sample, all one-hot features are zero, only those that match have a value of one.
  Examples:
  - House properties: [balcony, cellar, fireplace, jacuzzi] →
    samples: house 1: "balcony" →
    house 2: "fireplace" →
    Multi-class feature { house 3: "balcony and jacuzzi" →
    house 4: "cellar, fireplace and jacuzzi" →

balcony	cellar	fireplace	jacuzzi
1	0	0	0
0	0	1	0
1	0	0	1
0	1	1	1

Feature engineering – qualitative data

Qualitative (categorical) data cannot be fed into ML models directly, they have to be turned into quantitative data first. There are two common methods available, depending on the data type:

- **Label encoding**: *ordinal (ranked) data → discrete quantitative data*
- **One-hot encoding**: *nominal (unranked) data → binary coding of labels*
  For each possible class in a feature, a binary feature is introduced; for each sample, all one-hot features are zero, only those that match have a value of one.
  Examples:
  - House properties: [balcony, cellar, fireplace, jacuzzi] →
    samples: house 1: “balcony” →
    house 2: ”fireplace” →
    Multi-class feature { house 3: ”balcony and jacuzzi” →
    house 4: “cellar, fireplace and jacuzzi” →

balcony	cellar	fireplace	jacuzzi
1	0	0	0
0	0	1	0
1	0	0	1
0	1	1	1

(Caveat: if too many classes present, use label encoding instead; see *curse of dimensionality*)

Feature engineering – image data

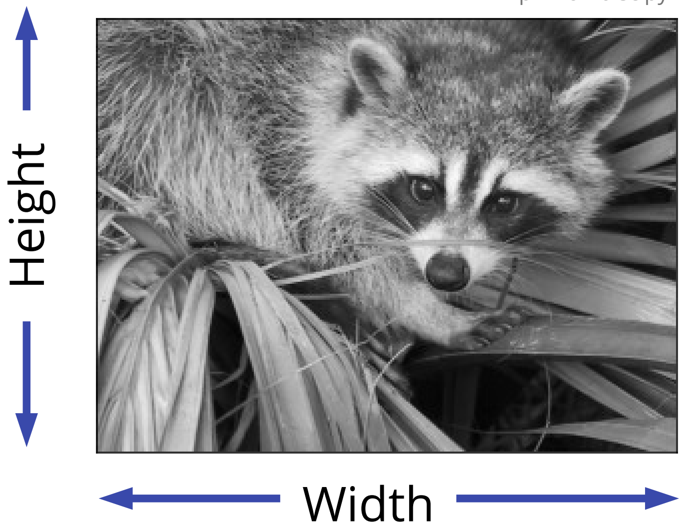
How are images represented?

pixnio via scipy

Feature engineering – image data

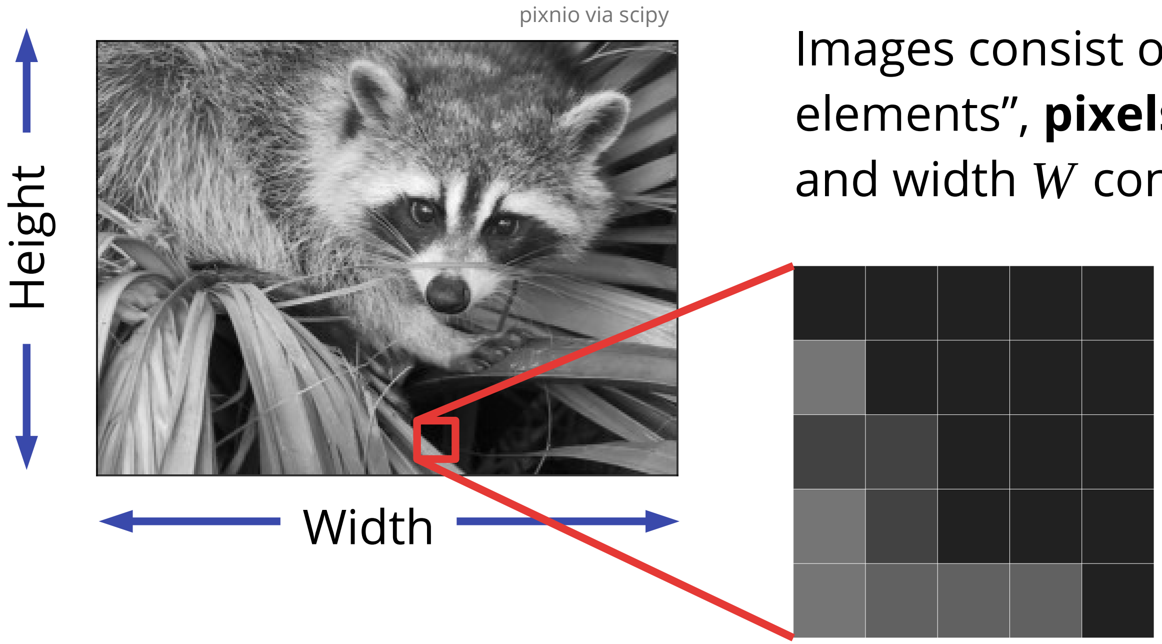
How are images represented?

pixnio via scipy



Feature engineering – image data

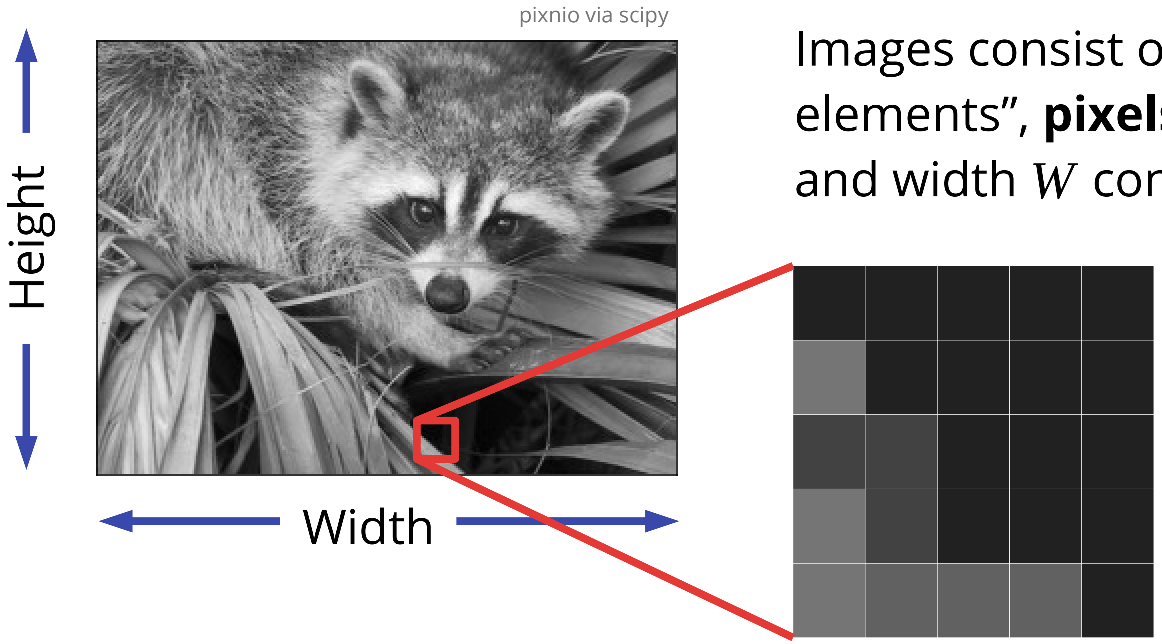
How are images represented?



Images consist of quadratic "Picture elements", **pixels**. An image of height H and width W contains $H \times W$ pixels.

Feature engineering – image data

How are images represented?



Images consist of quadratic “Picture elements”, **pixels**. An image of height H and width W contains $H \times W$ pixels.

In a **greyscale** image, the brightness of each pixel is represented by a single value [0, 1] (8-bit encoding: [0, 256]) where 0 refers to a black pixel and the maximum value to a white pixel.

0	0	0	0	0
0.3	0	0	0	0
0.1	0.1	0	0	0
0.3	0.1	0	0	0
0.3	0.2	0.2	0.2	0

Feature engineering – image data

How are images represented?

Greyscale

pixnio via scipy

```
[7] image.shape
```

“2-d” (600, 400)

Feature engineering – image data

How are images represented?

Greyscale

pixnio via scipy

```
[7] image.shape
```

"2-d" (600, 400)

```
[5] image.shape
```

"3-d" (600, 400, 3)

Color (RGB)

=

Color images consist of **three channels** (typically Red, Green and Blue; RGB), each of which is a grayscale image in itself. When displayed, the channels are combined to form a color image.

Feature engineering – image data

How are images represented?

Greyscale

pixnio via scipy

```
[7] image.shape
```

"2-d" (600, 400)

```
[5] image.shape
```

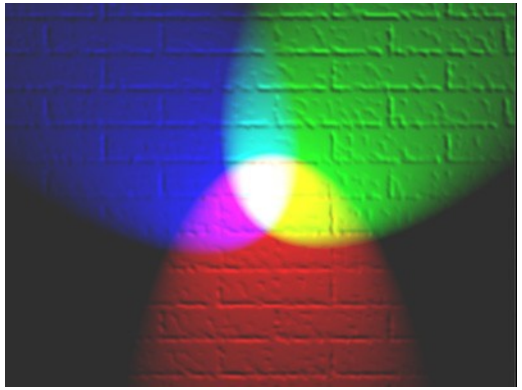
"3-d" (600, 400, 3)

Color (RGB)

=

Why RGB?

Every color can be synthesized through additive mixing of red, green and blue.



Bb3cxv @ wikipedia

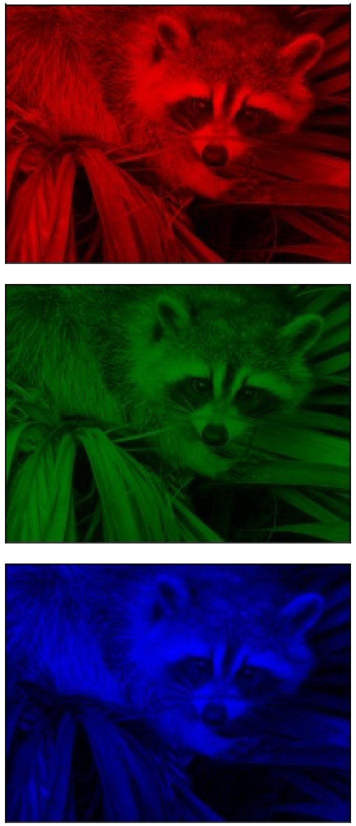
Color images consist of **three channels** (typically Red, Green and Blue; RGB), each of which is a grayscale image in itself. When displayed, the channels are combined to form a color image.

Feature engineering – image data

How can we feed image data into ML models?

- **Whole images**

pixnio via scipy



Split channels

Features

Concatenate all channels and feed the stack into the model.

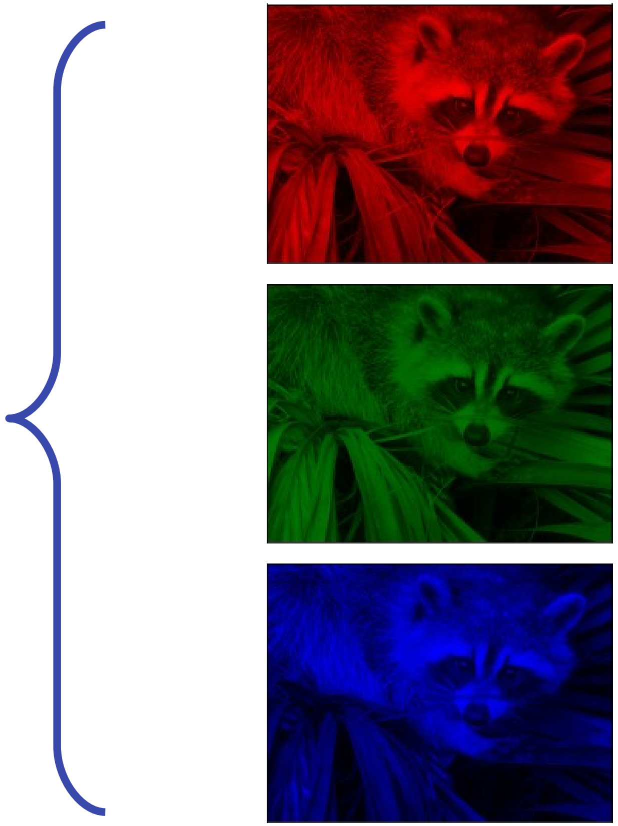
Caveat: model has to be able to deal with 2-d data (e.g., CNNs).

