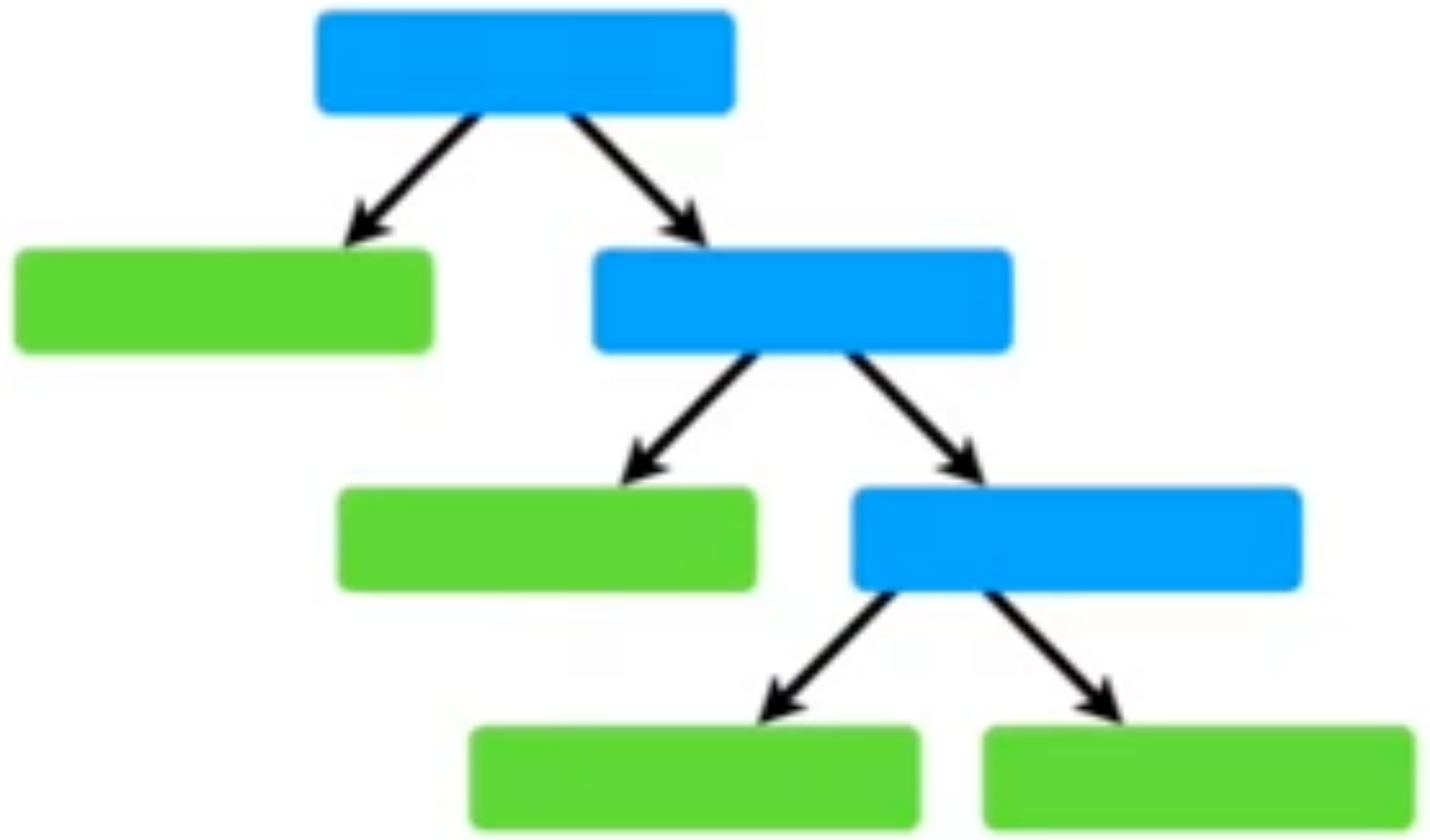
Lecture 24: Introduction to information theory

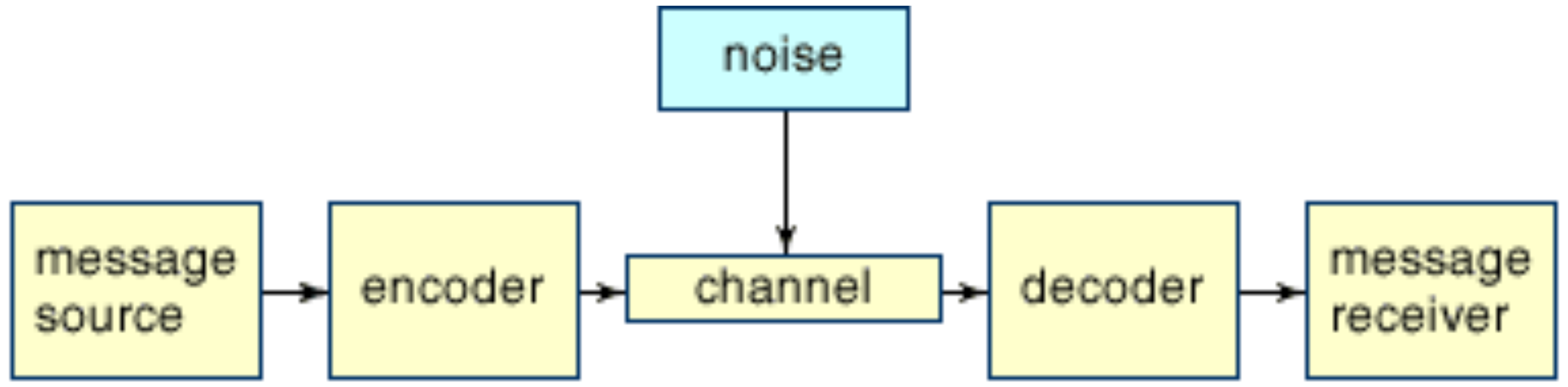
Instructor: Michael Szell
Nov 24, 2023



IT UNIVERSITY OF COPENHAGEN

Today you will learn information theory basics for ML

Entropy

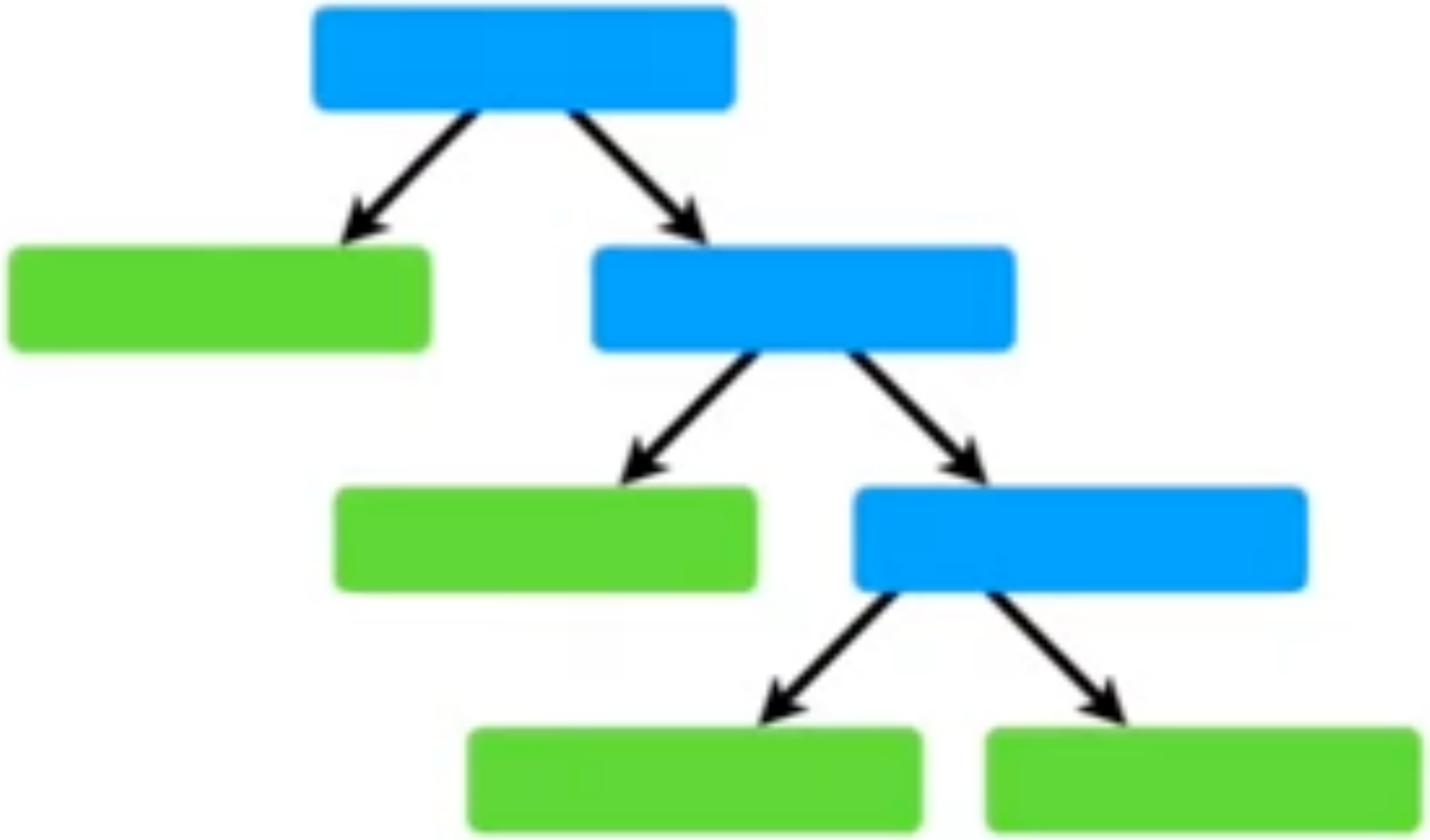


$$H(X) = \mathbb{E}(I(X))$$

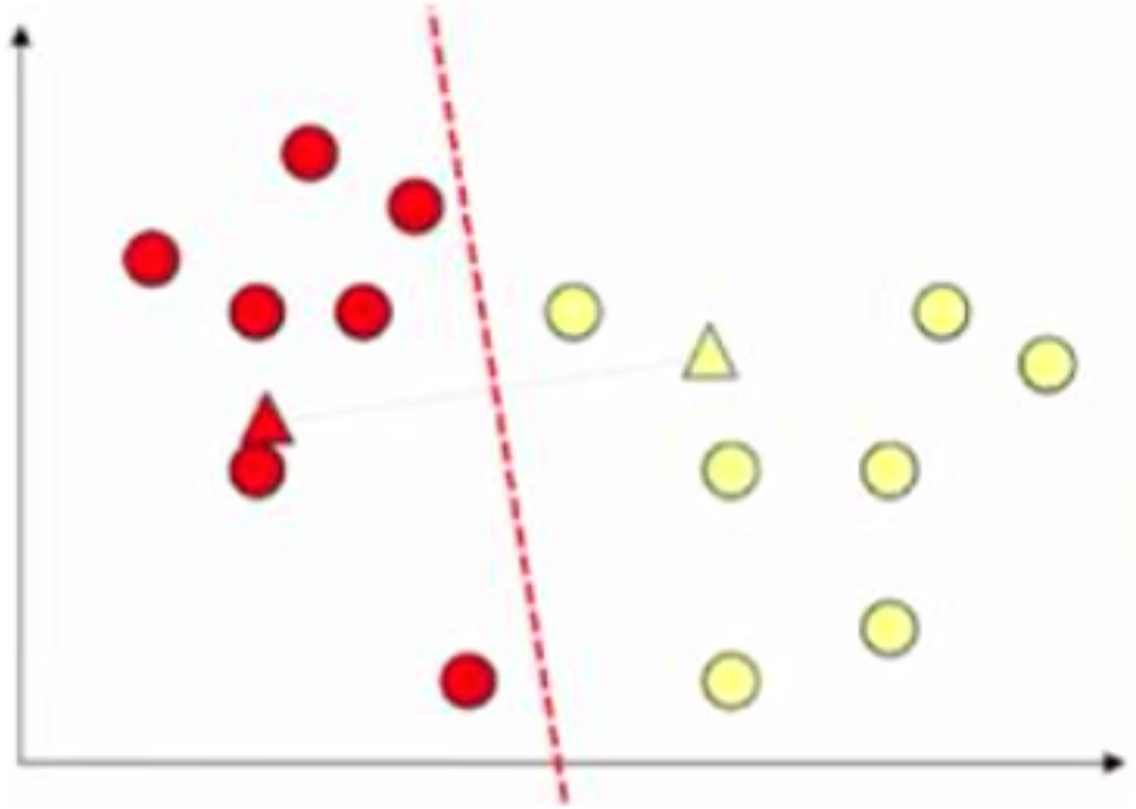
Data Science setup

git

Decision trees



Clustering



Let's play Bar Khokba

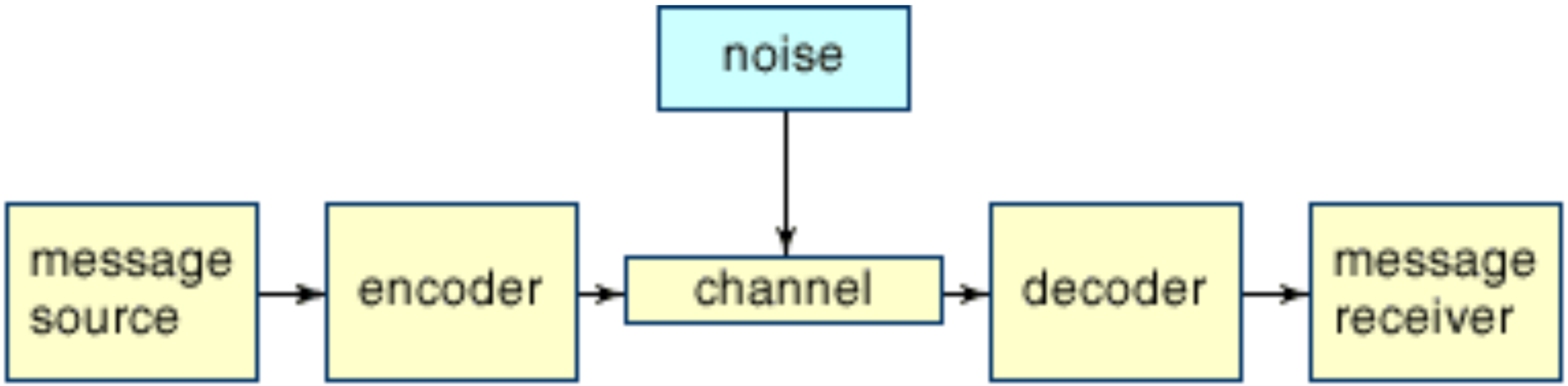
https://en.wikipedia.org/wiki/Twenty_questions

Let's play Bar Khokba

With 7 YES/NO questions, find one person in the room

Such problems can be analyzed with information theory

Information theory is the mathematical study of the quantification, storage, and communication of information.