Feature engineering – image data

How can we feed image data into ML models?

- **Whole images**
- **Linearized images**

pixnio via scipy

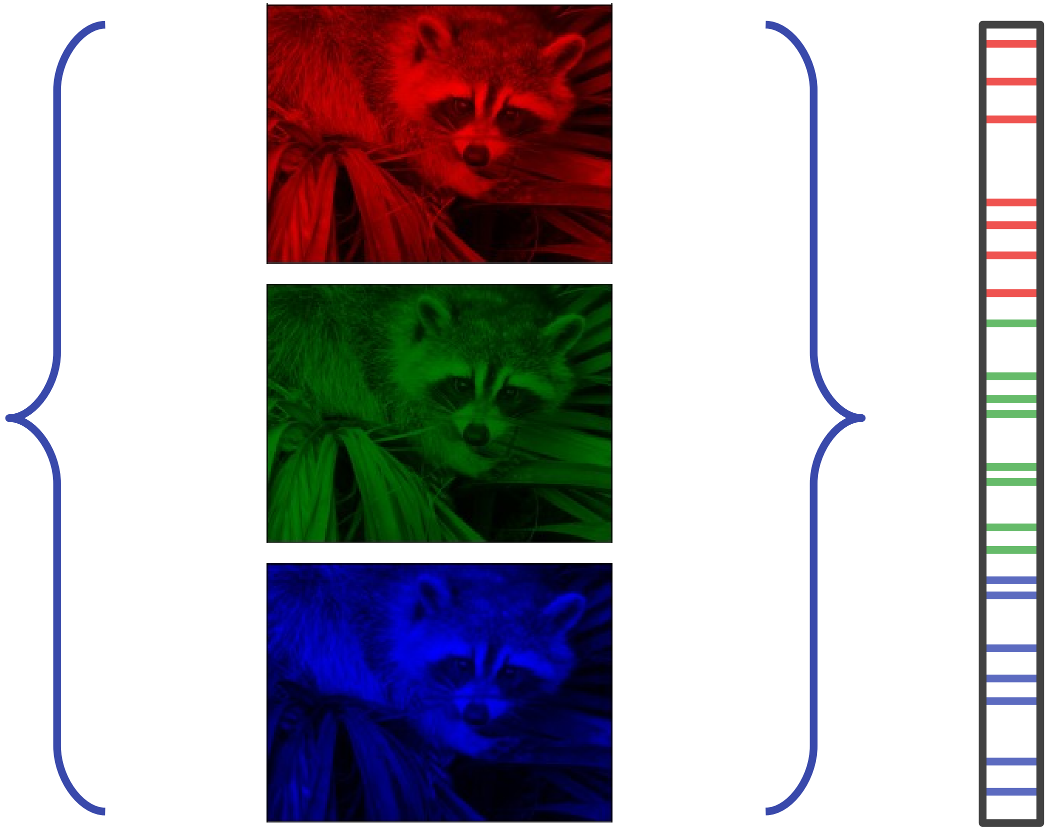


Feature engineering – image data

How can we feed image data into ML models?

- **Whole images**
- **Linearized images**

pixnio via scipy



Feature

Linearize channels and concatenate vectors.

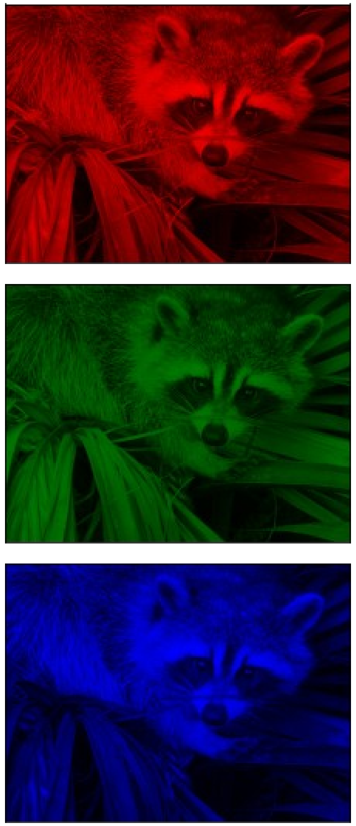
Caveat: spatial information is somewhat lost; works for models that expect linear input data (e.g., MLPs, k-NNs, etc).

Feature engineering – image data

How can we feed image data into ML models?

- **Whole images**
- **Linearized images**
- **Channel histograms**

pixnio via scipy

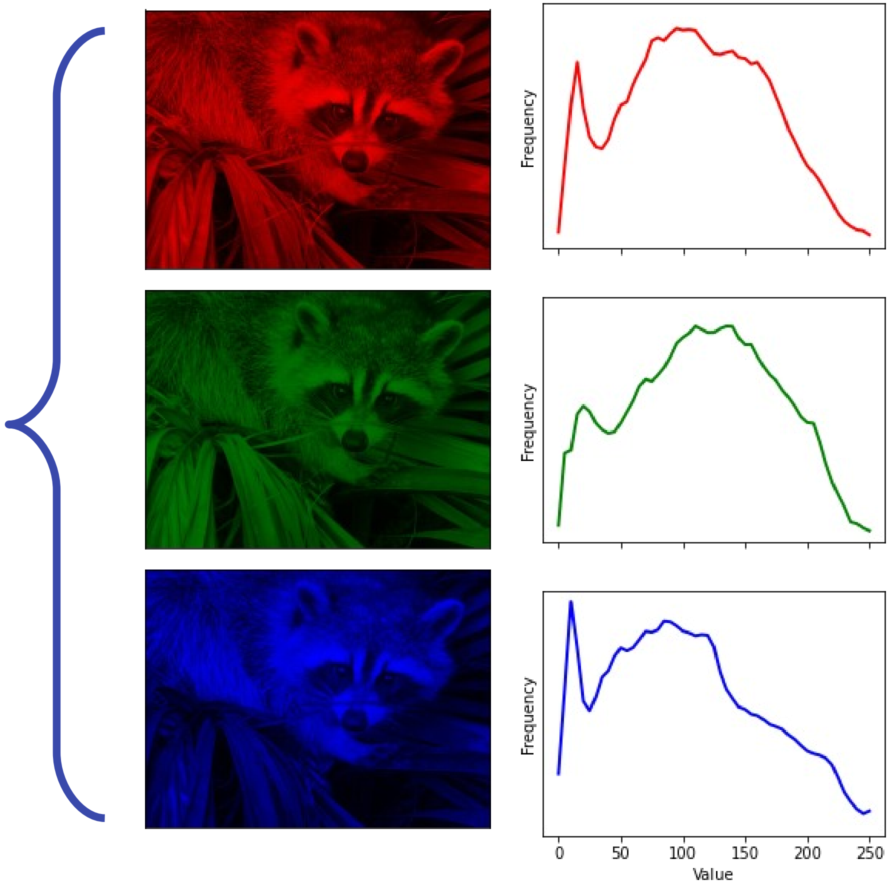


Feature engineering – image data

How can we feed image data into ML models?

- **Whole images**
- **Linearized images**
- **Channel histograms**

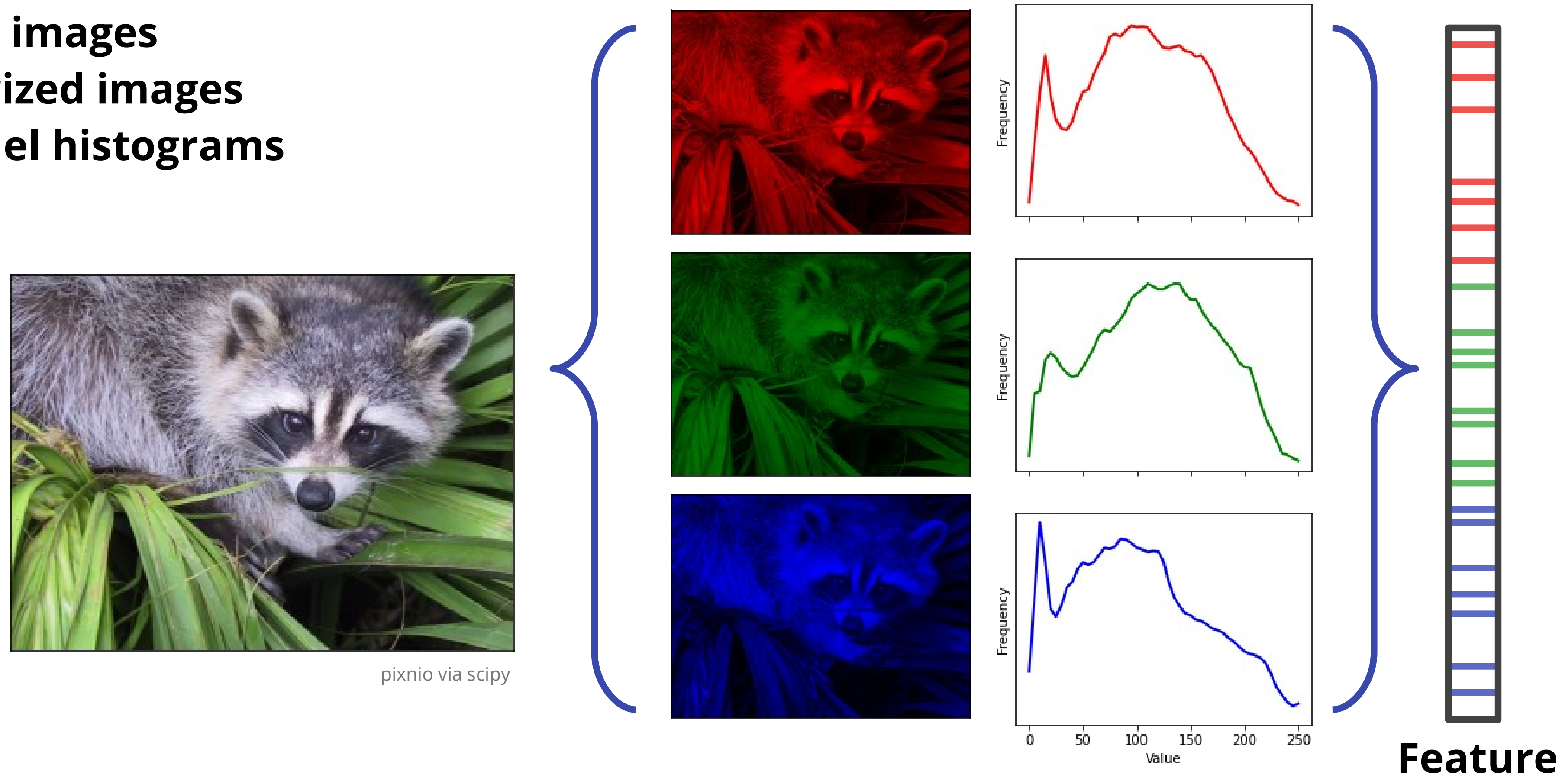
pixnio via scipy



Feature engineering – image data

How can we feed image data into ML models?

- **Whole images**
- **Linearized images**
- **Channel histograms**



pixnio via scipy

Build a **histogram for each channel**. Linearize and concatenate histograms.

Caveat: spatial information is fully lost.

Feature engineering – image data

How can we feed image data into ML models?

- **Whole images**
- **Linearized images**
- **Channel histograms**
- **Visual bag-of-words**

pixnio via scipy

Feature engineering – image data

How can we feed image data into ML models?