Entropy is the basis for many concepts in data science

Mutual information

Building Decision/Classification trees

Kullback-Leibler Divergence

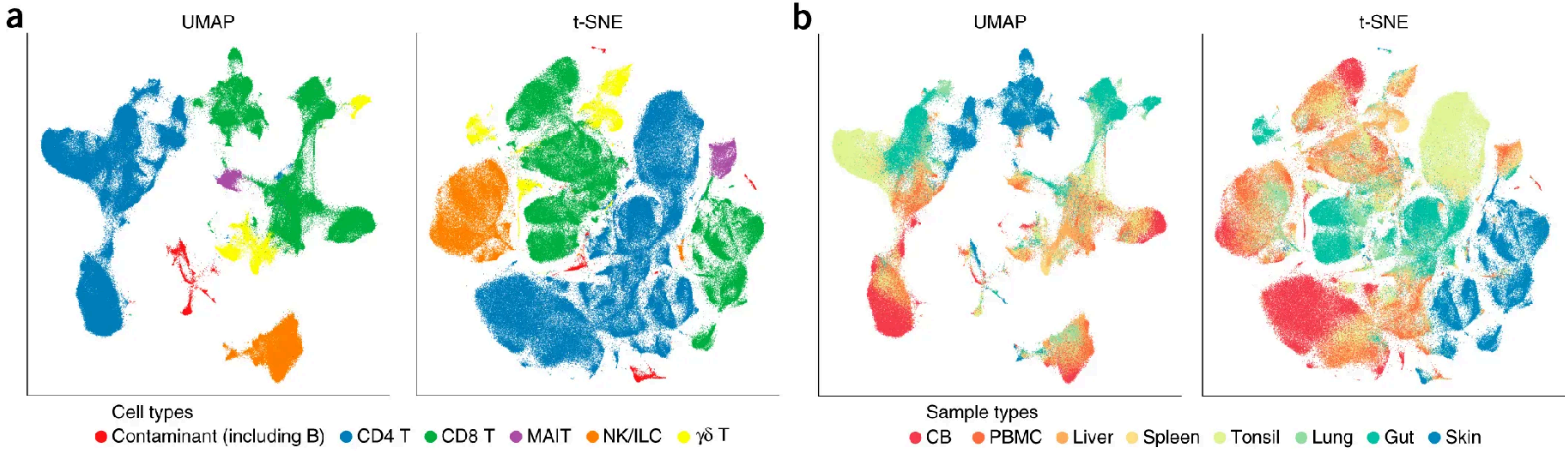
Entropy is the basis for many concepts in data science

Mutual information

Building Decision/Classification trees

Kullback-Leibler Divergence

t-SNE UMAP



Self-Information (or: information content, surprisal)

Blackboard

Self-information I of an event x is how "surprising" it is:

Self-Information

Blackboard

Self-information I of an event x is how "surprising" it is:

If $p(x) = 1$, it should be perfectly unsurprising, i.e. $I(x) = 0$

The less probable it is, the more surprising, and the higher I should be

Self-Information

Blackboard

Self-information I of an event x is how "surprising" it is:

If $p(x) = 1$, it should be perfectly unsurprising, i.e. $I(x) = 0$

The less probable it is, the more surprising, and the higher I should be

For two independent events, their total I should be the sum of their Is

Self-Information

Blackboard

Self-information I of an event x is how "surprising" it is:

If $p(x) = 1$, it should be perfectly unsurprising, i.e. $I(x) = 0$

The less probable it is, the more surprising, and the higher I should be

For two independent events, their total I should be the sum of their Is

$$I(x) = \log_2 \frac{1}{p(x)} = -\log_2 p(x)$$

Entropy is the expected surprisal

Blackboard

$$H(X) = \mathbb{E}(I(X))$$

Entropy is the expected surprisal

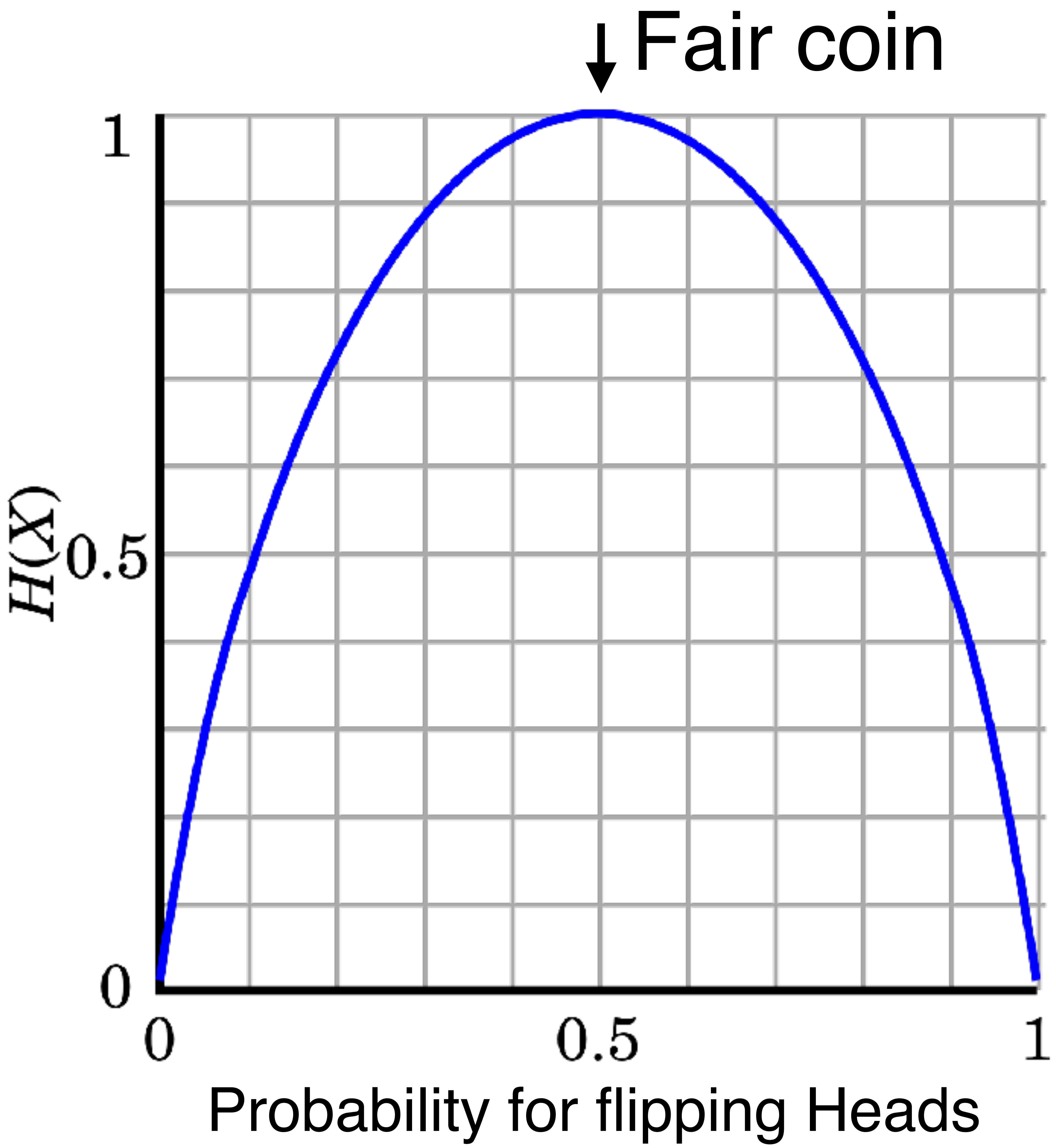
$$H(X) = \mathbb{E}(I(X))$$

$$= \sum_x p(x)\, I(x)$$

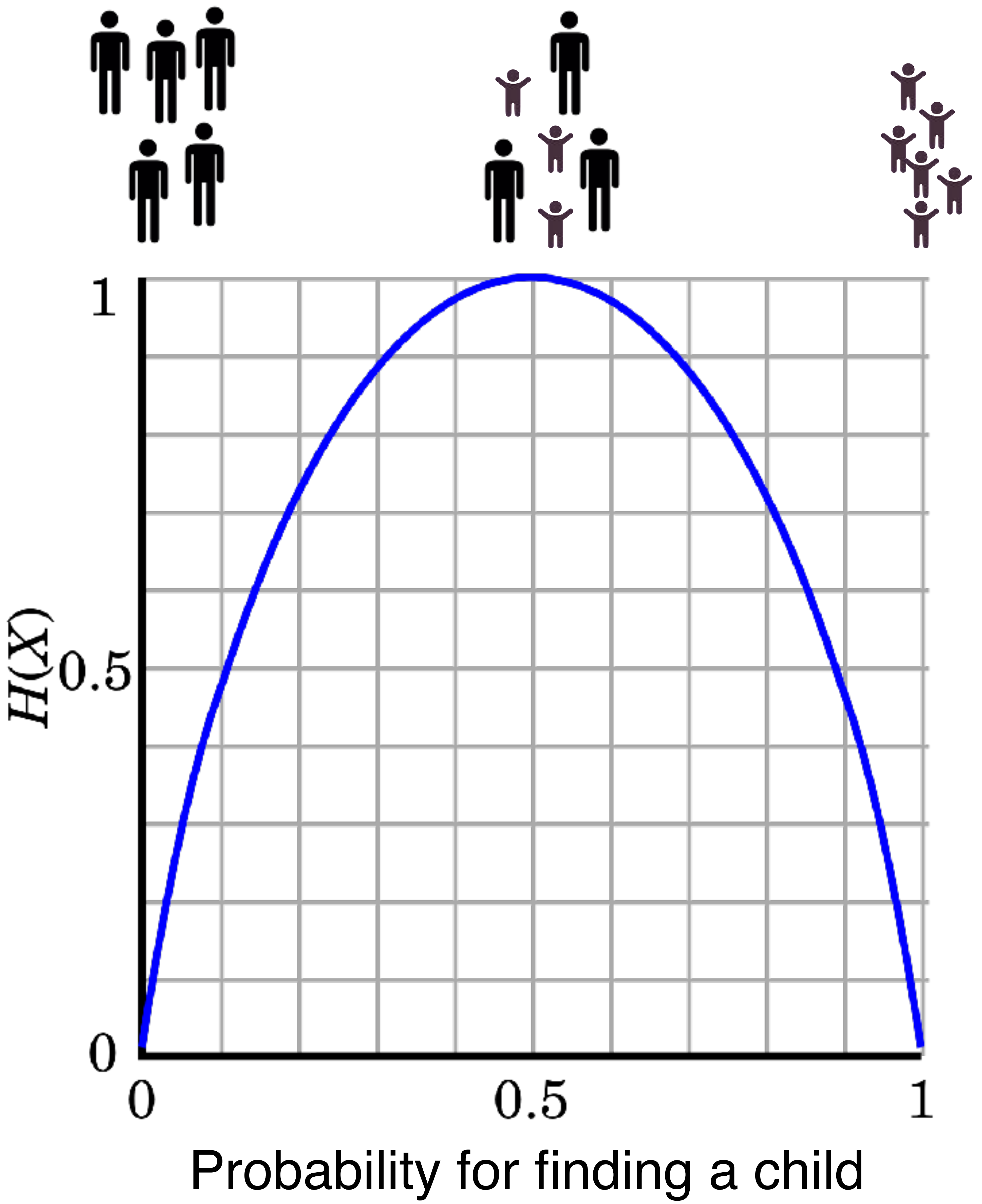
$$= -\sum_x p(x) \log_2 p(x)$$

See also: https://www.youtube.com/watch?v=YtebGVx-Fxw

Example: Entropy of a coin

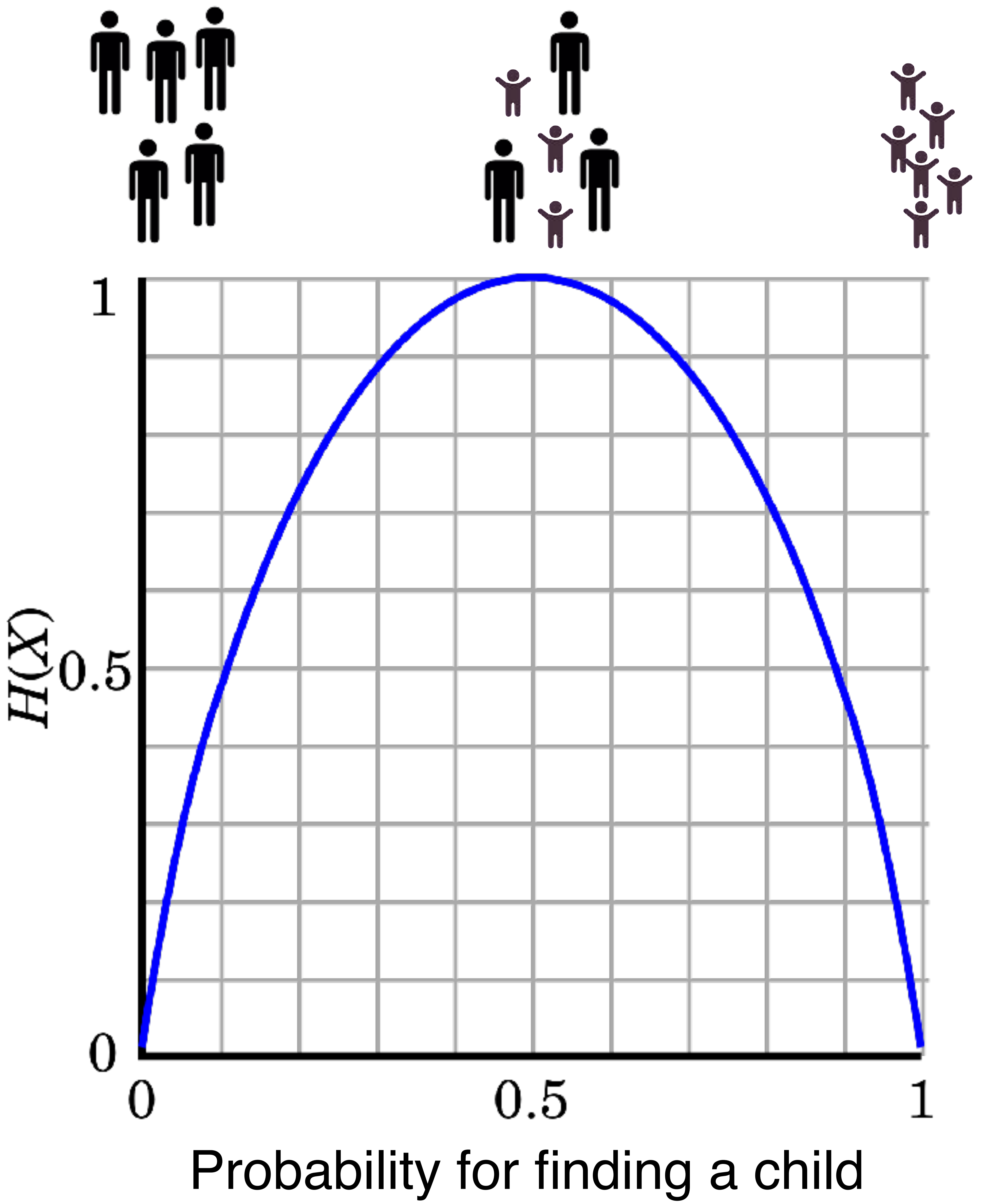


Example: Entropy of a binary data distribution



Here entropy is a measure of balance or equality

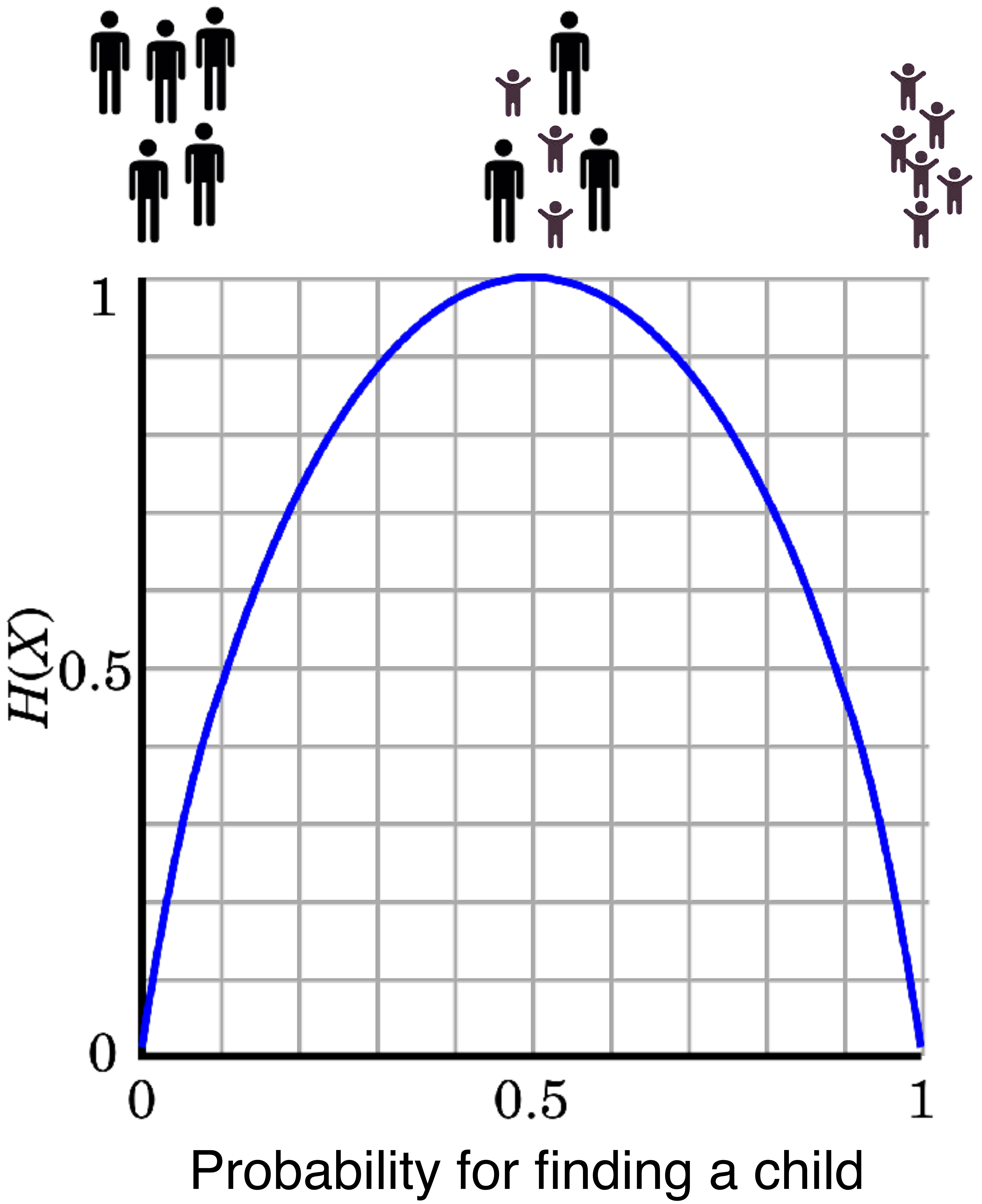
Example: Entropy of a binary data distribution