- **Whole images**
- **Linearized images**
- **Channel histograms**
- **Visual bag-of-words**

pixnio via scipy

Adopted from **Natural Language Processing** (**NLP**):

1) Split a document into single words (or *n-grams*)

```
John likes rain. It rains a lot here.
```

2) Count the frequency of each word → *bag-of-words*

```
{"John": 1, "likes": 1, "rain": 2, "it": 1,
 "a": 1, "lot":1, "here": 1}
```

Word frequencies that are stored in bags-of-words can be used for document classification.

Feature engineering – image data

How can we feed image data into ML models?

- **Whole images**
- **Linearized images**
- **Channel histograms**
- **Visual bag-of-words**

pixnio via scipy

Adopted from **Natural Language Processing** (**NLP**):

1) Split a document into single words (or *n-grams*)

```
John likes rain. It rains a lot here.
```

2) Count the frequency of each word → *bag-of-words*

```
{"John": 1, "likes": 1, "rain": 2, "it": 1,
 "a": 1, "lot":1, "here": 1}
```

Word frequencies that are stored in bags-of-words can be used for document classification.

How can we apply this concept to image data?

Feature engineering – image data

How can we feed image data into ML models?

- **Whole images**
- **Linearized images**
- **Channel histograms**
- **Visual bag-of-words**

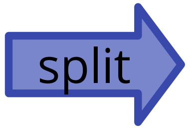
pixnio via scipy

Feature engineering – image data

How can we feed image data into ML models?

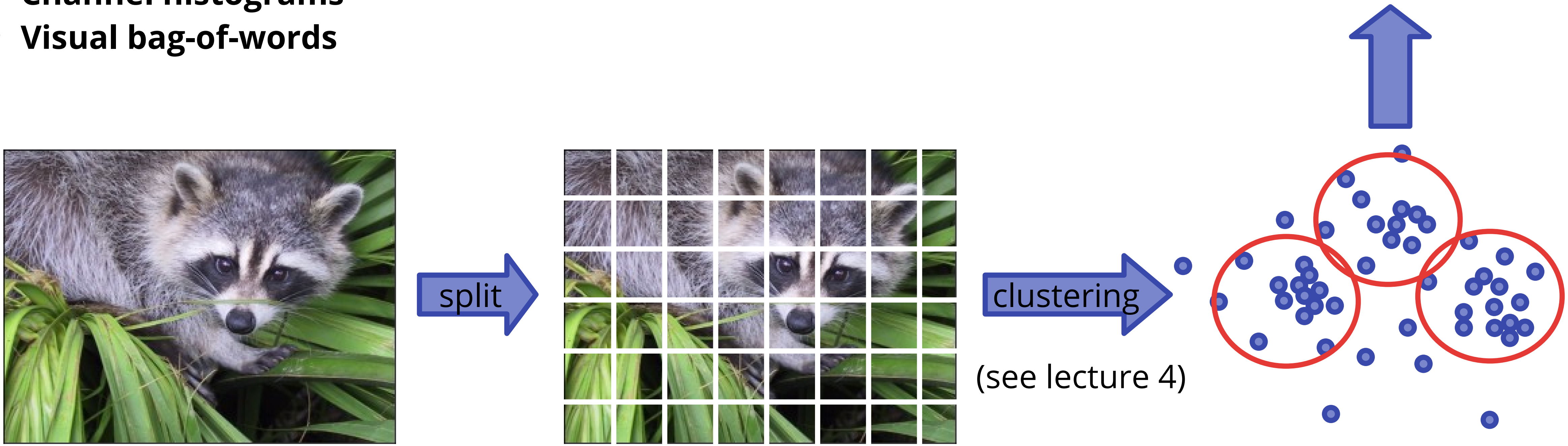
- **Whole images**
- **Linearized images**
- **Channel histograms**
- **Visual bag-of-words**


pixnio via scipy



How can we feed image data into ML models?

- **Whole images**
- **Linearized images**
- **Channel histograms**
- **Visual bag-of-words**



pixnio via scipy

Feature engineering – image data

How can we feed image data into ML models?

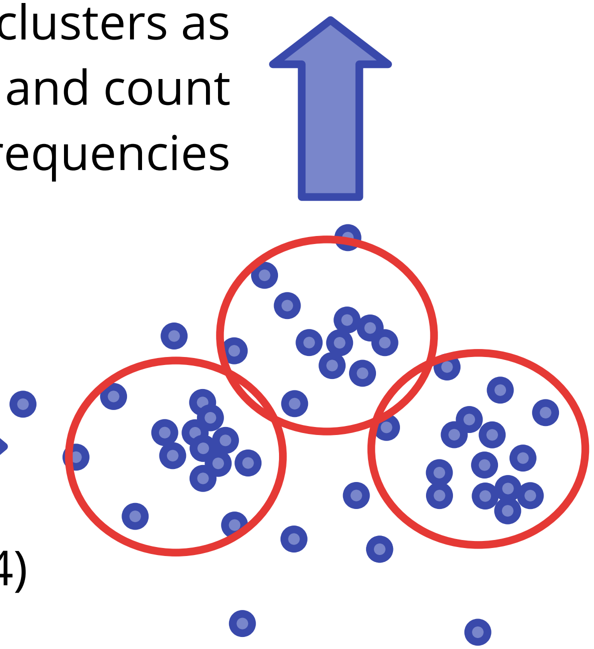
- **Whole images**
- **Linearized images**
- **Channel histograms**
- **Visual bag-of-words**

pixnio via scipy

split

clustering

(see lecture 4)

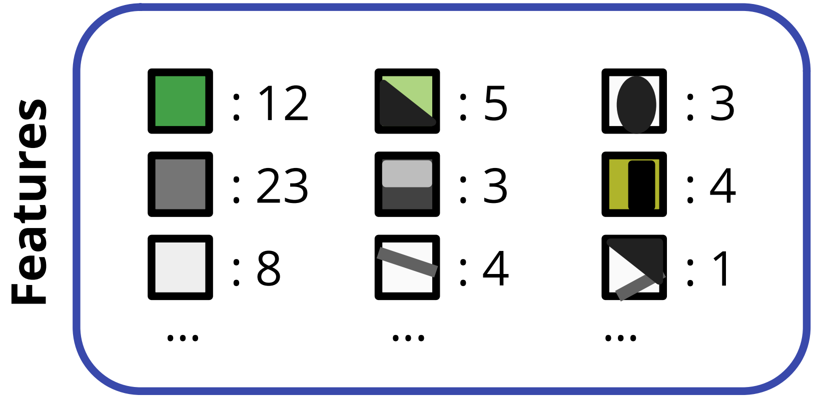


Use clusters as features and count their frequencies

Feature engineering – image data

How can we feed image data into ML models?

- **Whole images**
- **Linearized images**
- **Channel histograms**
- **Visual bag-of-words**



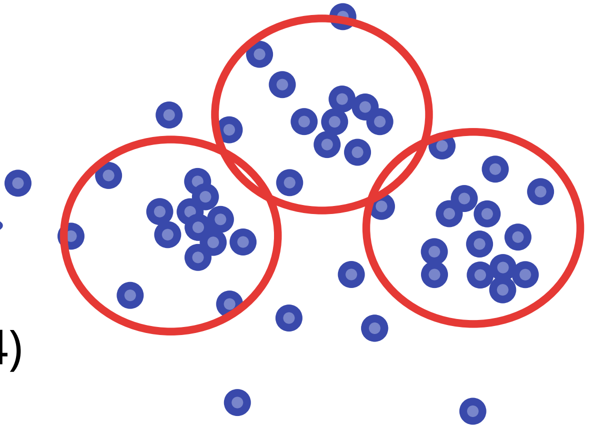
Use clusters as features and count their frequencies

pixnio via scipy

split

clustering

(see lecture 4)



Feature engineering – image data

How can we feed image data into ML models?

- **Whole images**
- **Linearized images**
- **Channel histograms**
- **Visual bag-of-words**
- **Histogram of oriented gradients (HOG)**

pixnio via scipy

split into 8x8 pixel **cells**

How can we feed image data into ML models?

- **Whole images**
- **Linearized images**
- **Channel histograms**
- **Visual bag-of-words**
- **Histogram of oriented gradients (HOG)**

pixnio via scipy

split

split into 8x8 pixel **cells**

derive gradients

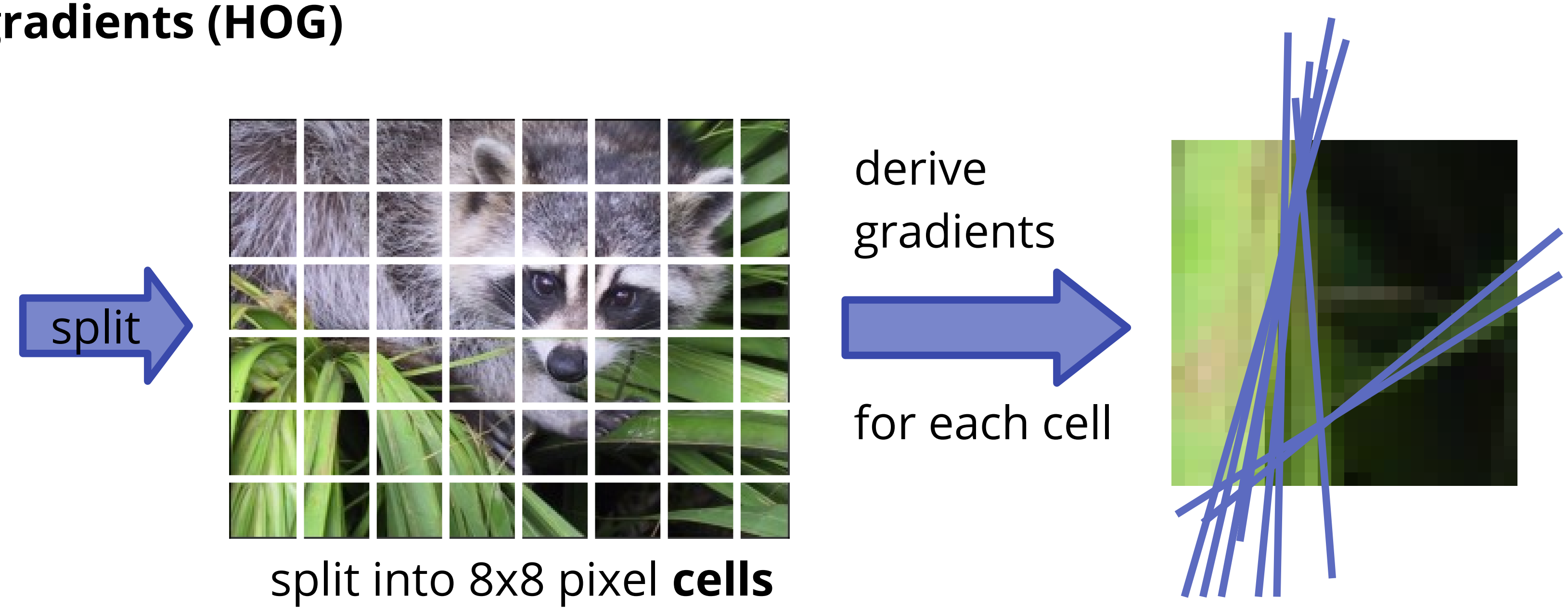
for each cell

Feature engineering – image data

How can we feed image data into ML models?

- **Whole images**
- **Linearized images**
- **Channel histograms**
- **Visual bag-of-words**
- **Histogram of oriented gradients (HOG)**

pixnio via scipy

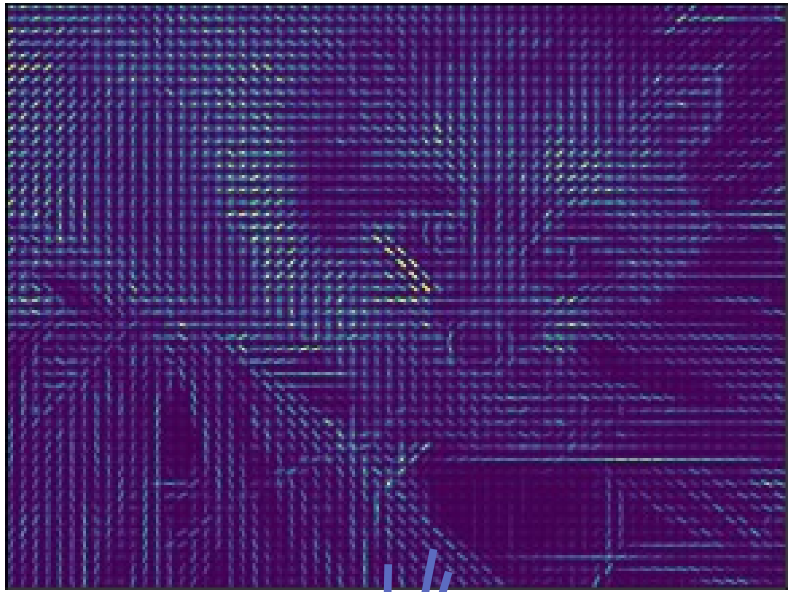


split into 8x8 pixel **cells**

Feature engineering – image data

How can we feed image data into ML models?