Here entropy is a measure of balance or equality

In machine learning: Impurity

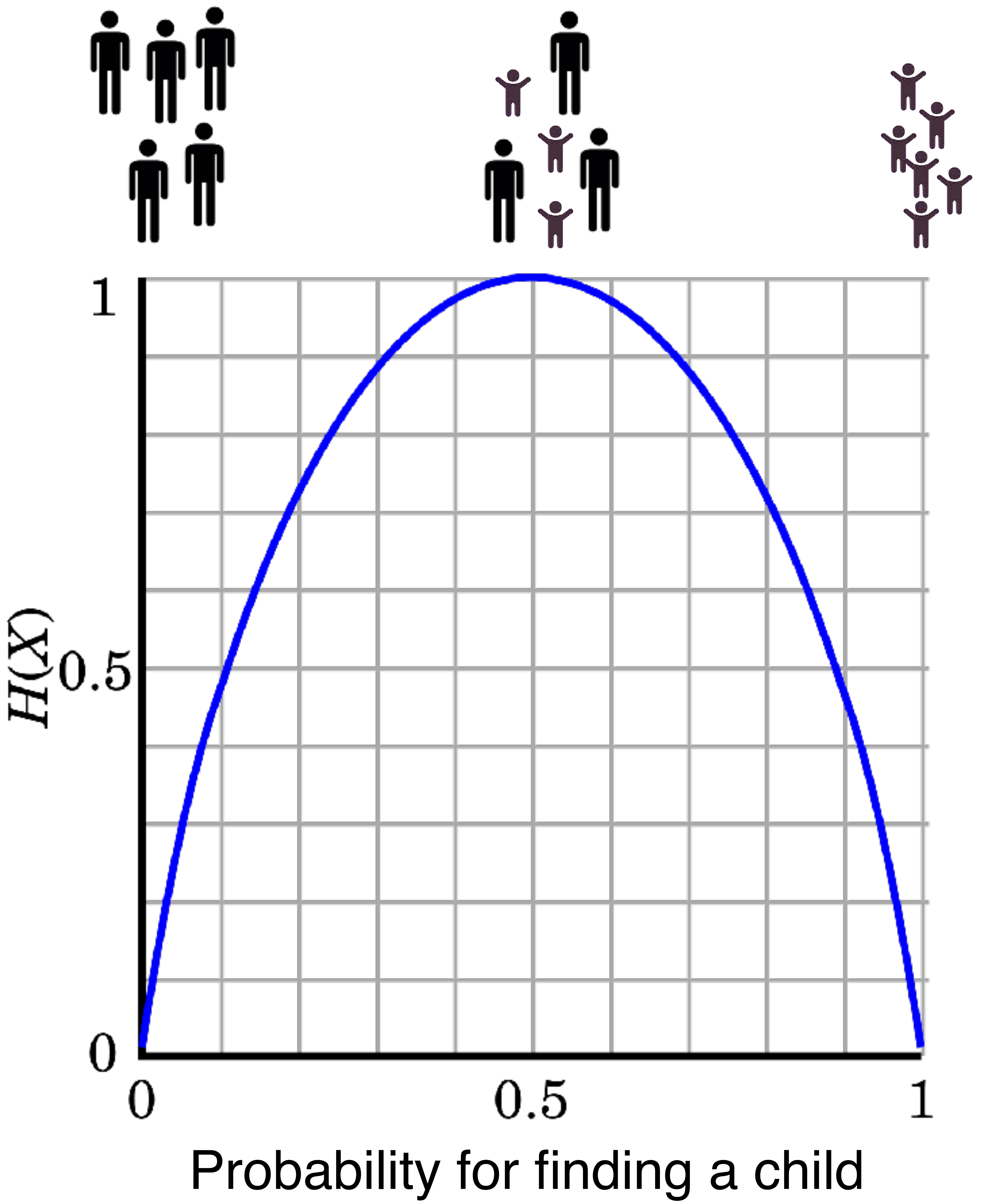
Example: Entropy of a binary data distribution



We could choose the age limit for "child" to maximize impurity..

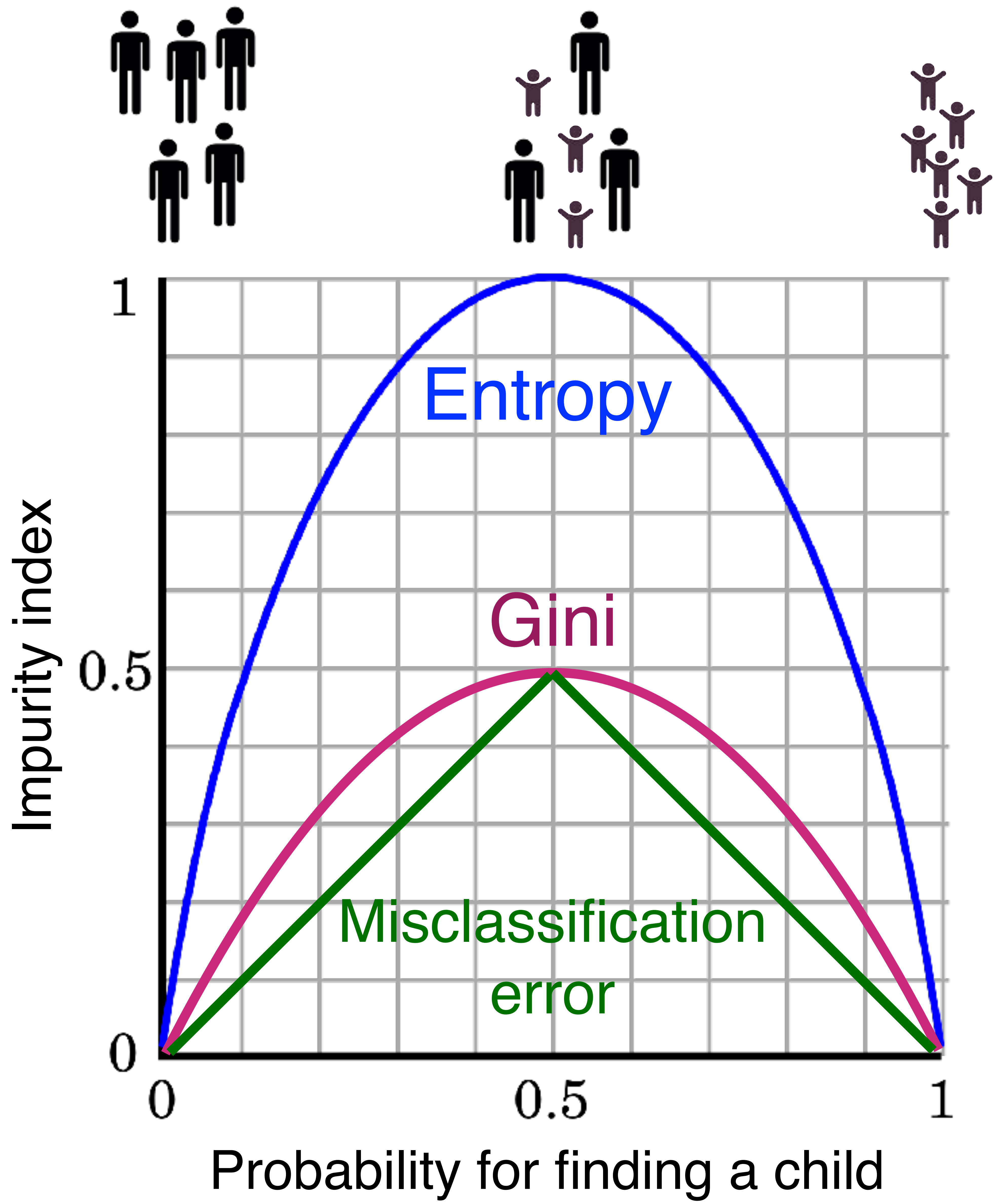
.. to give us the optimal strategy to minimize yes/no questions

Example: Maximize Entropy for optimal Bar Khokba strategy



We get an optimal strategy by choosing questions that divide the yes/no outcomes equiprobably (=maximizing entropy)

There are other impurity measures



Impurity measures are used in decision tree classification

https://www.youtube.com/watch?v=_L39rN6gz7Y

Number systems, counting in binary

Blackboard

unary

decimal

binary

hexadecimal

bits & bytes

Number systems, counting in binary

unary

decimal

binary

hexadecimal

bits & bytes

Positional systems:

We write

$$a_n a_{n-1} a_{n-2} \ldots a_0$$

to represent

$$a_n b^n + a_{n-1} b^{n-1} + a_{n-2} b^{n-2} + \cdots + a_0 b^0$$

in base b with the digits

$$a \in \{0, \ldots, b-1\}$$

Memory to store events

Blackboard

coin flip

die roll

Data Science Setup

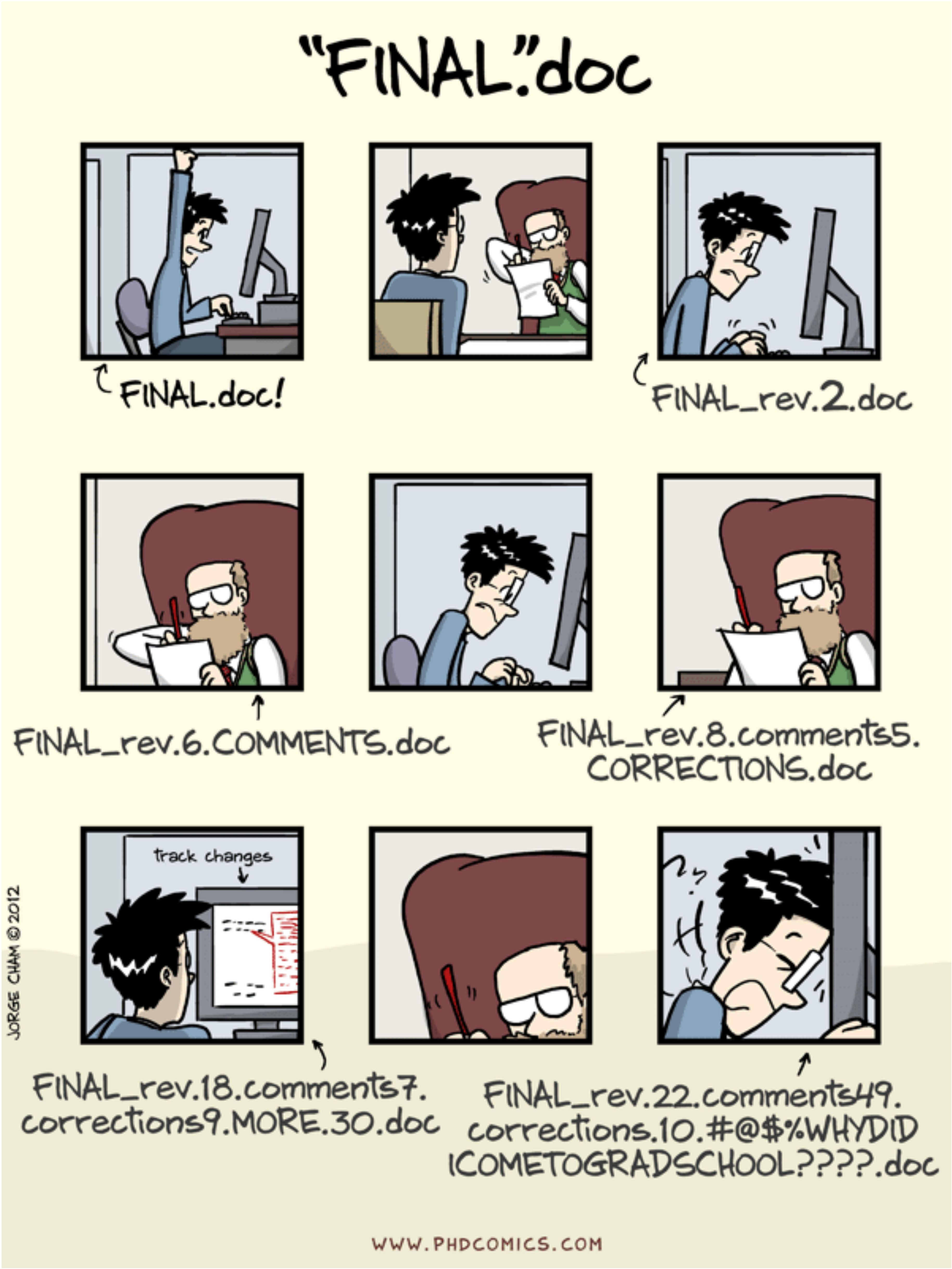
Practical tips

Get acquainted with git / GitHub

Git is a distributed version-control system for tracking changes in any set of files, originally designed to coordinate group work.

git

Similar systems: CSV, SVN, Mercurial



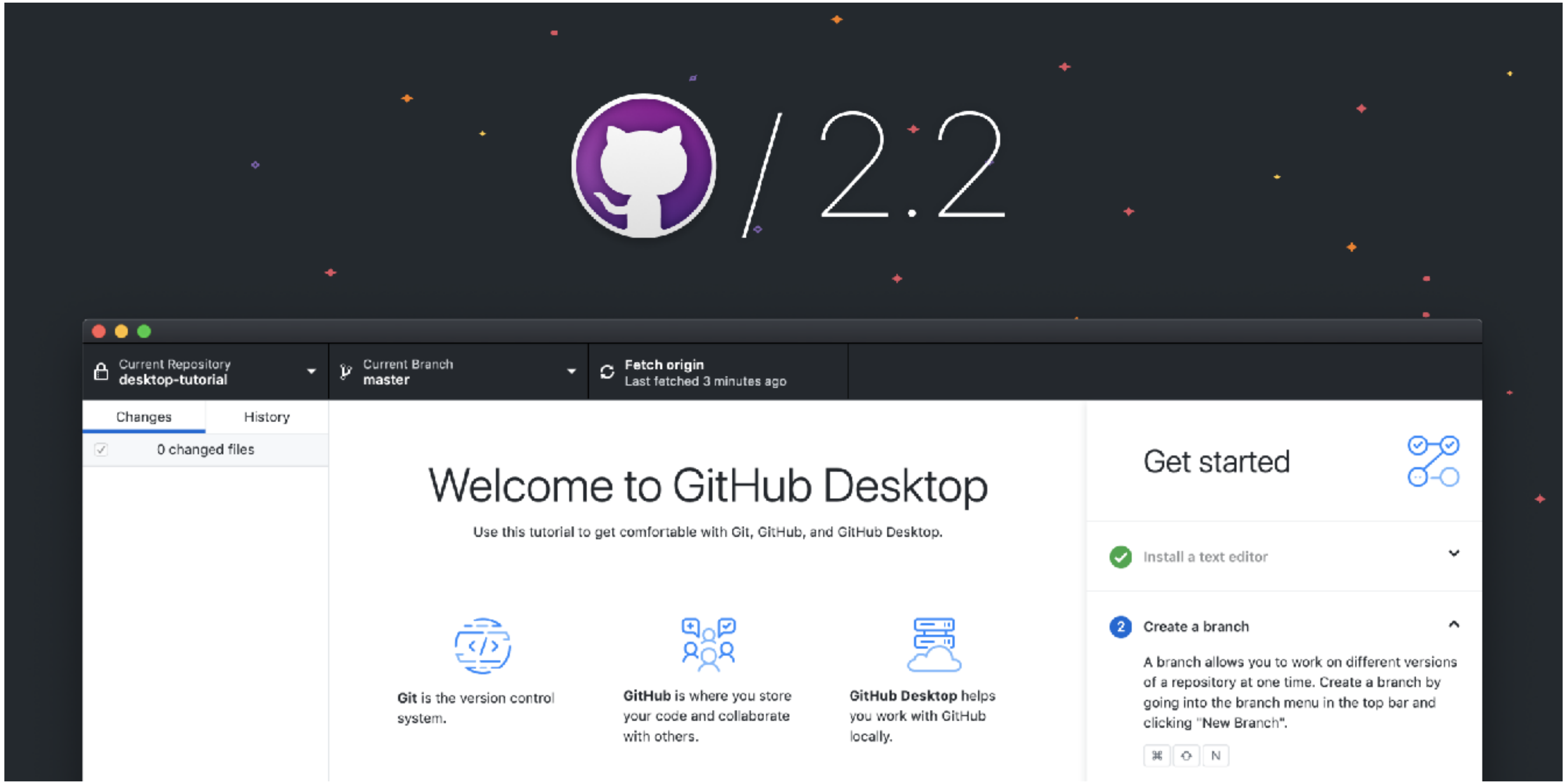
Get acquainted with git / GitHub

GitHub, Inc. is a subsidiary of Microsoft which provides hosting for software development and version control using Git

GitLab or CodeBerg are alternatives

Get acquainted with git / GitHub

GitHub Desktop is a Graphical User Interface (GUI) for Github



desktop.github.com

Before we delve into a project, you MUST get organized!

```
├── LICENSE
├── Makefile           <- Makefile with commands like `make data` or `make train`
├── README.md          <- The top-level README for developers using this project.
├── data
│   ├── external       <- Data from third party sources.
│   ├── interim        <- Intermediate data that has been transformed.
│   ├── processed      <- The final, canonical data sets for modeling.
│   └── raw            <- The original, immutable data dump.
│
├── docs               <- A default Sphinx project; see sphinx-doc.org for details
│
├── models             <- Trained and serialized models, model predictions, or model summaries
│
├── notebooks          <- Jupyter notebooks. Naming convention is a number (for ordering),
│                         the creator's initials, and a short `-` delimited description, e.g.
│                         `1.0-jqp-initial-data-exploration`.
│
├── references         <- Data dictionaries, manuals, and all other explanatory materials.
│
├── reports            <- Generated analysis as HTML, PDF, LaTeX, etc.
│   └── figures        <- Generated graphics and figures to be used in reporting
│
├── requirements.txt   <- The requirements file for reproducing the analysis environment, e.g.
│                         generated with `pip freeze > requirements.txt`
│
├── src                <- Source code for use in this project.
│   ├── __init__.py    <- Makes src a Python module
│   │
│   ├── data           <- Scripts to download or generate data
│   │   └── make_dataset.py
│   │
│   ├── features       <- Scripts to turn raw data into features for modeling
│   │   └── build_features.py
│   │
│   ├── models         <- Scripts to train models and then use trained models to make
│   │   │                 predictions
│   │   ├── predict_model.py
│   │   └── train_model.py
│   │
│   └── visualization  <- Scripts to create exploratory and results oriented visualizations
│       └── visualize.py
│
└── tox.ini            <- tox file with settings for running tox; see tox.readthedocs.io
```

Folder structure is not a science, but based on years of experience

Cookiecutter Data Science

https://github.com/drivendata/cookiecutter-data-science

Before we delve into a project, you MUST get organized!

```
├── LICENSE
├── Makefile           <- Makefile with commands like `make data` or `make train`
├── README.md          <- The top-level README for developers using this project.
├── data
│   ├── external       <- Data from third party sources.
│   ├── interim        <- Intermediate data that has been transformed.
│   ├── processed      <- The final, canonical data sets for modeling.
│   └── raw            <- The original, immutable data dump.
│
├── docs               <- A default Sphinx project; see sphinx-doc.org for details
│
├── models             <- Trained and serialized models, model predictions, or model summaries
│
├── notebooks          <- Jupyter notebooks. Naming convention is a number (for ordering),
│                         the creator's initials, and a short `-` delimited description, e.g.
│                         `1.0-jqp-initial-data-exploration`.
│
├── references         <- Data dictionaries, manuals, and all other explanatory materials.
│
├── reports            <- Generated analysis as HTML, PDF, LaTeX, etc.
│   └── figures        <- Generated graphics and figures to be used in reporting
│
├── requirements.txt   <- The requirements file for reproducing the analysis environment, e.g.
│                         generated with `pip freeze > requirements.txt`
│
├── src                <- Source code for use in this project.
│   ├── __init__.py    <- Makes src a Python module
│   │
│   ├── data           <- Scripts to download or generate data
│   │   └── make_dataset.py
│   │
│   ├── features       <- Scripts to turn raw data into features for modeling
│   │   └── build_features.py
│   │
│   ├── models         <- Scripts to train models and then use trained models to make
│   │   │                 predictions
│   │   ├── predict_model.py
│   │   └── train_model.py
│   │
│   └── visualization  <- Scripts to create exploratory and results oriented visualizations
│       └── visualize.py
│
└── tox.ini            <- tox file with settings for running tox; see tox.readthedocs.io
```

MUST haves:

- README.md
- Data: raw, interim, processed

- Notebooks / Code
- References
- Reports / Figures

Cookiecutter Data Science

https://github.com/drivendata/cookiecutter-data-science

Before we delve into a project, you MUST get organized!

How to Create a Professional Github Data Science Repository

Good practices in repository structure, documenting jupyter notebooks, and writing an informative README

Ahilan Srivishnumohan Jul 25, 2020 · 7 min read ★

ericmjl / ds-project-organization.md

Created 3 years ago • Report abuse

Star 82 | Fork 21

Code | Revisions 1 | Stars 82 | Forks 21 | Download ZIP

How to organize your Python data science project

ds-project-organization.md | Raw

How to organize your Python data science project

https://gist.github.com/ericmjl/27e50331f24db3e8f957d1fe7bbbe510

https://towardsdatascience.com/how-to-create-a-professional-github-data-science-repository-84e9607644a2

Data is usually NOT stored on Github

Github is for code and work documents, not data.

There are hard limitations: 100 MB files, 2 GB push, ~5 GB repo

If you have hundreds of MB data, add the data folder to .gitignore

https://stackoverflow.com/questions/38768454/repository-size-limits-for-github-com

Data is usually NOT stored on Github

Github is for code and work documents, not data.

There are hard limitations: 100 MB files, 2 GB push, ~5 GB repo

If you have hundreds of MB data, add the data folder to .gitignore

To make projects reproducible,
we store data here, for example:

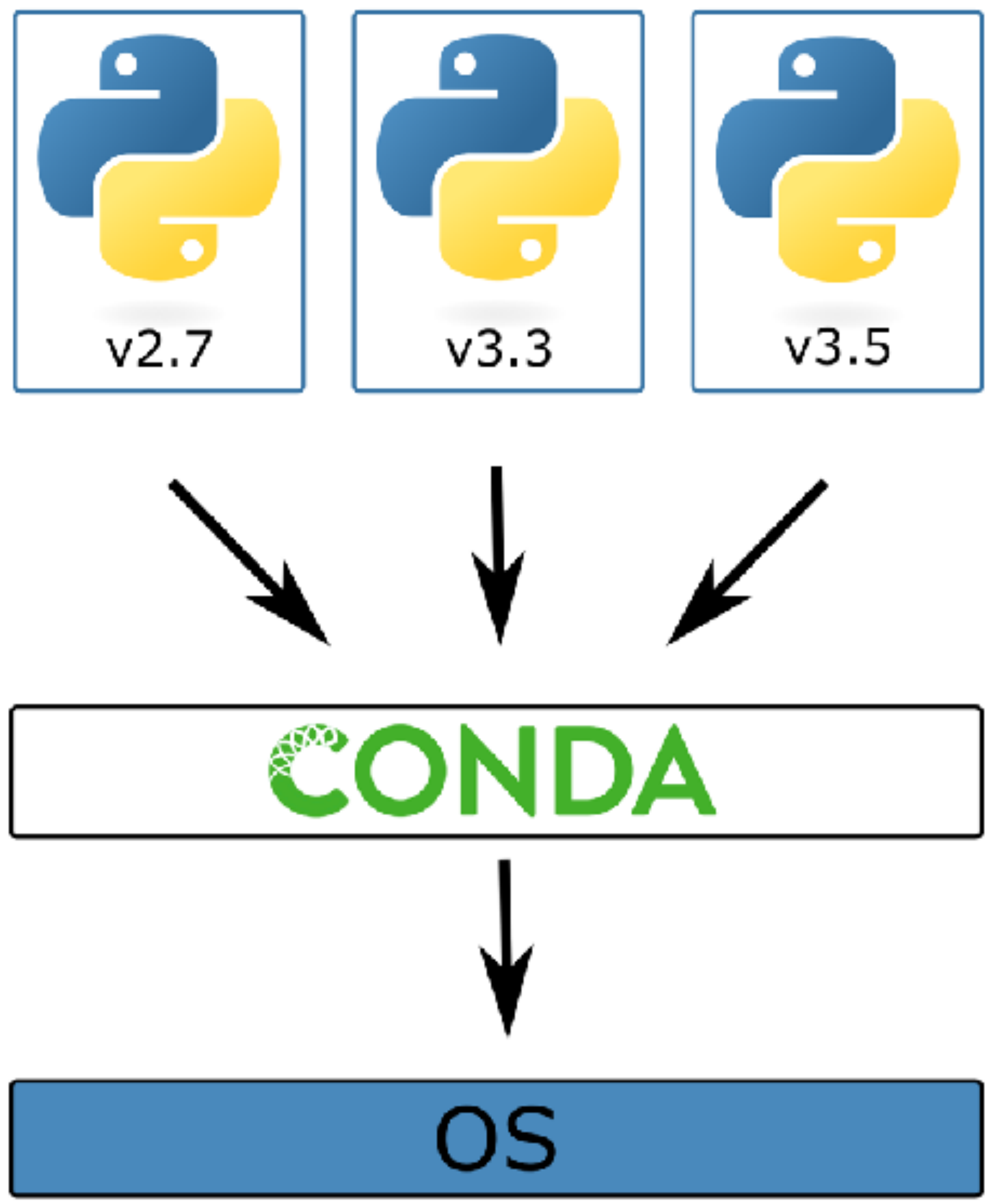
zenodo

https://stackoverflow.com/questions/38768454/repository-size-limits-for-github-com

Advanced organization

Often we use a virtual environment (venv) for each project

Why?



https://deparkes.co.uk/2016/10/21/anaconda-python-environments/

Advanced organization

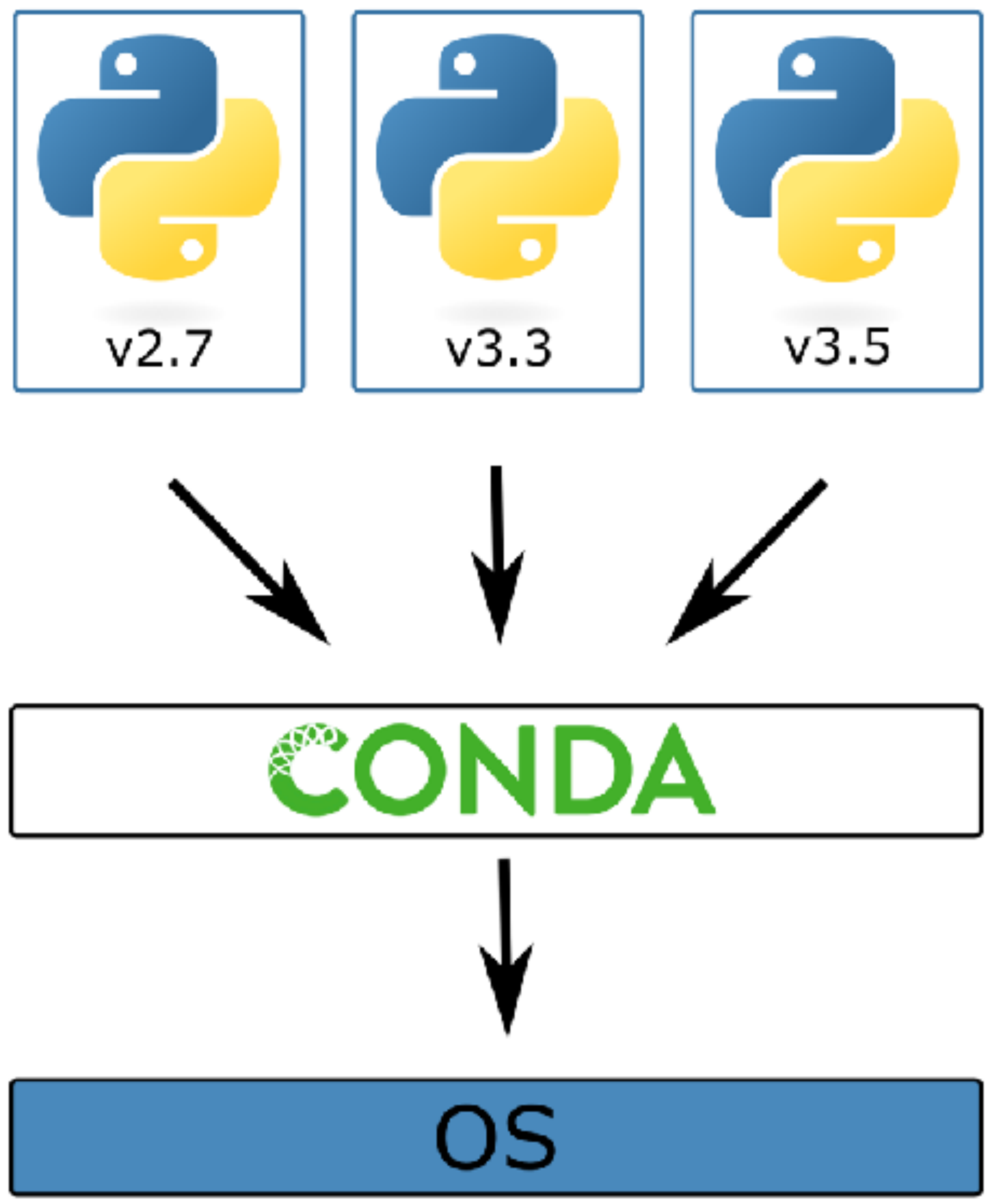
Often we use a virtual environment (venv) for each project

Why?