- **Whole images**
- **Linearized images**
- **Channel histograms**
- **Visual bag-of-words**
- **Histogram of oriented gradients (HOG)**



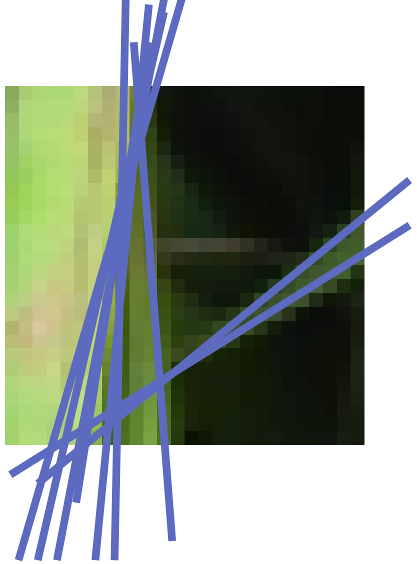
pixnio via scipy

split

split into 8x8 pixel **cells**

derive gradients

for each cell



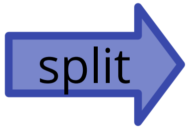
Feature engineering – image data

How can we feed image data into ML models?

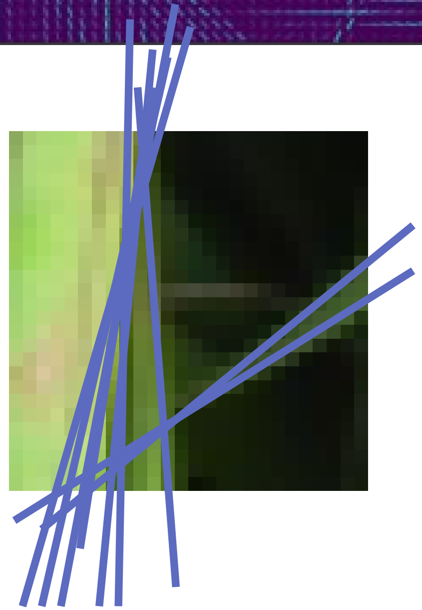
- **Whole images**
- **Linearized images**
- **Channel histograms**
- **Visual bag-of-words**
- **Histogram of oriented gradients (HOG)**

For each cell, create a **histogram of gradients** as feature.

pixnio via scipy



split into 8x8 pixel **cells**



Feature engineering – image data

How can we feed image data into ML models?

- **Whole images**
- **Linearized images**
- **Channel histograms**
- **Visual bag-of-words**
- **Histogram of oriented gradients (HOG)**

Feature engineering – image data

How can we feed image data into ML models?

- **Whole images**
- **Linearized images**
- **Channel histograms**
- **Visual bag-of-words**
- **Histogram of oriented gradients (HOG)**

But which method works best?

Feature engineering – image data

How can we feed image data into ML models?

- **Whole images**
- **Linearized images**
- **Channel histograms**
- **Visual bag-of-words**
- **Histogram of oriented gradients (HOG)**

But which method works best?

It depends. Some offer more information than others but they generally describe different concepts.

Feature engineering – image data

How can we feed image data into ML models?

- **Whole images**
- **Linearized images**
- **Channel histograms**
- **Visual bag-of-words**
- **Histogram of oriented gradients (HOG)**

But which method works best?

It depends. Some offer more information than others but they generally describe different concepts.

Which features work best depends on your task and your data set.

Final data set nomenclature

Final data set nomenclature

Feature engineering results in a compilation of features that we can use to train our ML models.

Example:

Weight	Height	Wings	Legs	Cuteness
0.1	0.1	true	2	1
3.5	0.3	false	4	1
12.0	0.7	false	4	1
500	1.8	false	4	2
800	3.0	true	4	3
...	...	...	...	...

Pet	Type
true	bird
true	cat
true	dog
false	rhinoceros
false	chimera
...	...

Final data set nomenclature

Feature engineering results in a compilation of features that we can use to train our ML models.

Example:

Features/Attributes (input variables, $\boldsymbol{x}$) $f(\boldsymbol{x}) = \boldsymbol{y}$ **Targets/Labels** (output variables, $\boldsymbol{y}$)

Ground-Truth

Weight	Height	Wings	Legs	Cuteness
0.1	0.1	true	2	1
3.5	0.3	false	4	1
12.0	0.7	false	4	1
500	1.8	false	4	2
800	3.0	true	4	3
...	...	...	...	...

Pet	Type
true	bird
true	cat
true	dog
false	rhinoceros
false	chimera
...	...

Final data set nomenclature

Feature engineering results in a compilation of features that we can use to train our ML models.

Example:

Features/Attributes (input variables, $\boldsymbol{x}$) $f(\boldsymbol{x}) = \boldsymbol{y}$ **Targets/Labels** (output variables, $\boldsymbol{y}$)

Ground-Truth

Samples/Instances

Weight	Height	Wings	Legs	Cuteness
0.1	0.1	true	2	1
3.5	0.3	false	4	1
12.0	0.7	false	4	1
500	1.8	false	4	2
800	3.0	true	4	3
...	...	...	...	...

Pet	Type
true	bird
true	cat
true	dog
false	rhinoceros
false	chimera
...	...

Final data set nomenclature

Feature engineering results in a compilation of features that we can use to train our ML models.

Example:

Features/Attributes (input variables, $\boldsymbol{x}$)

$$f(\boldsymbol{x}) = \boldsymbol{y}$$

Targets/Labels (output variables, $\boldsymbol{y}$)

Ground-Truth

Samples/Instances

Weight	Height	Wings	Legs	Cuteness
0.1	0.1	true	2	1
3.5	0.3	false	4	1
12.0	0.7	false	4	1
500	1.8	false	4	2
800	3.0	true	4	3
...	...	...	...	...

Pet	Type
true	bird
true	cat
true	dog
false	rhinoceros
false	chimera
...	...

classes of label "Type"

Final data set nomenclature

Feature engineering results in a compilation of features that we can use to train our ML models.

Example:

Features/Attributes (input variables, $\boldsymbol{x}$) $f(\boldsymbol{x}) = \boldsymbol{y}$ **Targets/Labels** (output variables, $\boldsymbol{y}$)

Ground-Truth

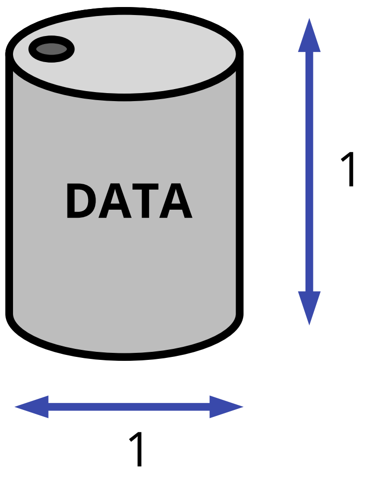
Samples/Instances

Weight	Height	Wings	Legs	Cuteness	Pet	Type
0.1	0.1	true	2	1	true	bird
3.5	0.3	false	4	1	true	cat
12.0	0.7	false	4	1	true	dog
500	1.8	false	4	2	false	rhinoceros
800	3.0	true	4	3	false	chimera
...	...	...	...	...	...	...
continuous	continuous	binary	discrete	ordinal	binary	categorical (multi-class)

classes of label "Type"

Data Types:

Data scaling



Data scaling

Data scaling means to linearly transform your data in order to normalize them.

Data scaling

Data scaling means to linearly transform your data in order to normalize them.

Why scale data?

Data scaling

Data scaling means to linearly transform your data in order to normalize them.

Why scale data?

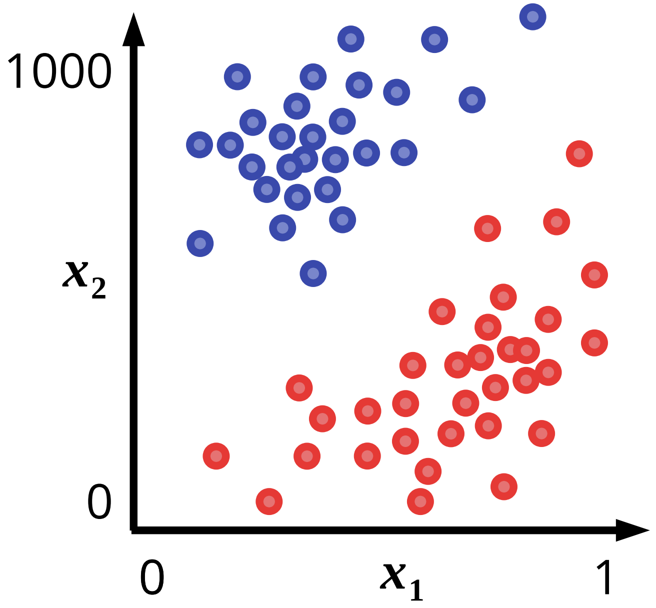
- Many ML models are based on a notion of “distance” between samples; improperly scaled data may jeopardize the learning capability of such models.

Data scaling

Data scaling means to linearly transform your data in order to normalize them.

Why scale data?

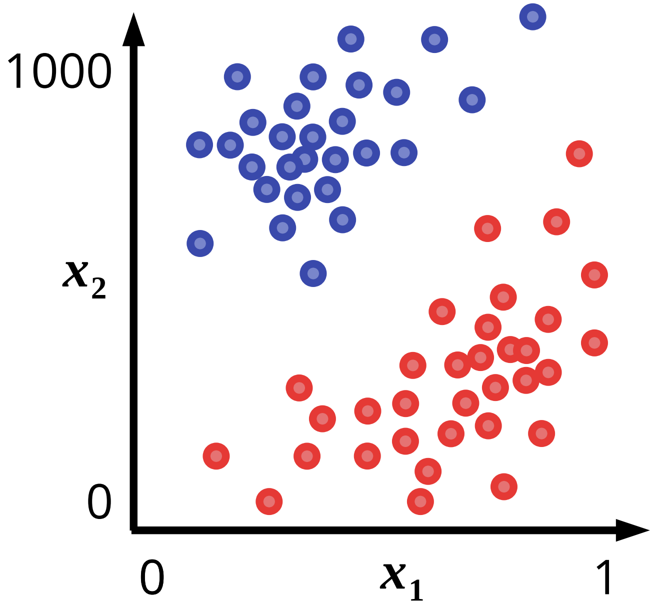
- Many ML models are based on a notion of “distance” between samples; improperly scaled data may jeopardize the learning capability of such models.



Data scaling means to linearly transform your data in order to normalize them.

Why scale data?

- Many ML models are based on a notion of "distance" between samples; improperly scaled data may jeopardize the learning capability of such models.