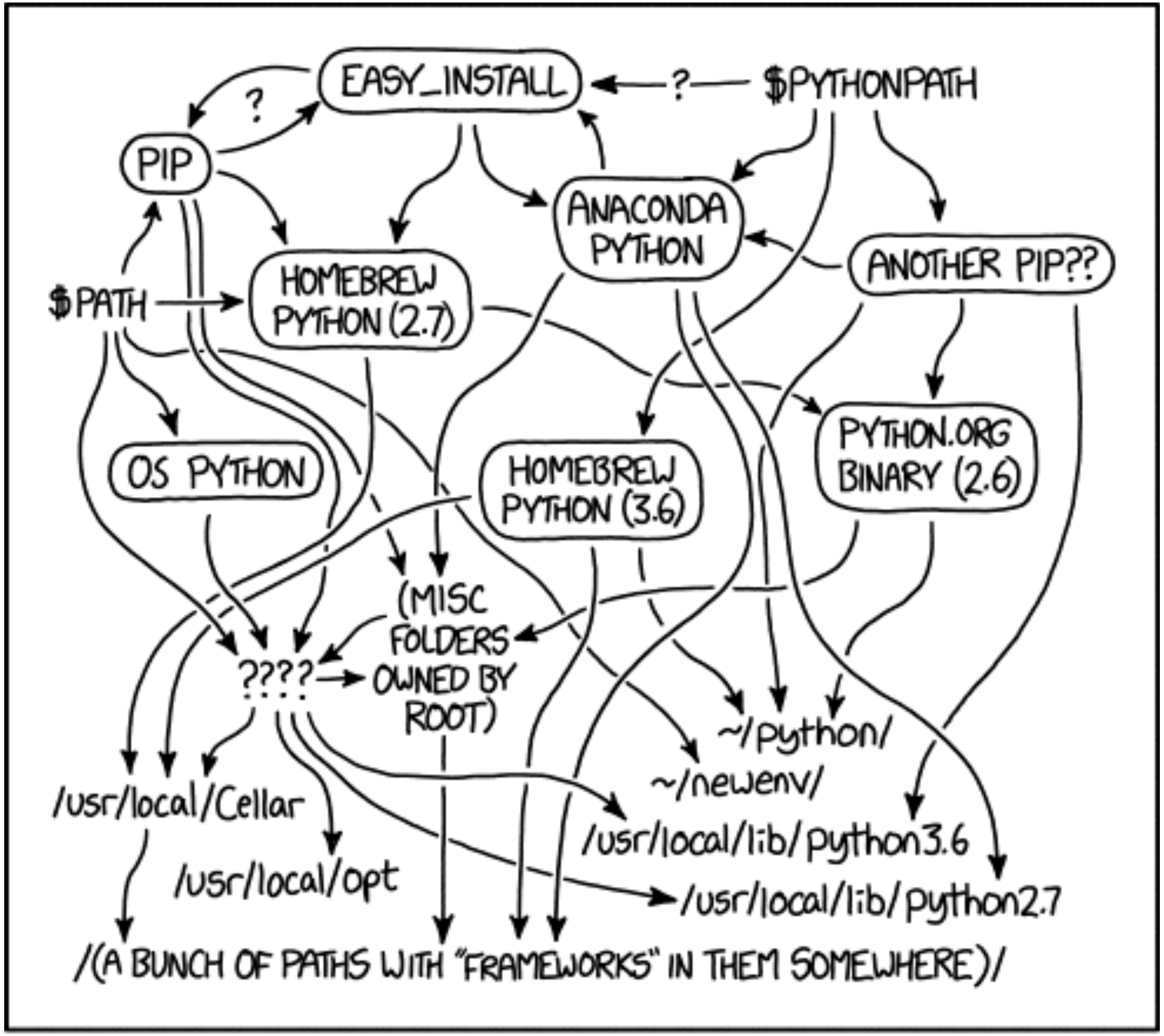
1) Reproducibility:

- Python changes all the time
- Python packages change all the time
- requirements.txt differ between projects

2) To not mess up your Python install



https://deparkes.co.uk/2016/10/21/anaconda-python-environments/



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

Advanced organization

requirements.txt

```
matplotlib>=3.3.3
numpy>=1.19.4
pandas>=1.0.3
pyproj>=2.6.1.post1
geojson>=2.5.0
shapely>=1.7.0
csv>=1.0
networkx>=2.5
igraph>=0.8.3
fiona>=1.8.18
osmnx==0.16.2
geopandas>=0.8.1
tqdm>=4.55.0
haversine>=2.3.0
```

```
pip freeze > requirements.txt
```

Advanced organization

LICENSE

If you publish your project, the license is the legal document telling others how they can use it.

No serious company will ever risk building on your code if you don't have a license.

§

Advanced organization

LICENSE

If you publish your project, the license is the legal document telling others how they can use it.

Choose an established license, don't write it yourself.

Consider copyleft and creative commons to protect freedoms.

https://creativecommons.org/

https://www.gnu.org/philosophy/free-sw.en.html

https://en.wikipedia.org/wiki/JSLint#License

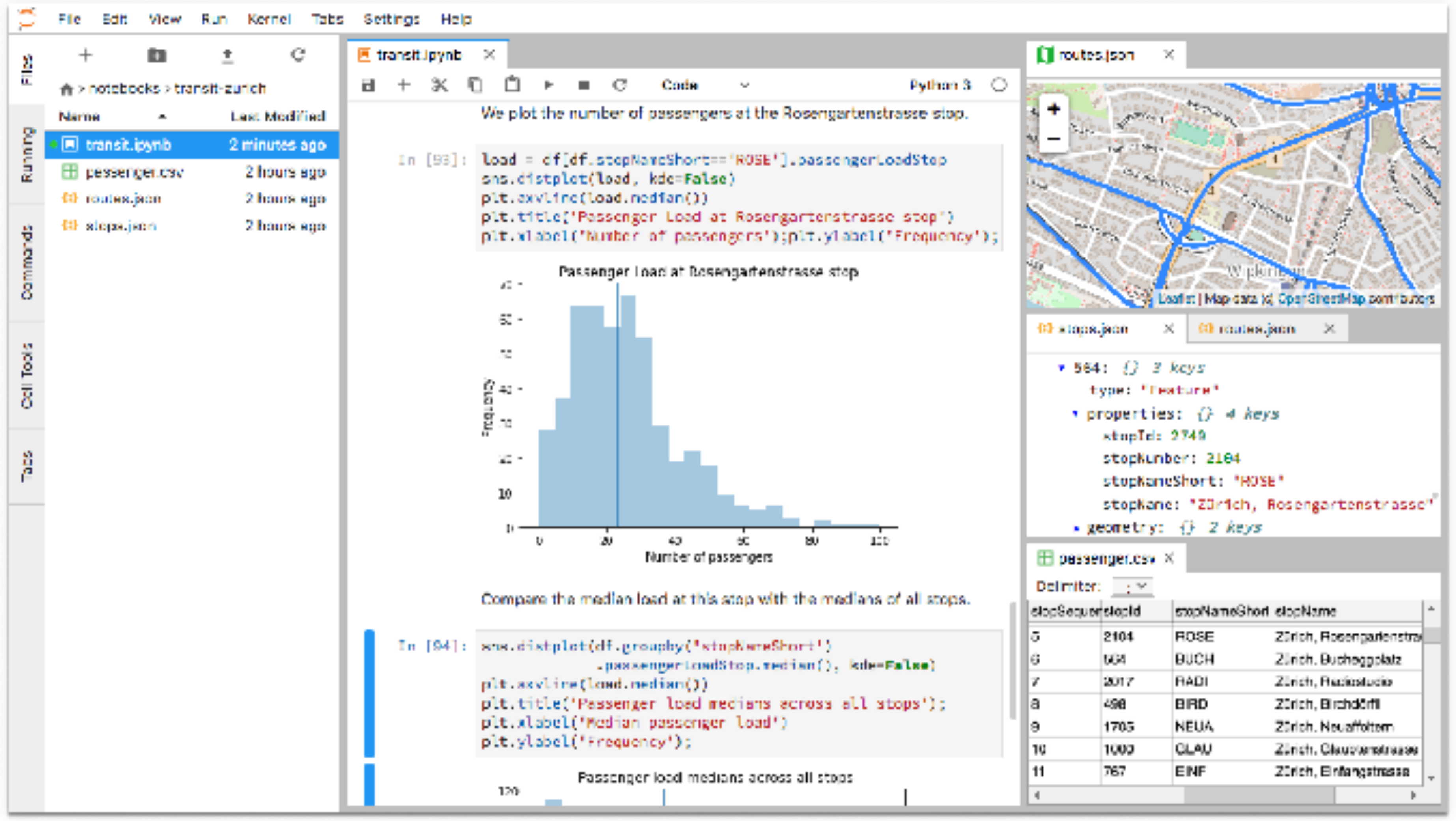
Advanced organization

DOCS

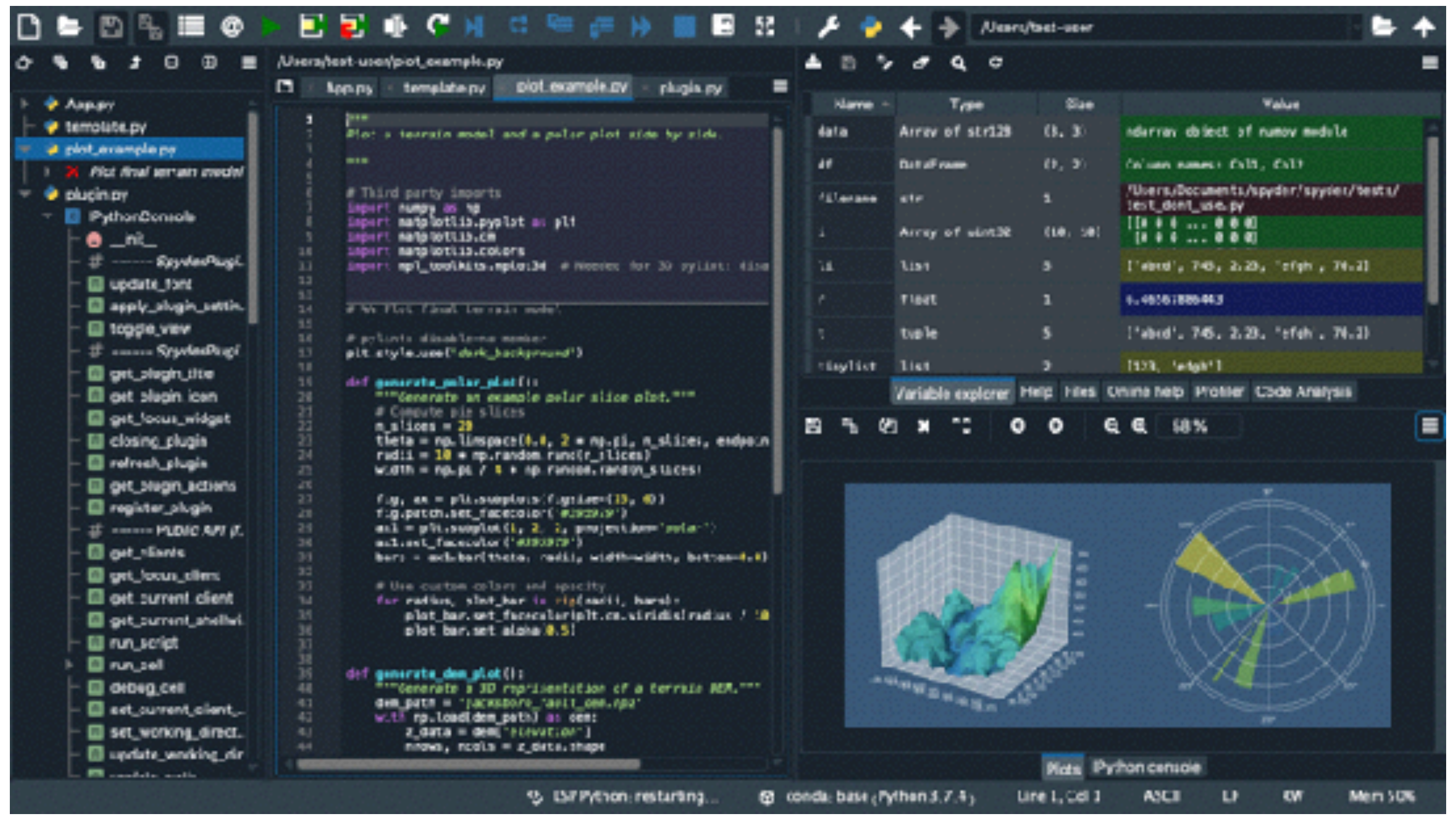
If you want others to use and build on your code, you need excellent documentation. It makes or breaks its success.

Expect investing >20% of the whole project time on docs.

Advanced organization: IDEs (integrated development environments)