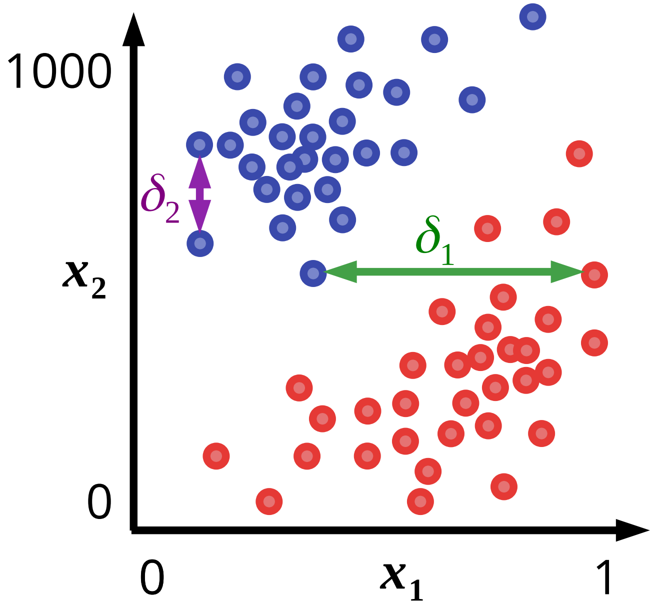
$$\delta = \sqrt{x_1^2 + x_2^2}$$

Euclidean distance metric

Data scaling means to linearly transform your data in order to normalize them.

Why scale data?

- Many ML models are based on a notion of “distance” between samples; improperly scaled data may jeopardize the learning capability of such models.



$$\delta = \sqrt{x_1^2 + x_2^2}$$

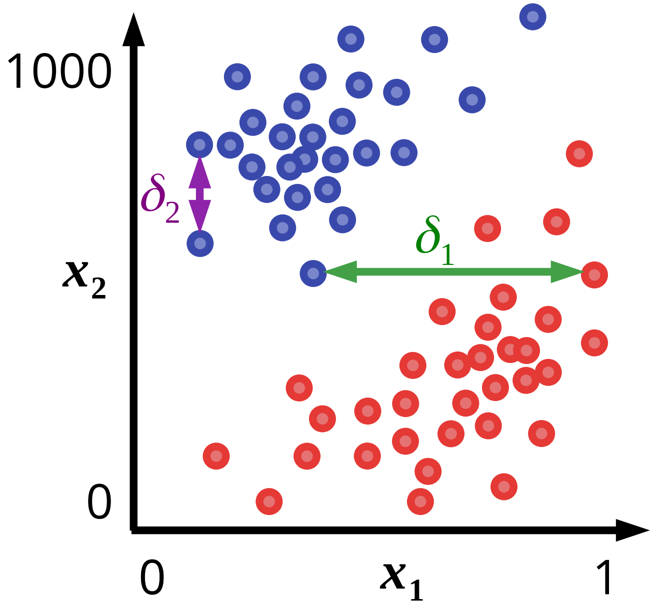
Euclidean distance metric

$$\delta_1 \ll \delta_2$$

Data scaling means to linearly transform your data in order to normalize them.

Why scale data?

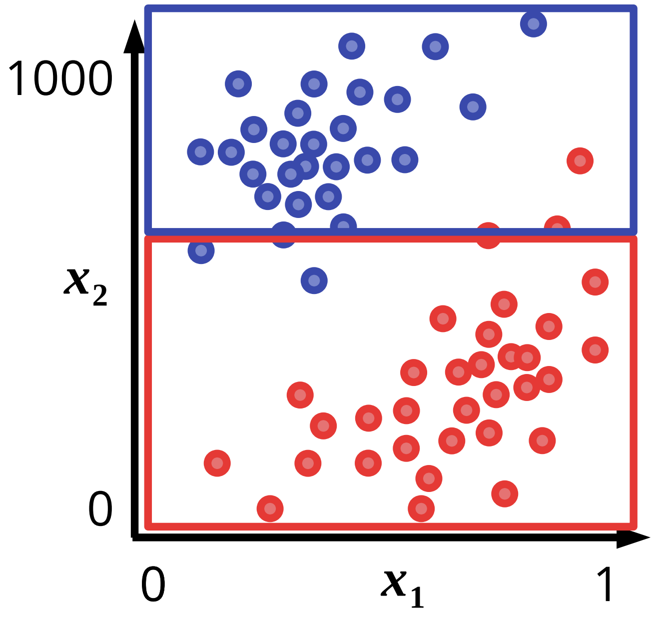
- Many ML models are based on a notion of “distance” between samples; improperly scaled data may jeopardize the learning capability of such models.



$$\delta = \sqrt{x_1^2 + x_2^2}$$

Euclidean distance metric

$\delta_1 \ll \delta_2$



Decision regions of a hypothetical distance-based classifier.

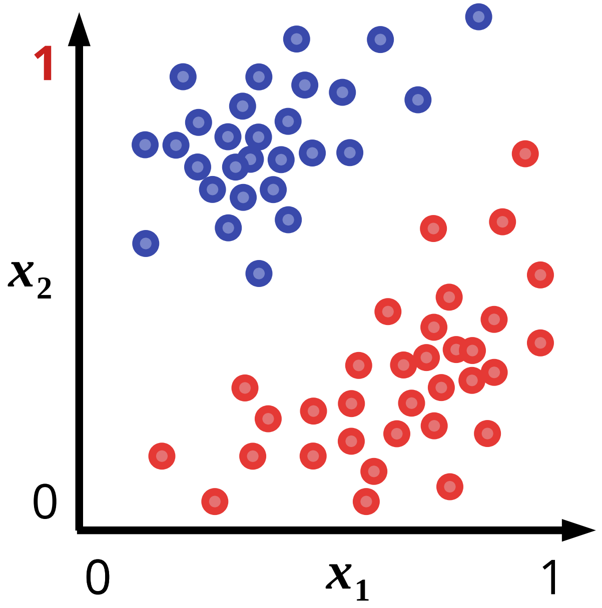
Results are ok-ish, but could be much better…

Data scaling

Data scaling means to linearly transform your data in order to standardize them.

Why scale data?

- Many ML models are based on a notion of “distance” between samples; improperly scaled data may jeopardize the learning capability of such models.



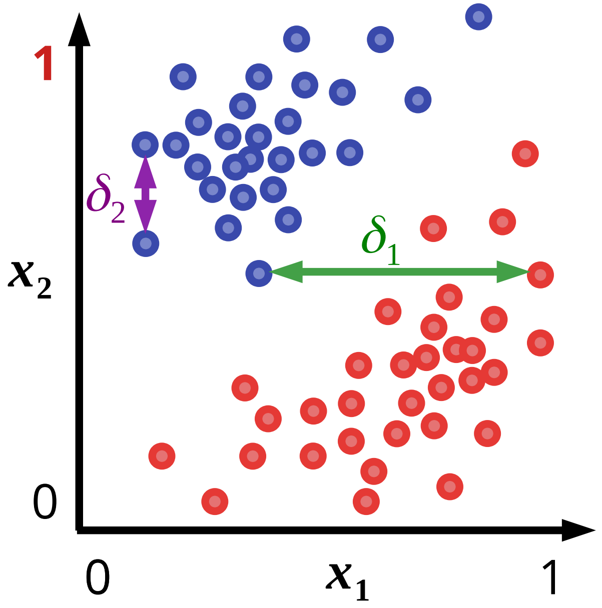
$$\delta = \sqrt{x_1^2 + x_2^2}$$

Euclidean distance metric

Data scaling means to linearly transform your data in order to standardize them.

Why scale data?

- Many ML models are based on a notion of "distance" between samples; improperly scaled data may jeopardize the learning capability of such models.



$$\delta = \sqrt{x_1^2 + x_2^2}$$

Euclidean distance metric

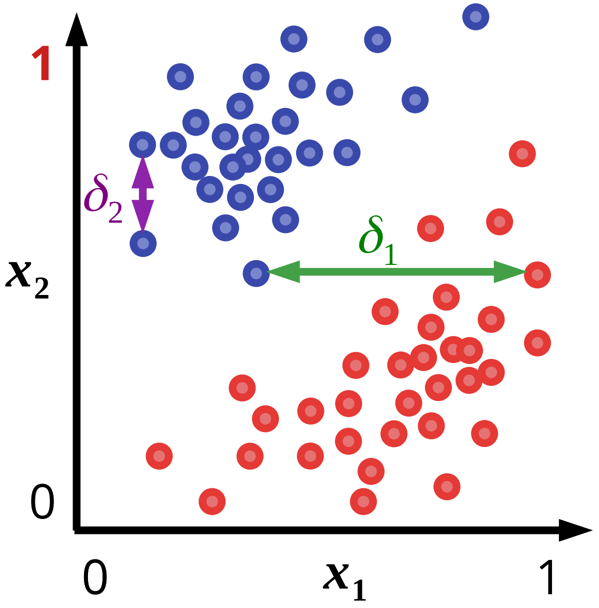
$\delta_1 > \delta_2$

Data scaling

Data scaling means to linearly transform your data in order to standardize them.

Why scale data?

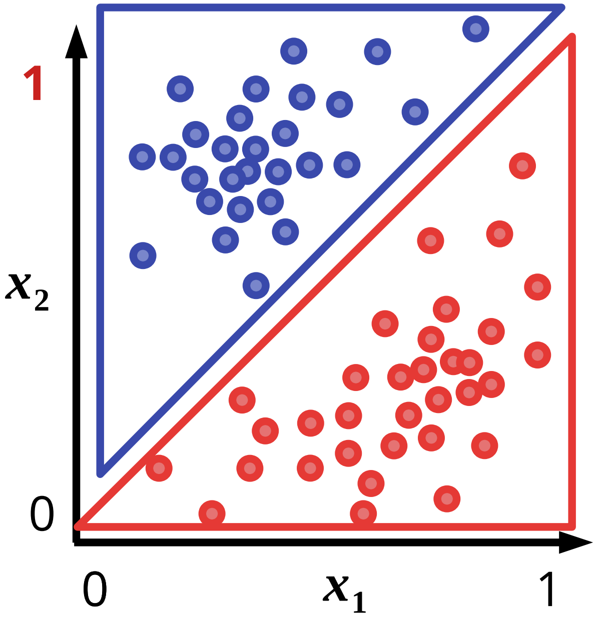
- Many ML models are based on a notion of “distance” between samples; improperly scaled data may jeopardize the learning capability of such models.



$$\delta = \sqrt{x_1^2 + x_2^2}$$

Euclidean distance metric

$\delta_1 > \delta_2$



Decision regions of a hypothetical distance-based classifier.

This is much better!

Data should be scaled!

Data scaling

Data scaling means to linearly transform your data in order to standardize them.

Why scale data?

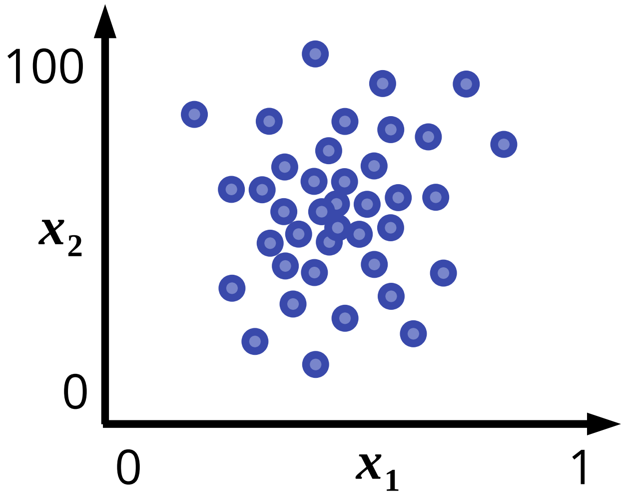
- Many ML models are based on a notion of "distance" between samples; improperly scaled data may jeopardize the learning capability of such models.
- Some ML models intrinsically presume that data are distributed following a Gaussian fashion with similar variances along all features; high variance along one feature leads to bias.

Data scaling

Data scaling means to linearly transform your data in order to standardize them.

Why scale data?

- Many ML models are based on a notion of “distance” between samples; improperly scaled data may jeopardize the learning capability of such models.
- Some ML models intrinsically presume that data are distributed following a Gaussian fashion with similar variances along all features; high variance along one feature leads to bias.