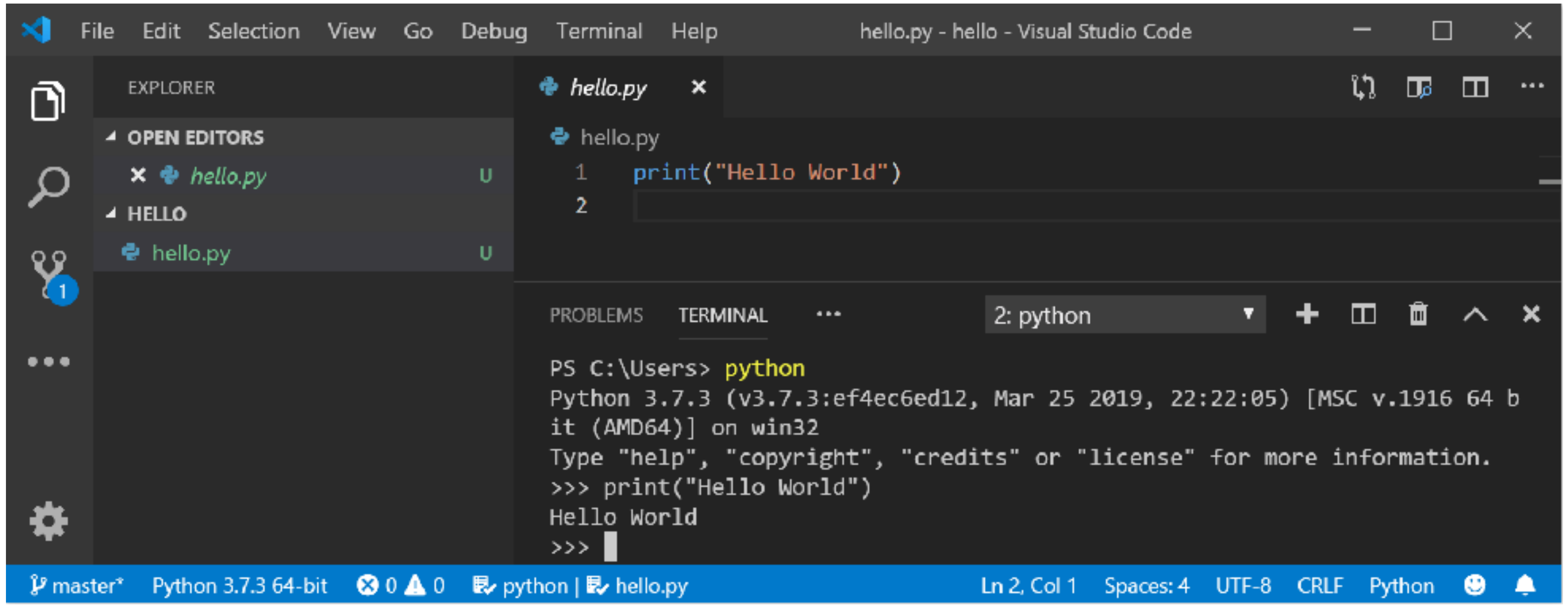


jupyter Lab



SPYDER

Visual Studio Code



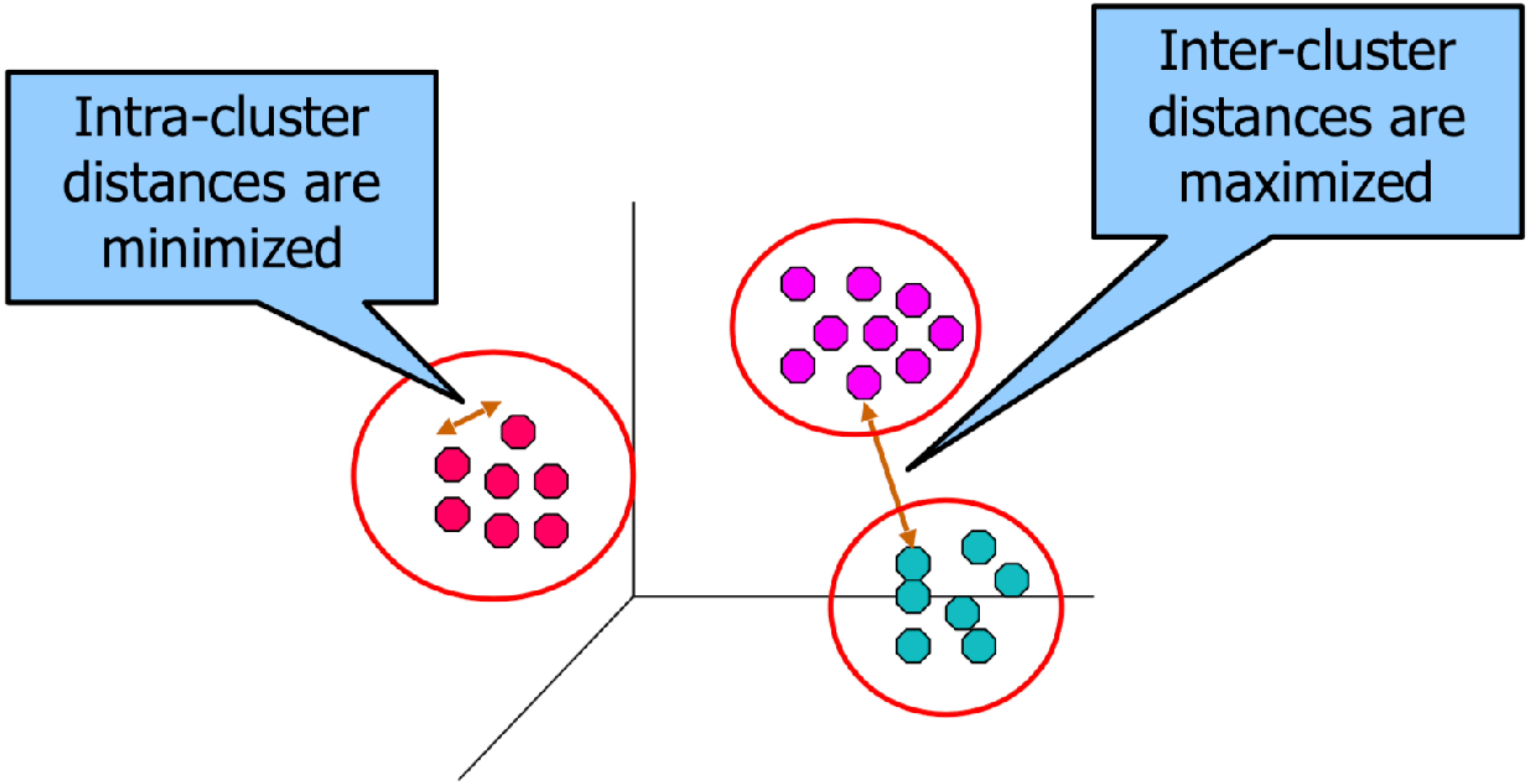
Clustering

Clustering Example

Image segmentation

Goal: Break up the image into meaningful or perceptually similar regions

Clustering means grouping similar objects



Clustering = Distance + Method

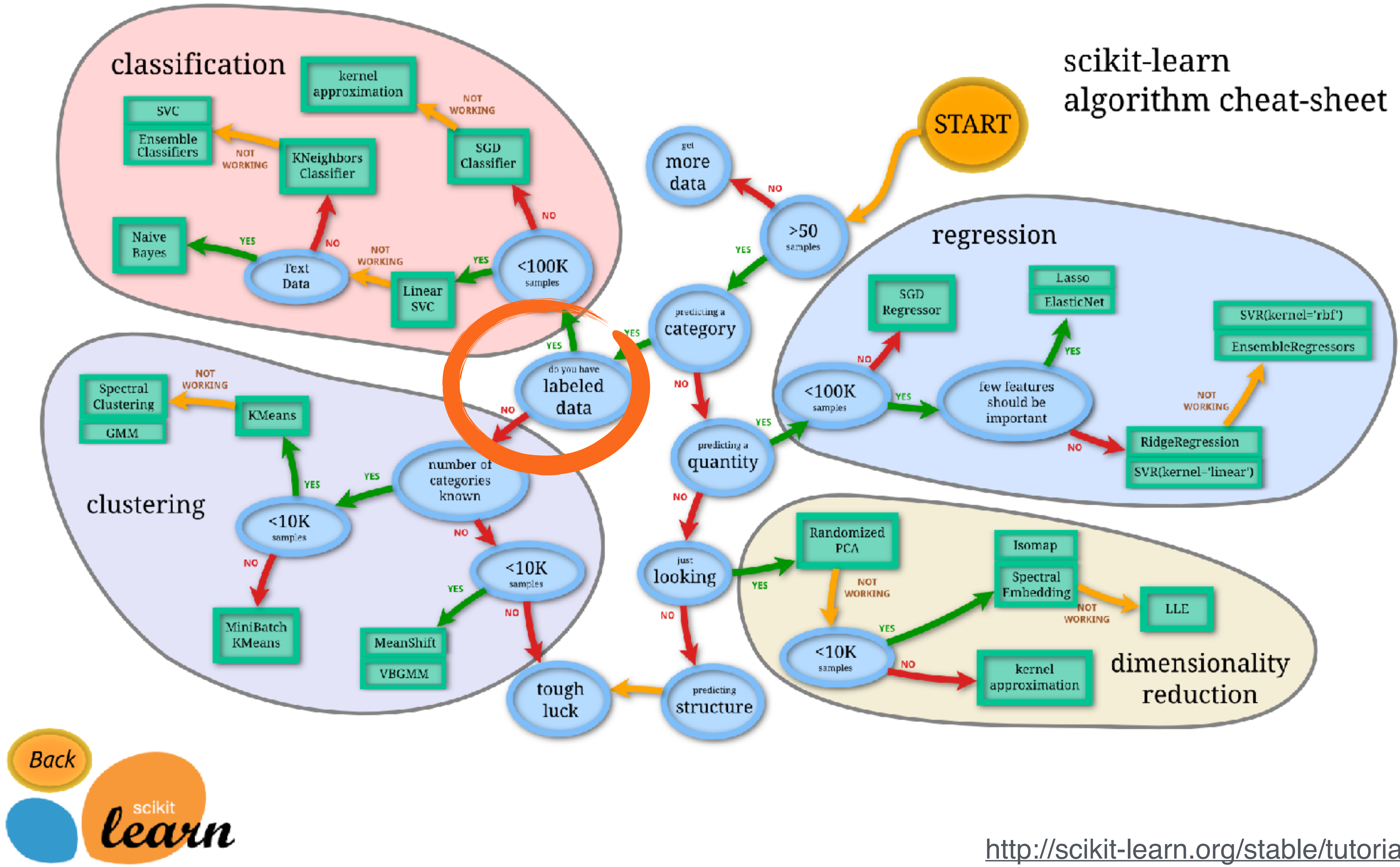
Clustering vs. Classification

Clustering vs. Classification

unsupervised learning technique that groups data points based on their similarities

supervised learning technique that assigns data points to predefined categories

labeled data: Classification   no labeled data: Clustering

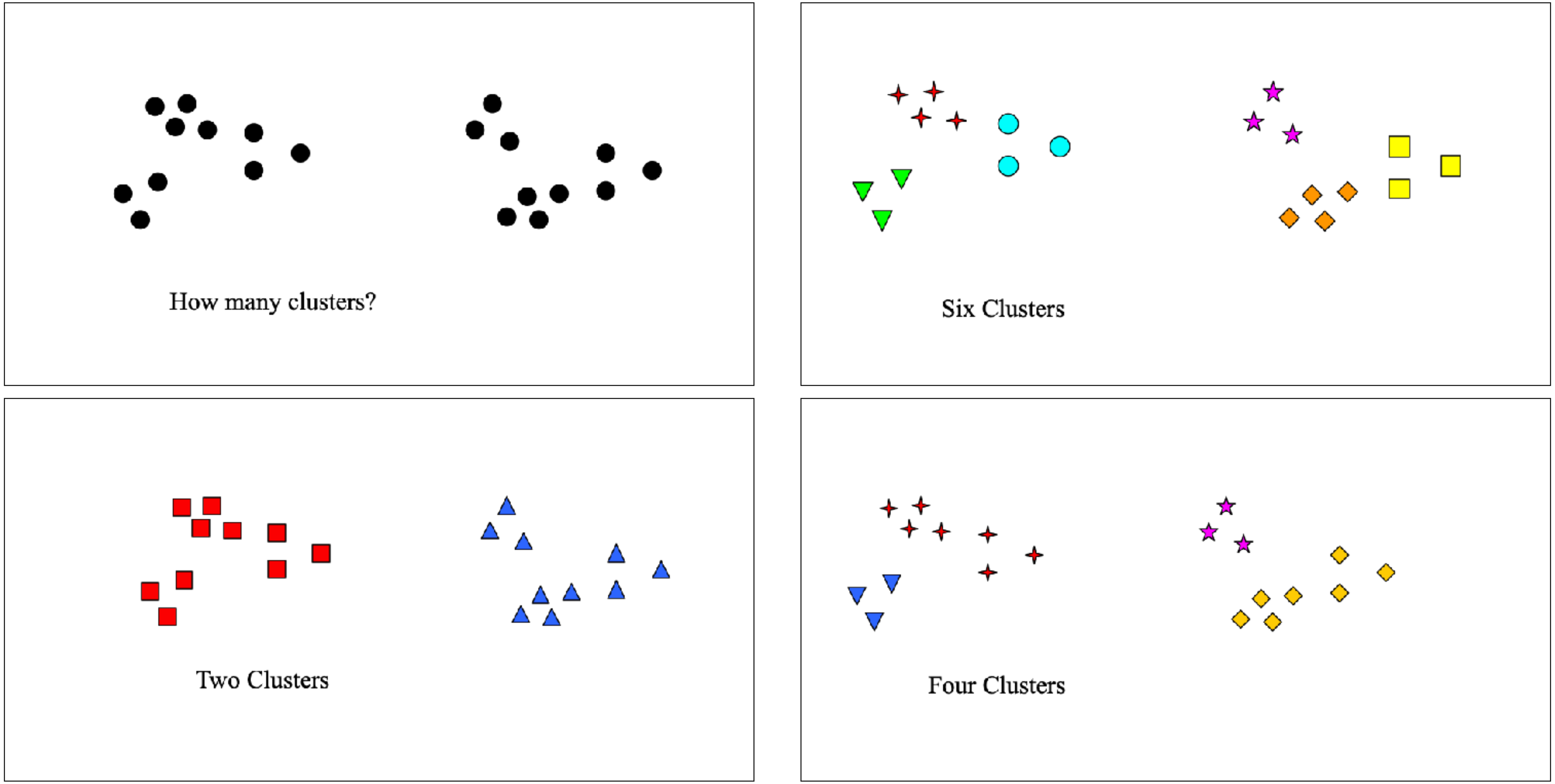


http://scikit-learn.org/stable/tutorial/machine_learning_map/

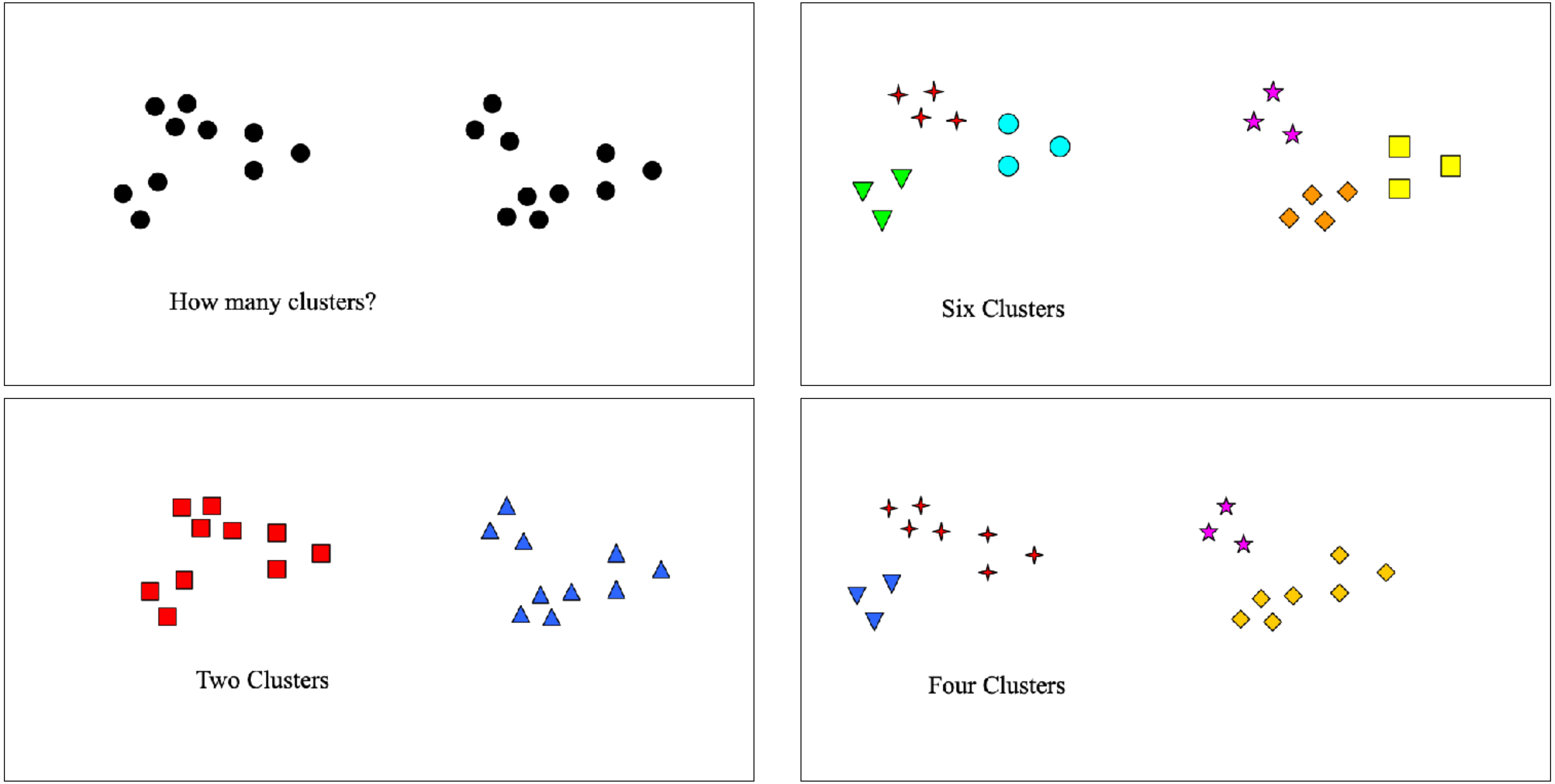
Clustering is Ambiguous

How many clusters?

Clustering is Ambiguous



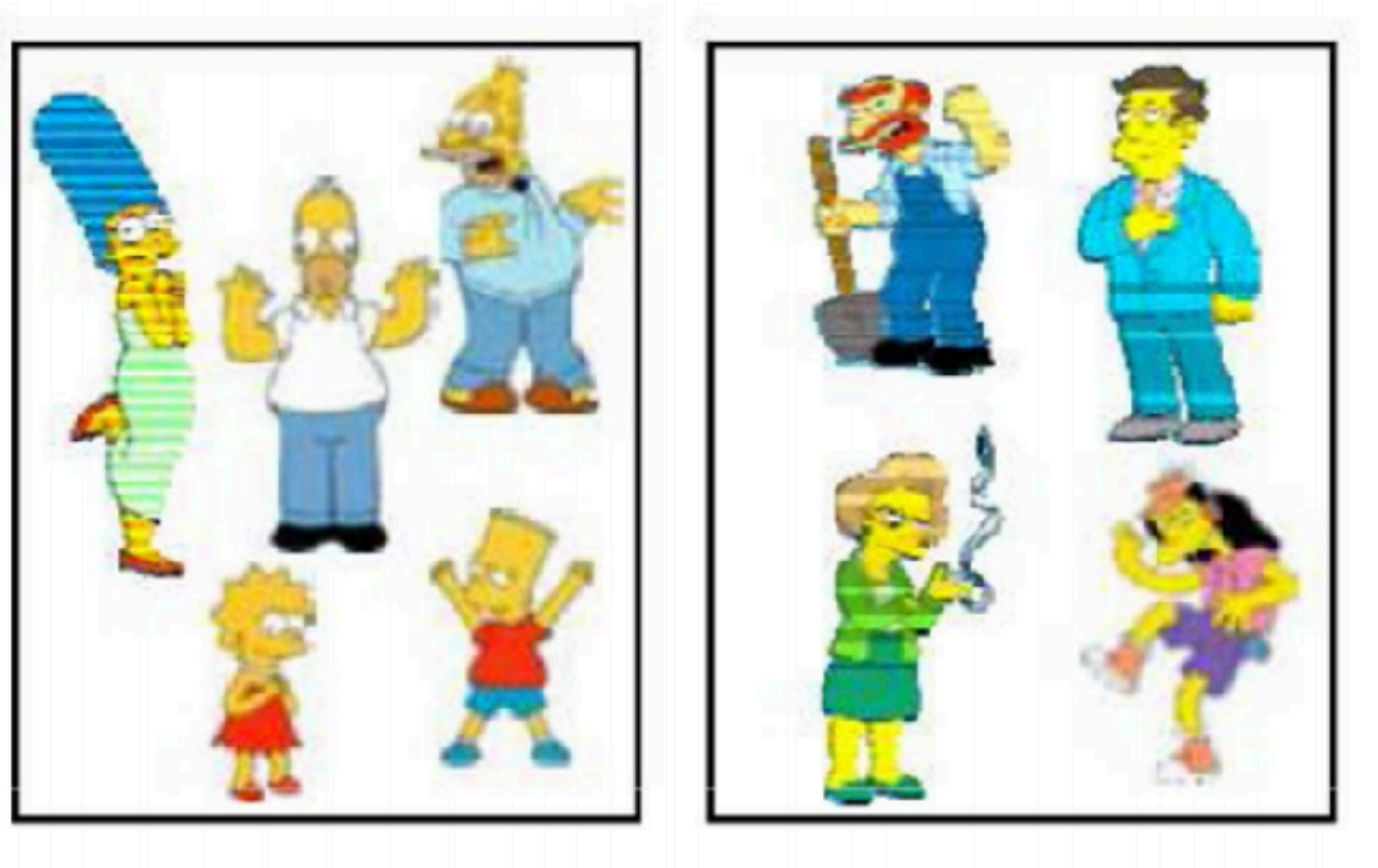
Clustering is Ambiguous



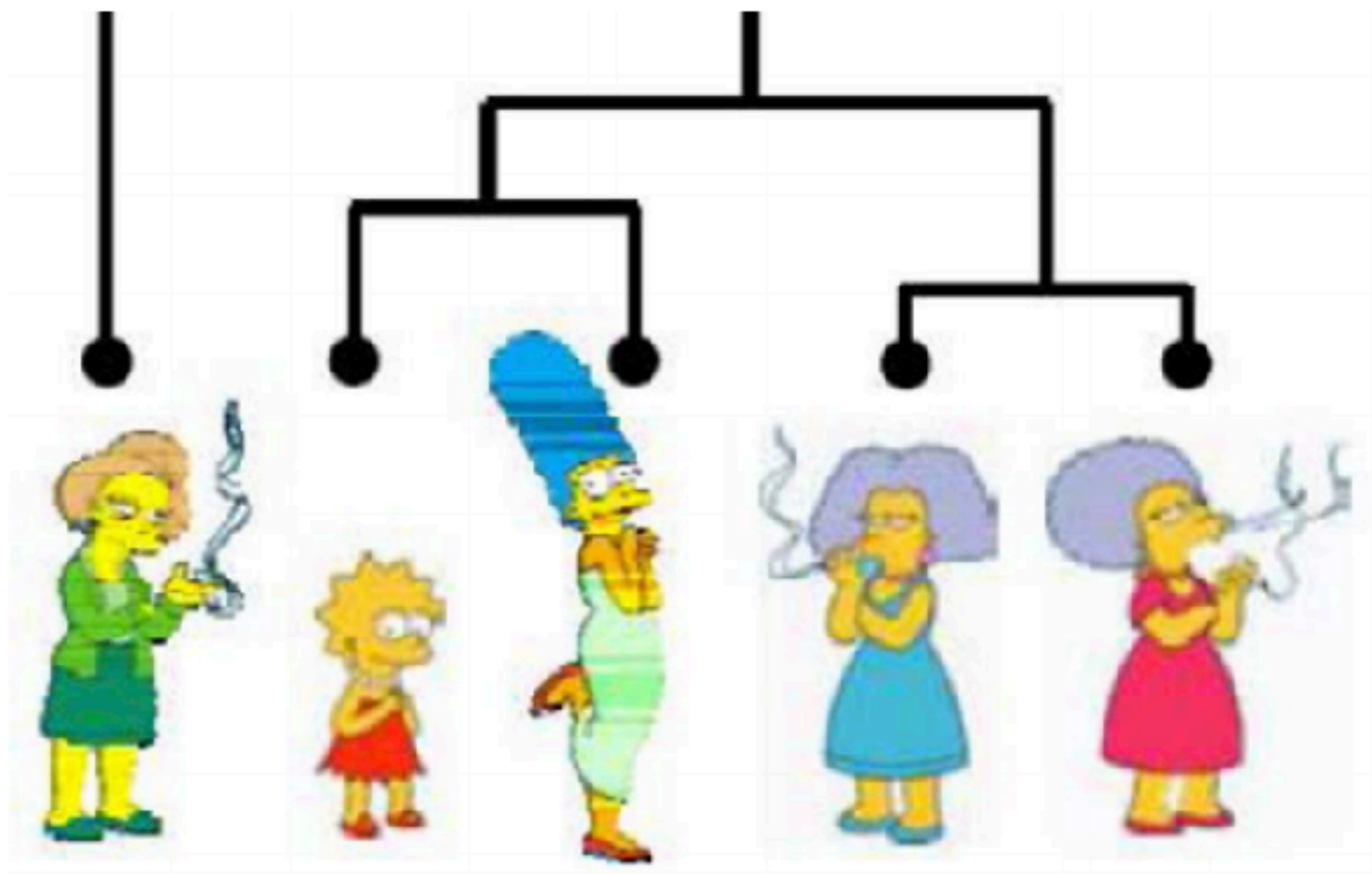
There is no universally applicable clustering technique

Clustering Algorithms

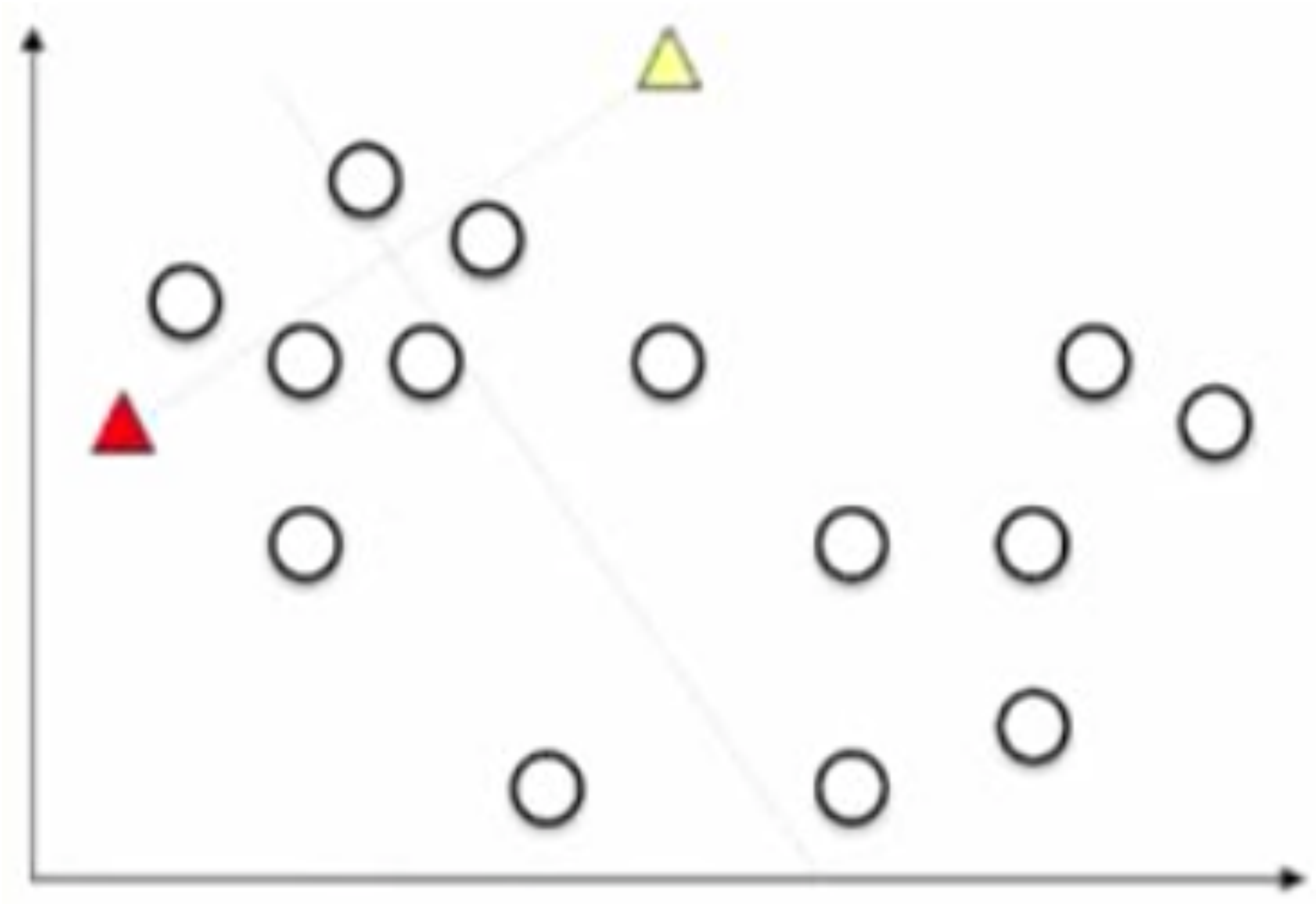
- Partition algorithms (Flat)
  - K-means
  - Mixture of Gaussian
  - Spectral Clustering



- Hierarchical algorithms
  - Bottom up – agglomerative
  - Top down – divisive



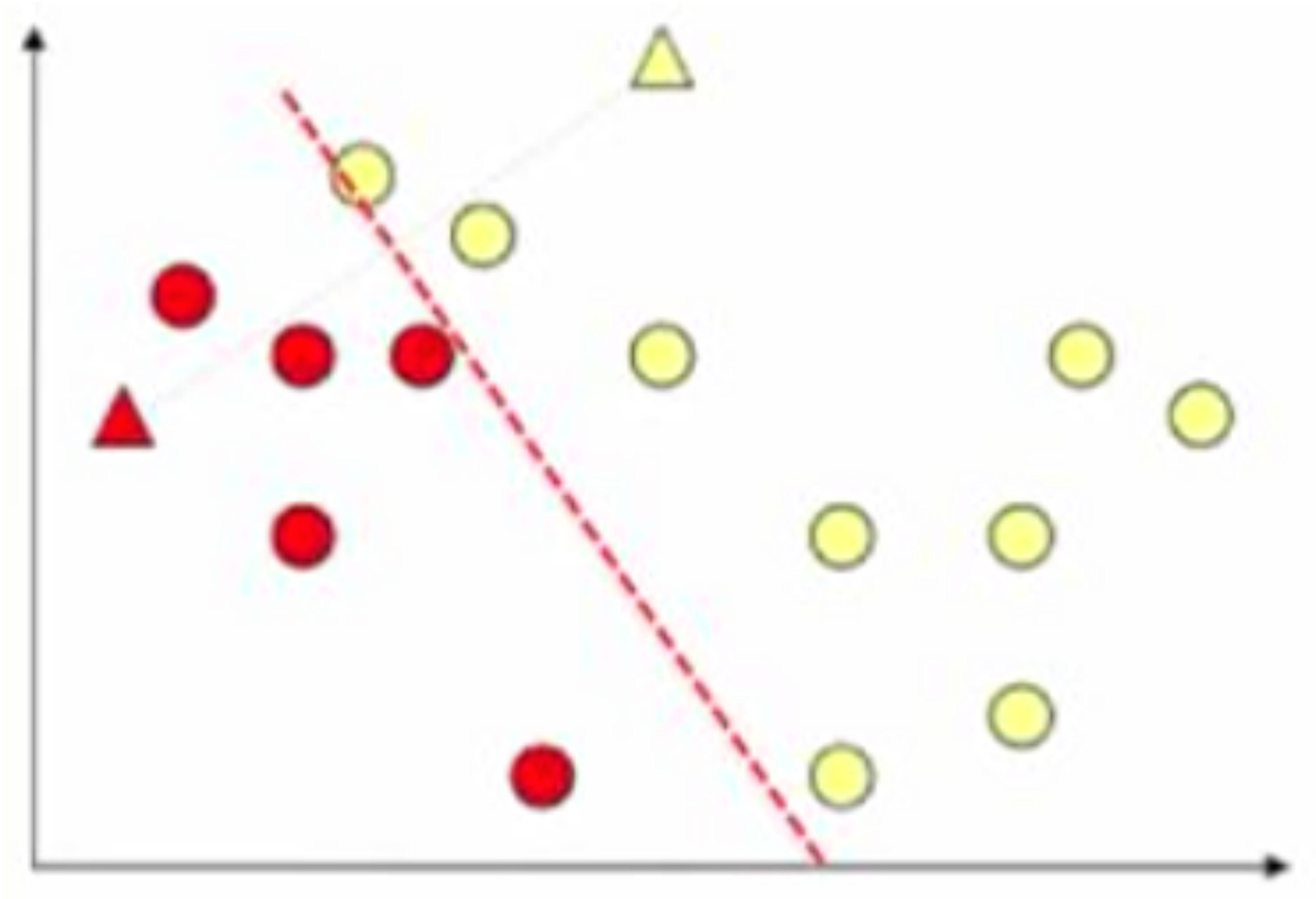
K-means clustering: Example



Algorithm 8.1 Basic K-means algorithm.

```
Select K points as initial centroids.
repeat
    Form K clusters by assigning each point to its closest centroid.
    Recompute the centroid of each cluster.
until Centroids do not change.
```