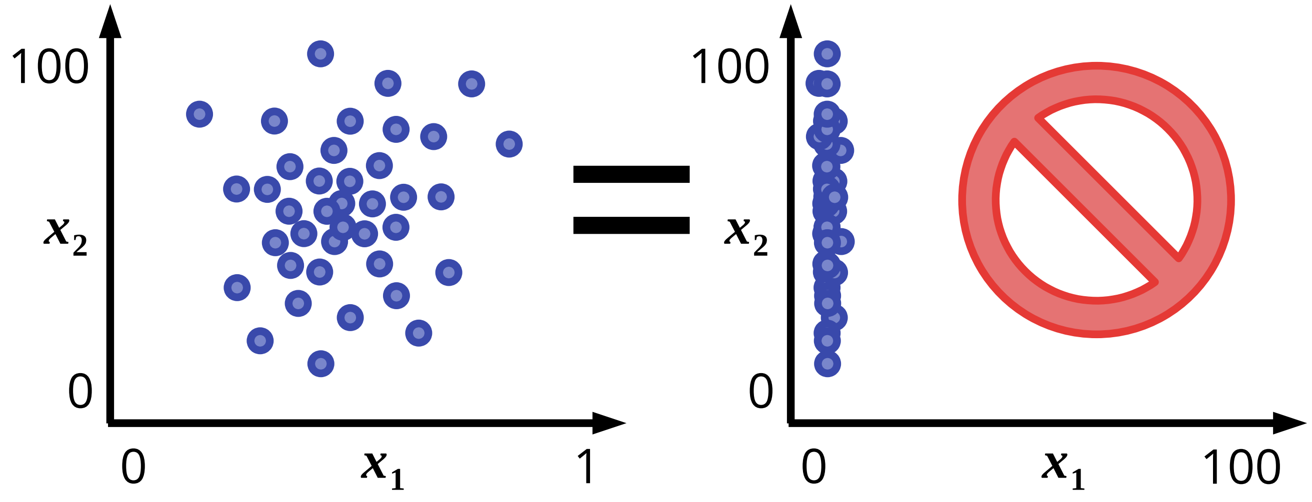


Data scaling

Data scaling means to linearly transform your data in order to standardize them.

Why scale data?

- Many ML models are based on a notion of "distance" between samples; improperly scaled data may jeopardize the learning capability of such models.
- Some ML models intrinsically presume that data are distributed following a Gaussian fashion with similar variances along all features; high variance along one feature leads to bias.

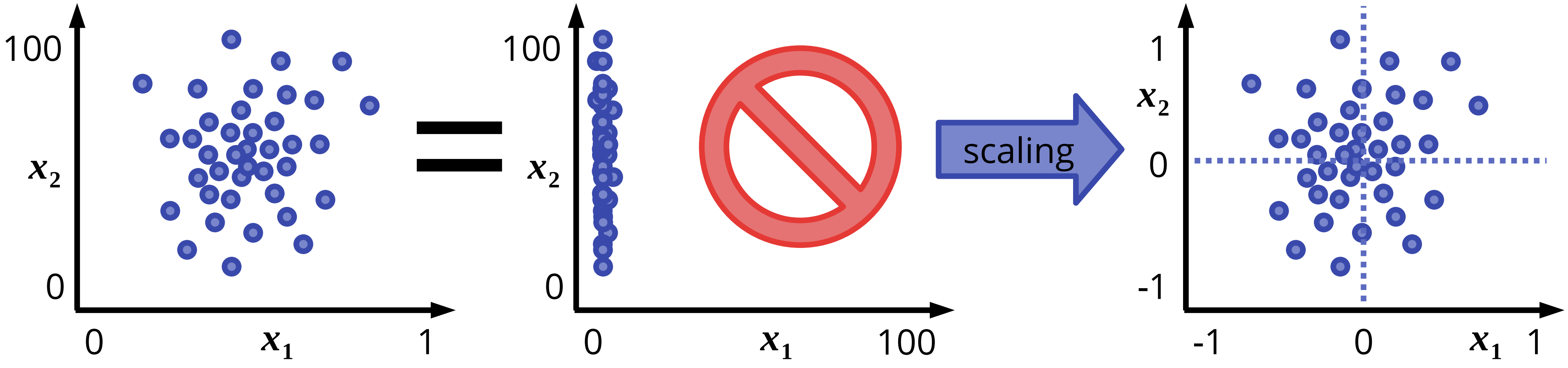


Data scaling

Data scaling means to linearly transform your data in order to standardize them.

Why scale data?

- Many ML models are based on a notion of “distance” between samples; improperly scaled data may jeopardize the learning capability of such models.
- Some ML models intrinsically presume that data are distributed following a Gaussian fashion with similar variances along all features; high variance along one feature leads to bias.



Data scaling

Data scaling means to linearly transform your data in order to standardize them.

Why scale data?

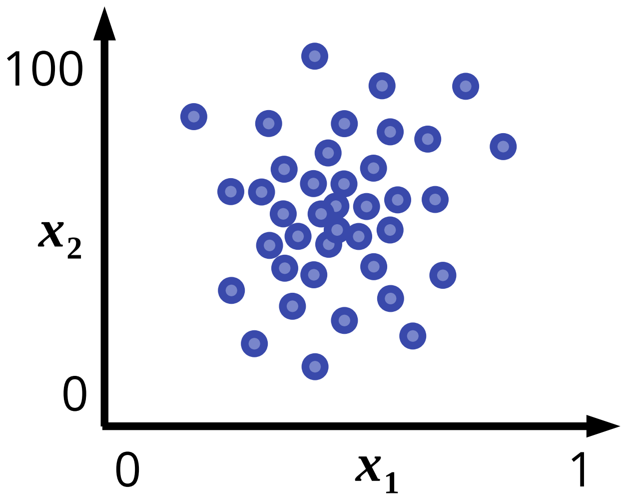
- Many ML models are based on a notion of “distance” between samples; improperly scaled data may jeopardize the learning capability of such models.
- Some ML models intrinsically presume that data are distributed following a Gaussian fashion with similar variances along all features; high variance along one feature leads to bias.

How to scale data?

- Normalize feature variances (to give similar weights to the different features)
- Normalize feature mean values (assumed by a number of ML models)

Data scaling – MinMax scaler

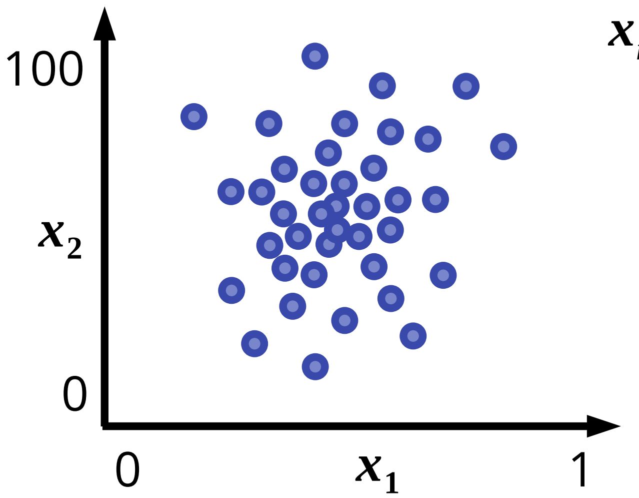
Scale every feature onto a range from 0 to 1 based on the minimum and maximum of the underlying distribution.



$x_i \in \boldsymbol{X_i}$

Data scaling – MinMax scaler

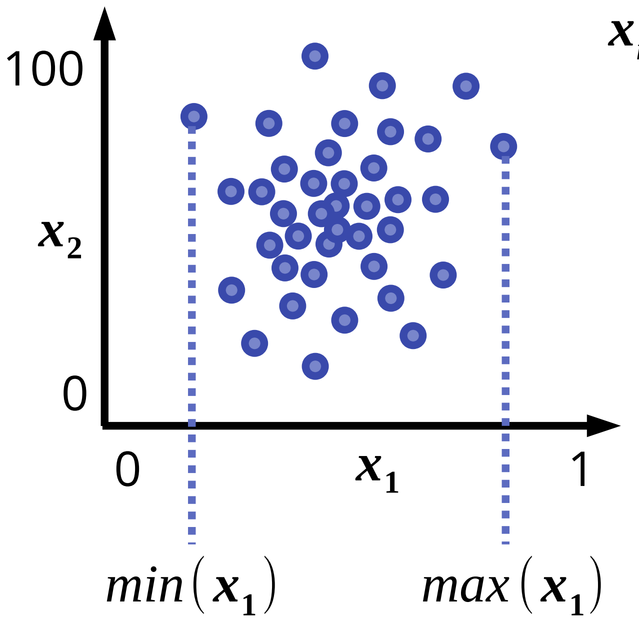
Scale every feature onto a range from 0 to 1 based on the minimum and maximum of the underlying distribution.



$$x_i' = \frac{x_i - min(X_i)}{max(X_i) - min(X_i)}$$

$$x_i \in X_i$$

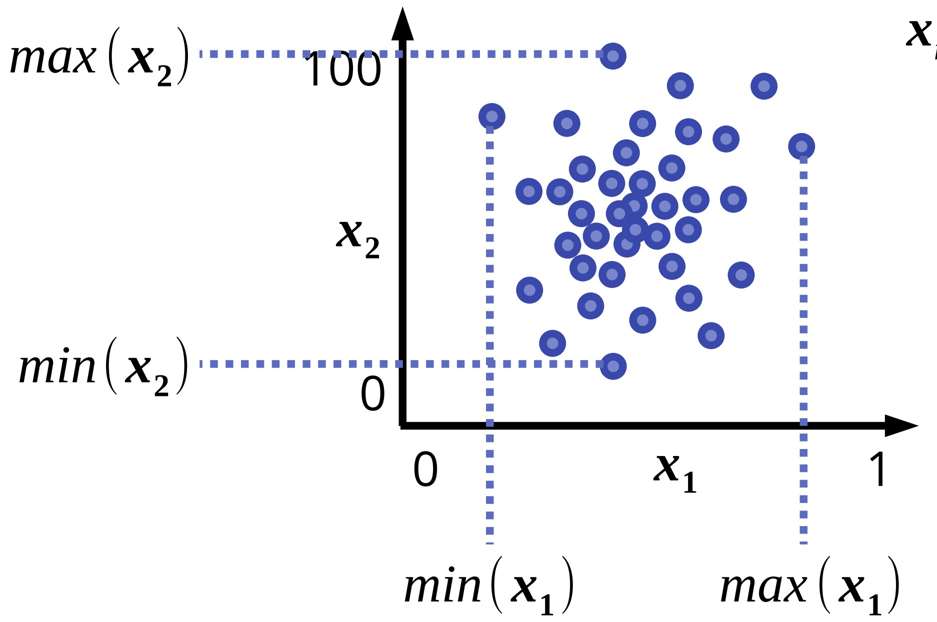
Scale every feature onto a range from 0 to 1 based on the minimum and maximum of the underlying distribution.



$$x_i' = \frac{x_i - min(X_i)}{max(X_i) - min(X_i)}$$

$$x_i \in X_i$$

Scale every feature onto a range from 0 to 1 based on the minimum and maximum of the underlying distribution.

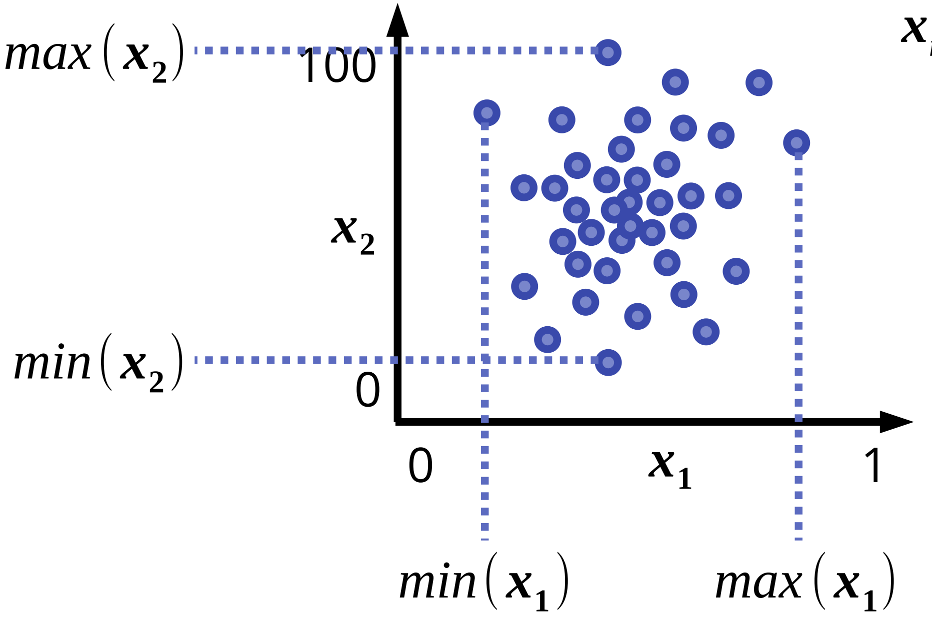


$$\boldsymbol{x}_i{}' = \frac{\boldsymbol{x}_i - min(\boldsymbol{X}_i)}{max(\boldsymbol{X}_i) - min(\boldsymbol{X}_i)}$$

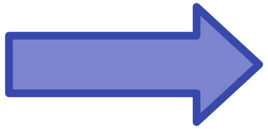
$$x_i \in \boldsymbol{X}_i$$

Data scaling – MinMax scaler

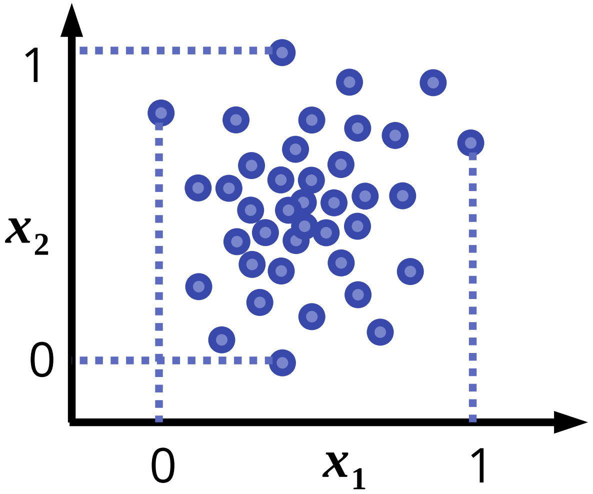
Scale every feature onto a range from 0 to 1 based on the minimum and maximum of the underlying distribution.



$$x_i' = \frac{x_i - min(X_i)}{max(X_i) - min(X_i)}$$



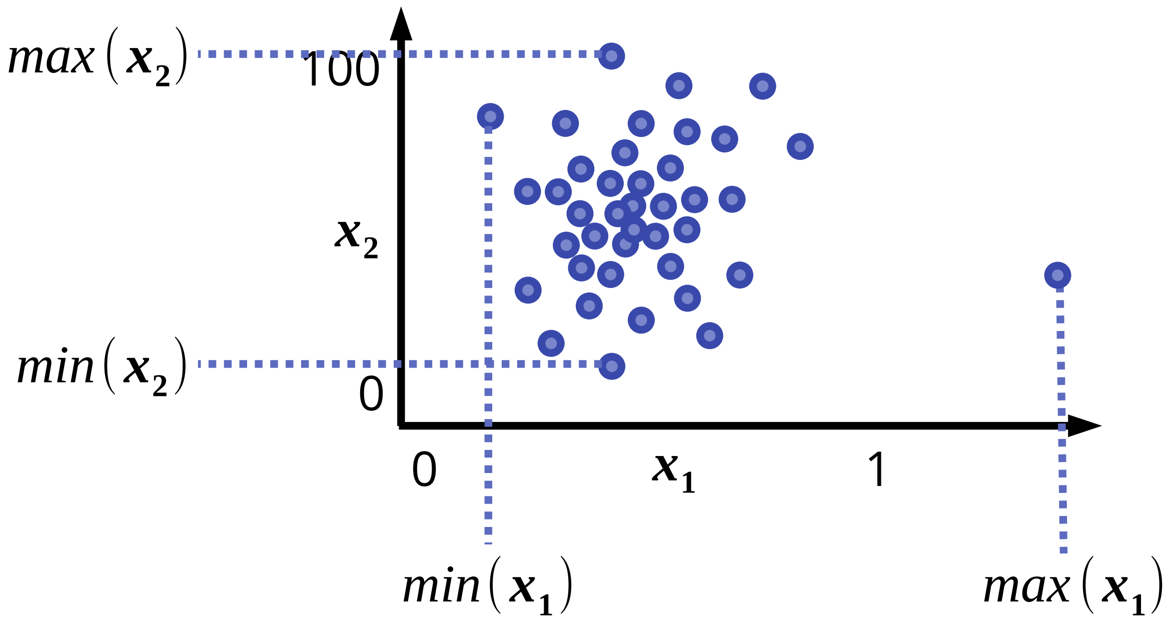
$$x_i \in X_i$$



Data scaling – MinMax scaler

Scale every feature onto a range from 0 to 1 based on the minimum and maximum of the underlying distribution.

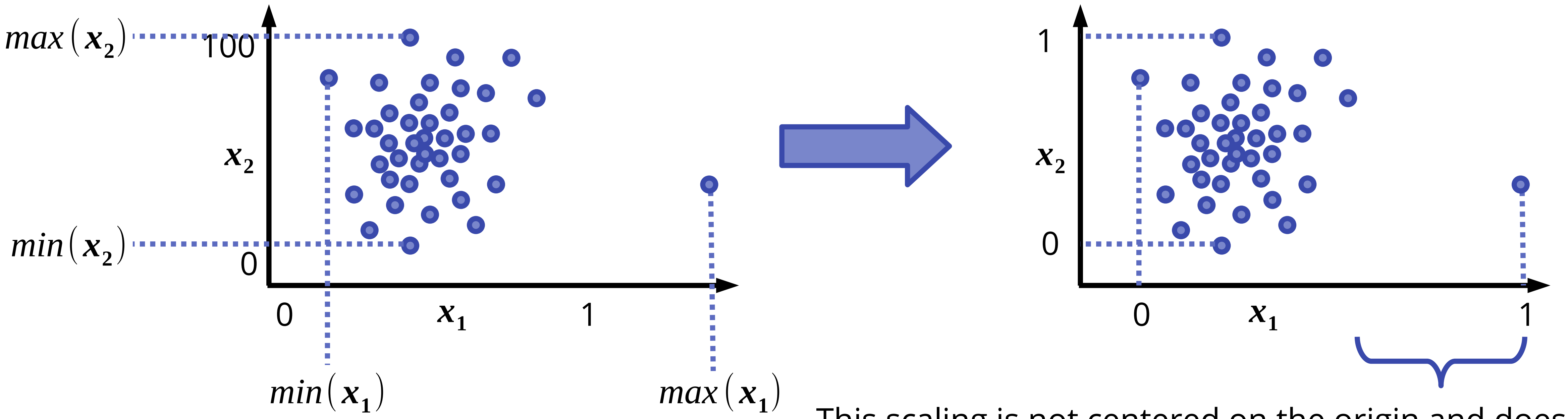
Disadvantage: the MinMax scaler is prone to outliers and does not center the distribution in the origin.



Data scaling – MinMax scaler

Scale every feature onto a range from 0 to 1 based on the minimum and maximum of the underlying distribution.

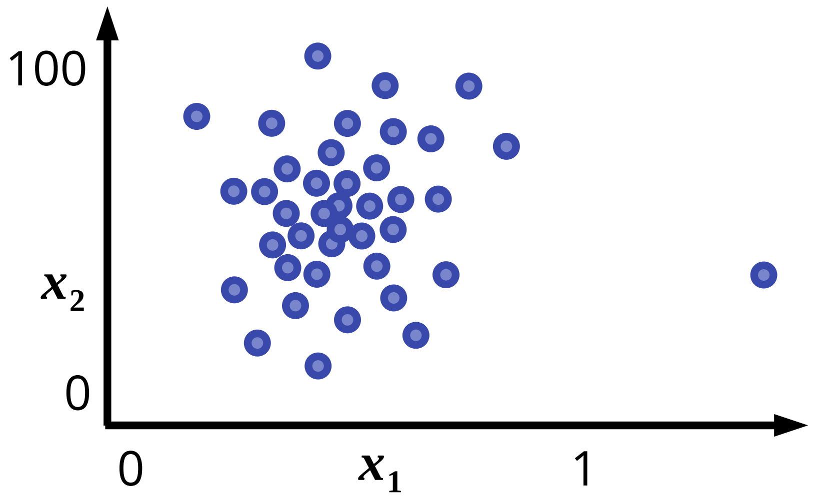
Disadvantage: the MinMax scaler is prone to outliers and does not center the distribution in the origin.



This scaling is not centered on the origin and does not describe the data distribution well.

Data scaling – Standard scaler

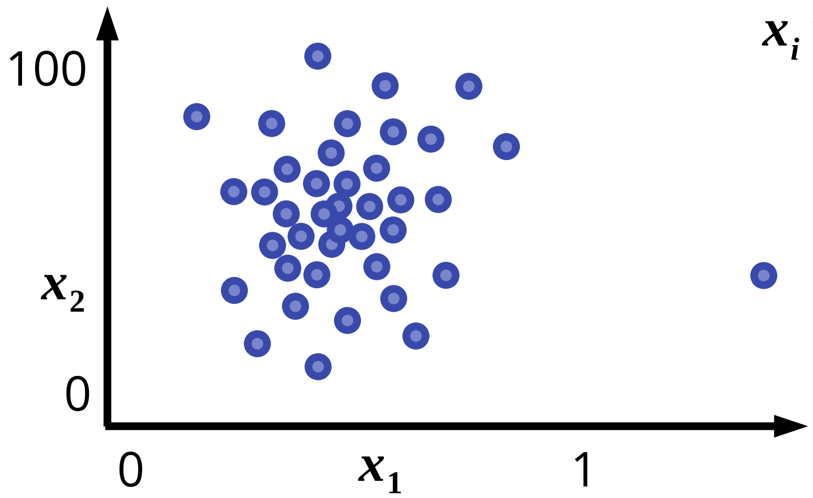
Scale every feature onto a variable range based on the mean and standard deviation of the underlying distribution.



$x_i \in \boldsymbol{X_i}$

Data scaling – Standard scaler

Scale every feature onto a variable range based on the mean and standard deviation of the underlying distribution.

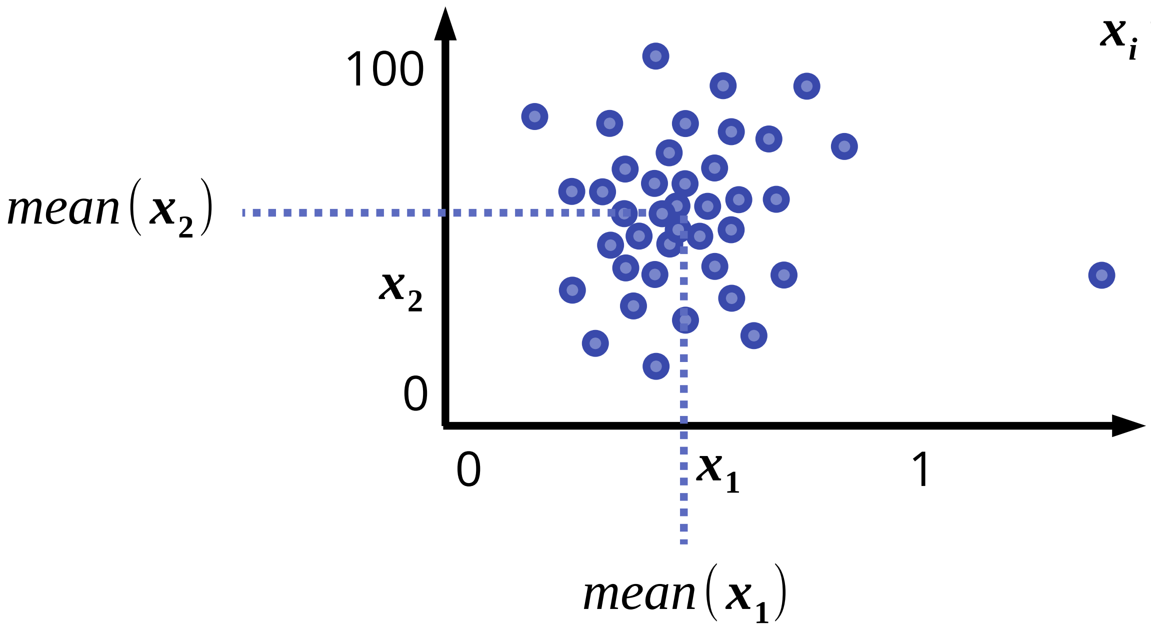


$$\boldsymbol{x_i}' = \frac{\boldsymbol{x_i} - mean(\boldsymbol{X_i})}{\sigma(\boldsymbol{X_i})}$$

$$x_i \in \boldsymbol{X_i}$$

Data scaling – Standard scaler

Scale every feature onto a variable range based on the mean and standard deviation of the underlying distribution.

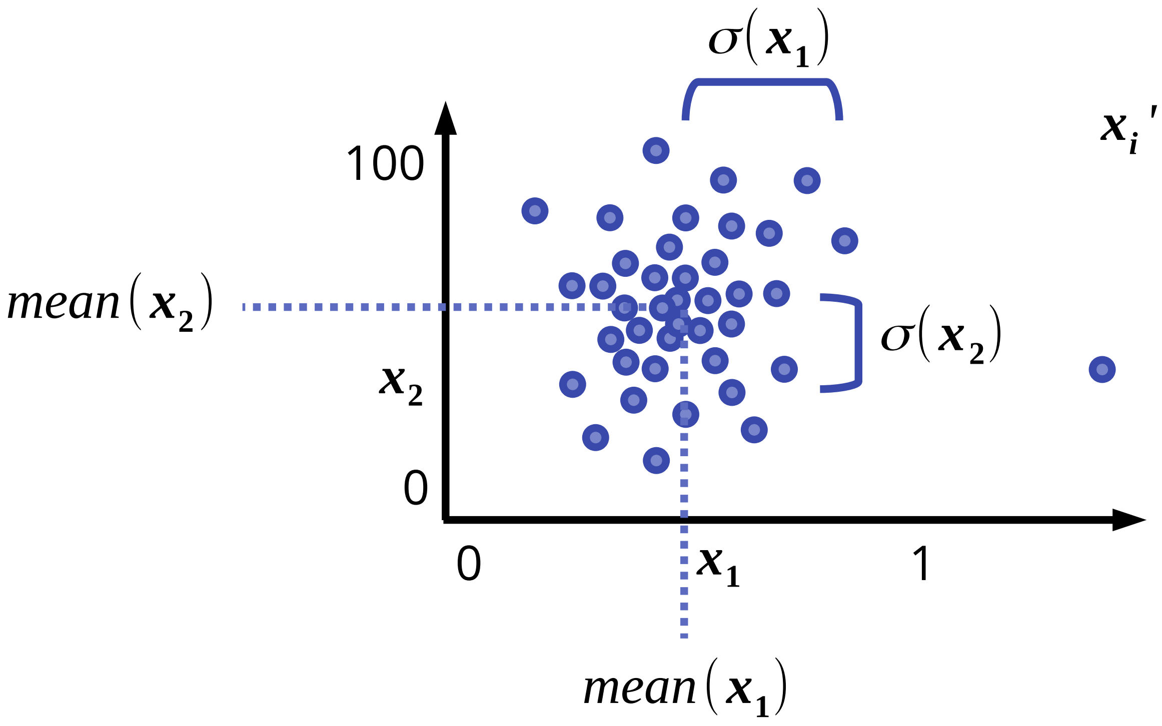


$$\boldsymbol{x}_i' = \frac{\boldsymbol{x}_i - mean(\boldsymbol{X}_i)}{\sigma(\boldsymbol{X}_i)}$$

$$x_i \in \boldsymbol{X}_i$$

Data scaling – Standard scaler

Scale every feature onto a variable range based on the mean and standard deviation of the underlying distribution.

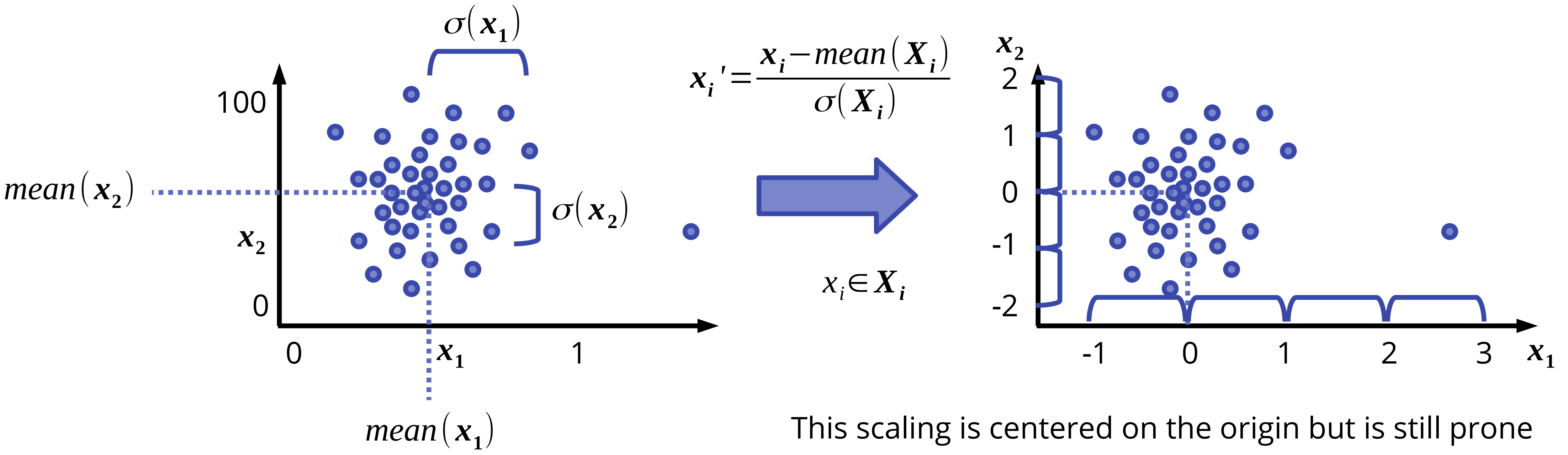


$$x_i' = \frac{x_i - mean(X_i)}{\sigma(X_i)}$$

$$x_i \in X_i$$

Data scaling – Standard scaler

Scale every feature onto a variable range based on the mean and standard deviation of the underlying distribution.



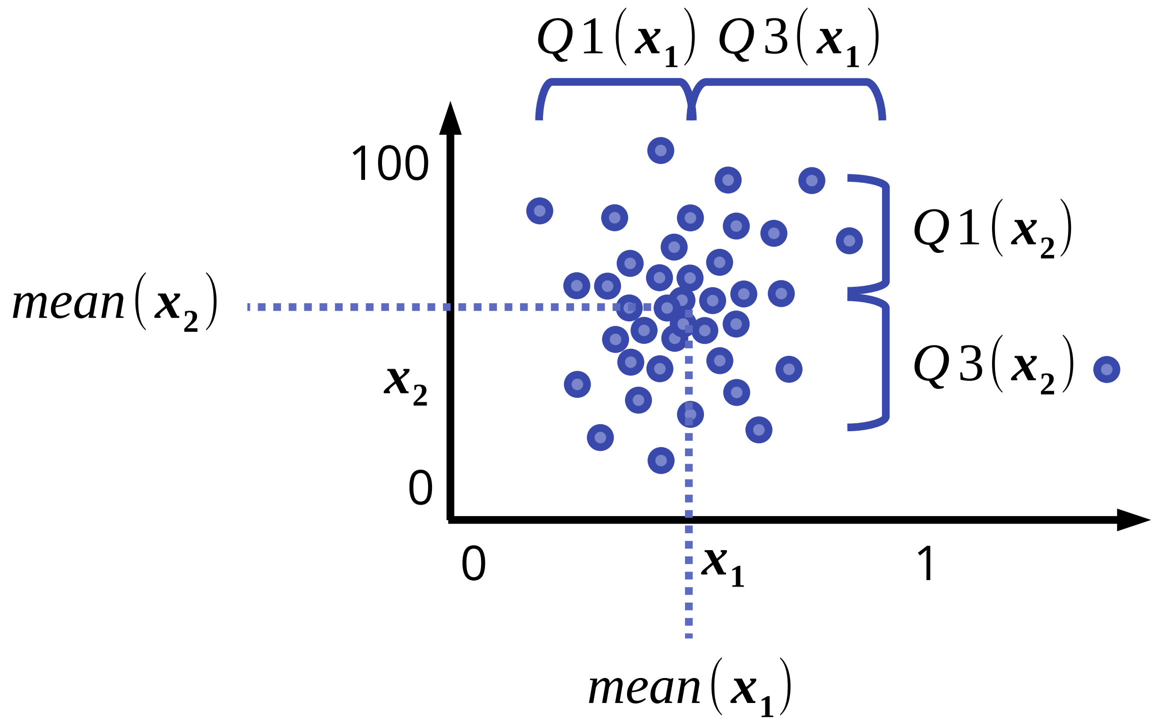
$$\boldsymbol{x}_i' = \frac{\boldsymbol{x}_i - mean(\boldsymbol{X}_i)}{\sigma(\boldsymbol{X}_i)}$$

$$x_i \in \boldsymbol{X}_i$$

This scaling is centered on the origin but is still prone to outliers to some extent.

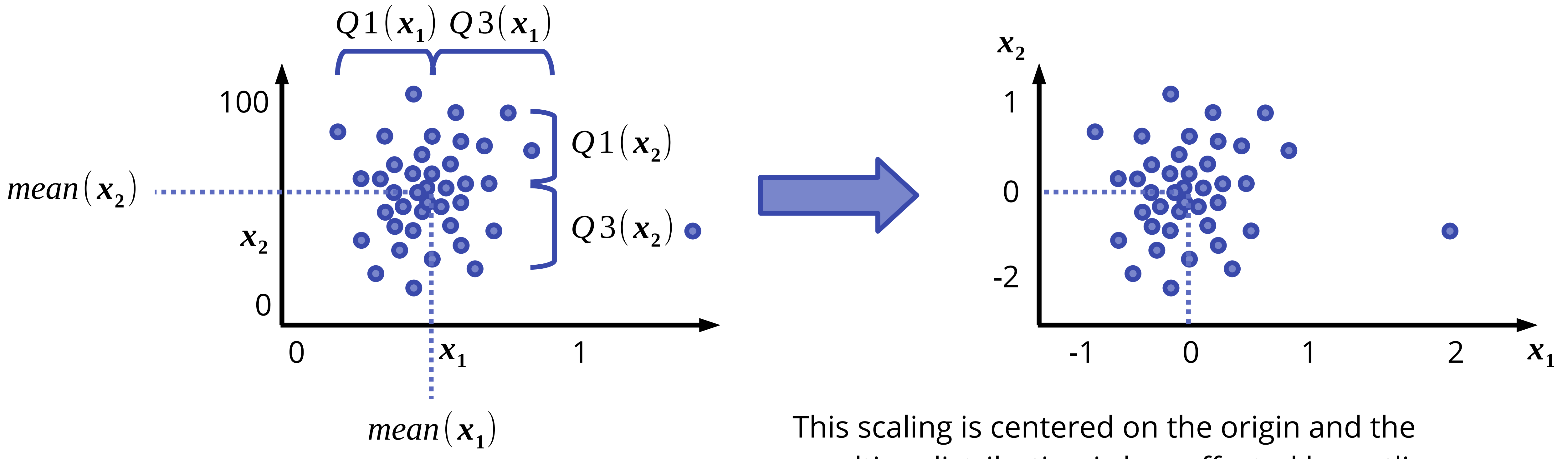
Data scaling – Robust scaler

Scale every feature onto a variable range based on the mean and the quantiles of the underlying distribution.



Data scaling – Robust scaler

Scale every feature onto a variable range based on the mean and the quantiles of the underlying distribution.



This scaling is centered on the origin and the resulting distribution is less affected by outliers

That's all folks!

Today's lecture

2 – Data and Features

Types of data

Features and feature engineering

Data scaling

Next lecture (6th Mar)

3 – Supervised Learning

Supervised learning setup

Supervised learning concepts

Benchmarking and metrics

Linear models

Nearest Neighbor models

Tree-based models