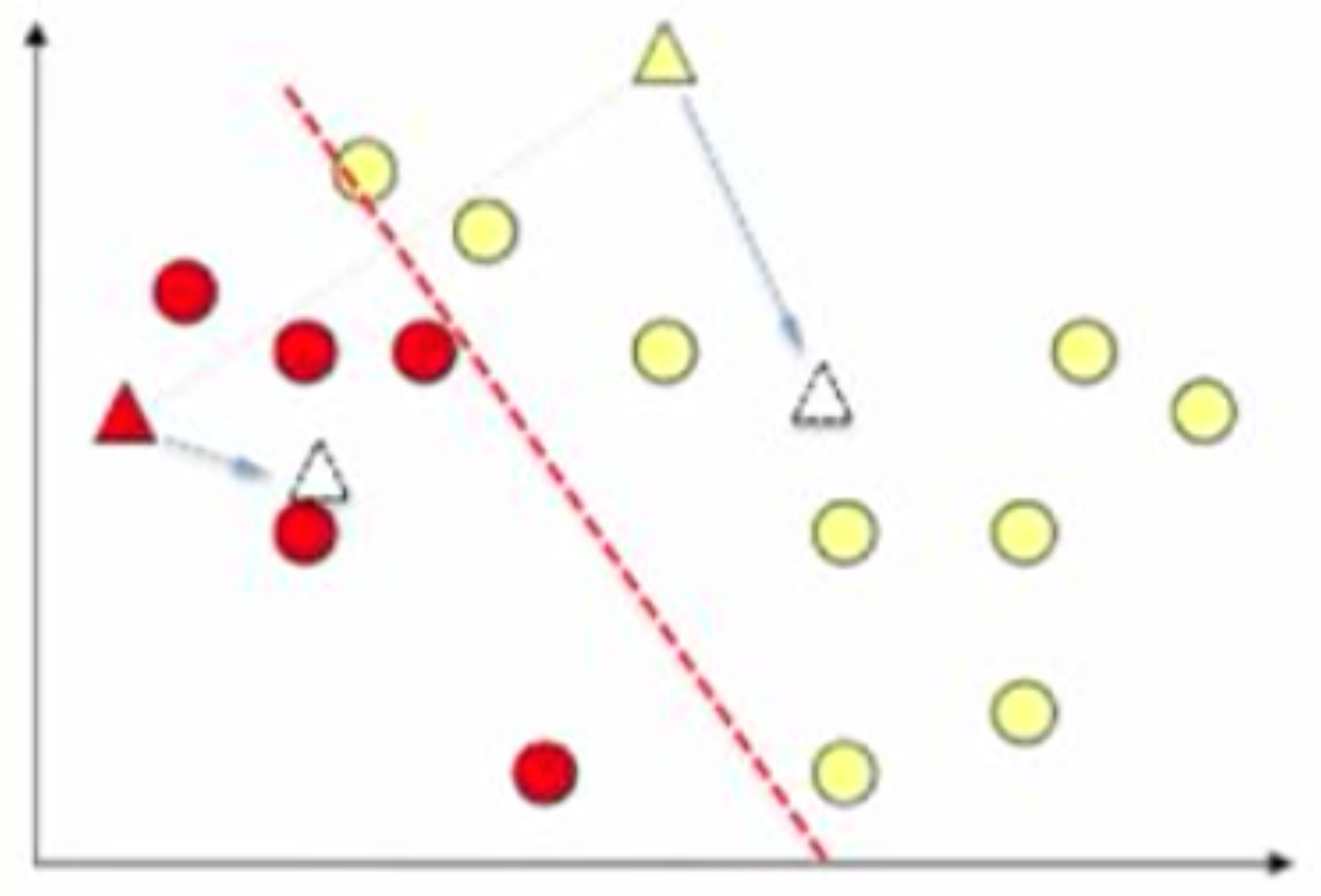
K-means clustering: Example



Algorithm 8.1 Basic K-means algorithm.

1: Select K points as initial centroids.
2: **repeat**
3: Form K clusters by assigning each point to its closest centroid.
4: Recompute the centroid of each cluster.
5: **until** Centroids do not change.

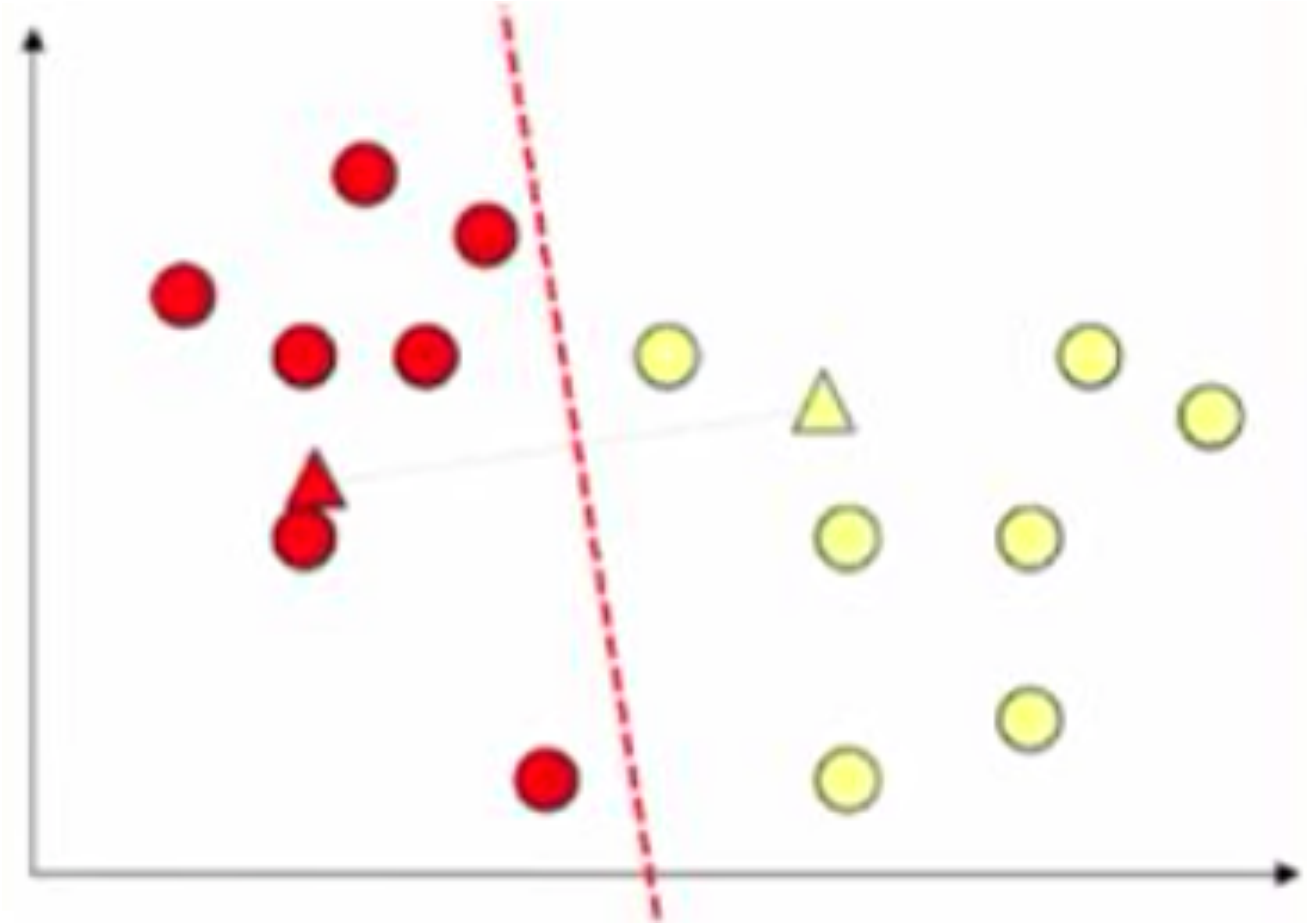
K-means clustering: Example



Algorithm 8.1 Basic K-means algorithm.

1: Select K points as initial centroids.
2: **repeat**
3:     Form K clusters by assigning each point to its closest centroid.
4:     Recompute the centroid of each cluster.
5: **until** Centroids do not change.

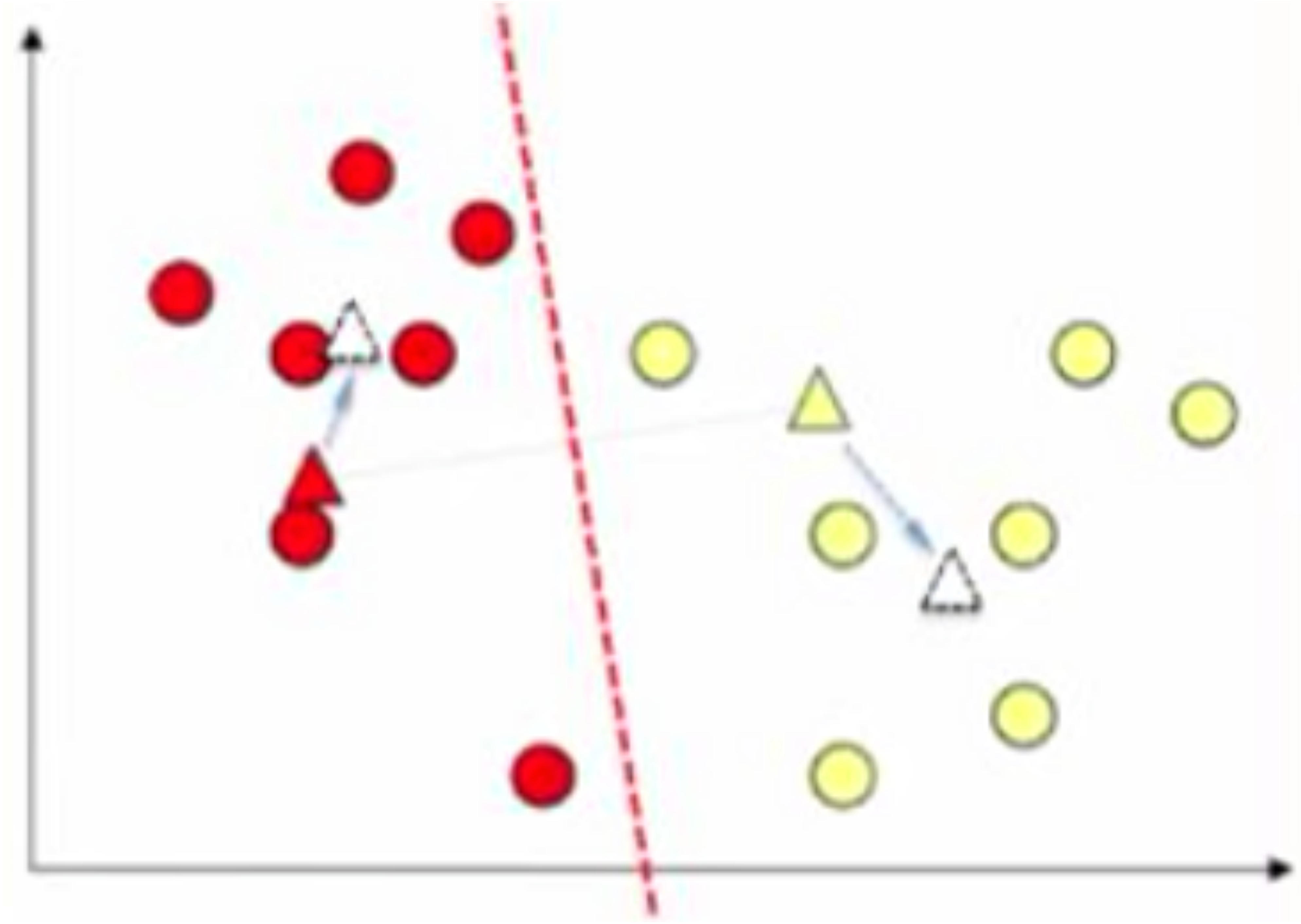
K-means clustering: Example



Algorithm 8.1 Basic K-means algorithm.

1: Select K points as initial centroids.
2: **repeat**
3: Form K clusters by assigning each point to its closest centroid.
4: Recompute the centroid of each cluster.
5: **until** Centroids do not change.

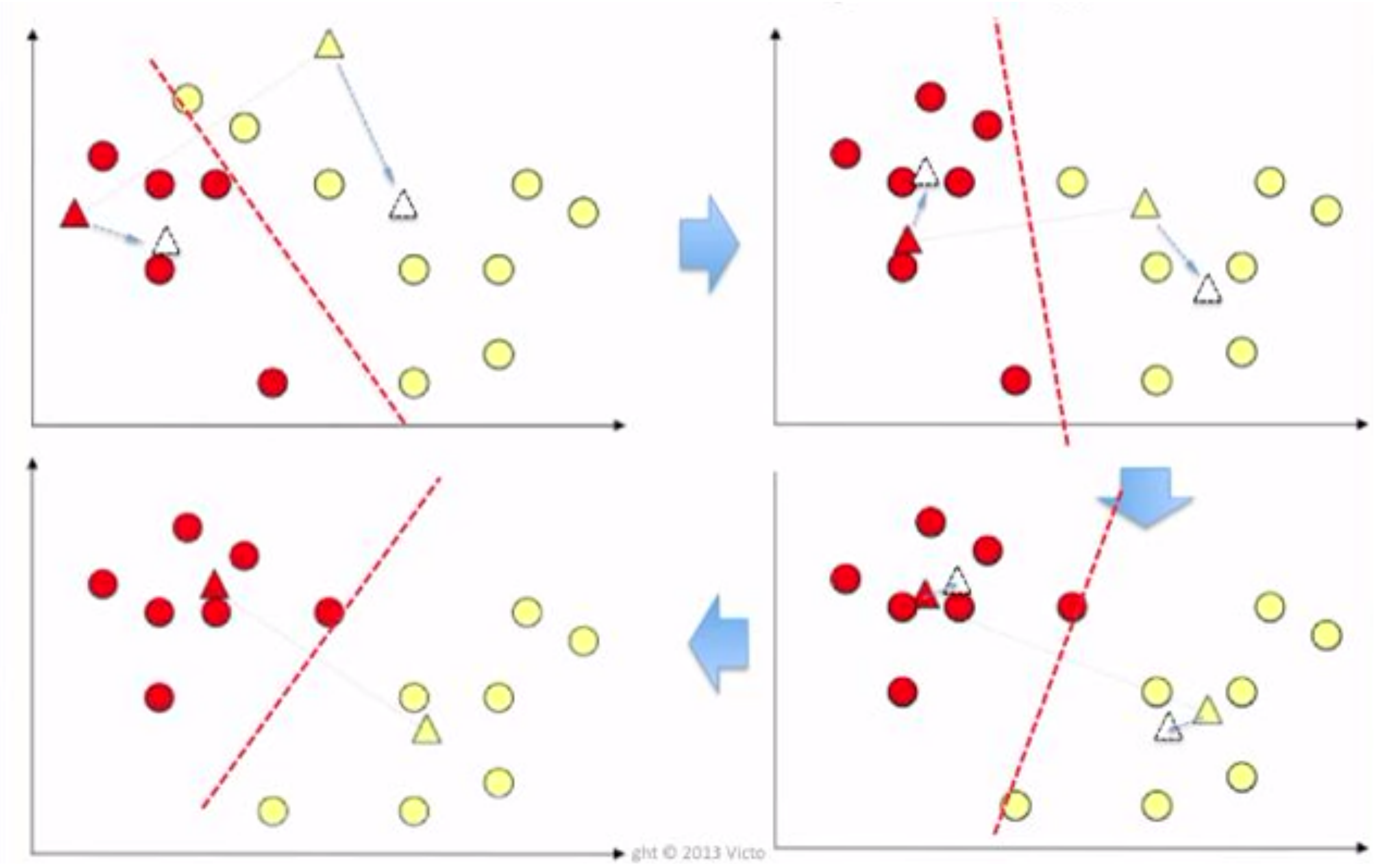
K-means clustering: Example



Algorithm 8.1 Basic K-means algorithm.

1: Select K points as initial centroids.
2: **repeat**
3: Form K clusters by assigning each point to its closest centroid.
4: Recompute the centroid of each cluster.
5: **until** Centroids do not change.

K-means clustering: Example



Algorithm 8.1 Basic K-means algorithm.

1: Select K points as initial centroids.
2: **repeat**
3:     Form K clusters by assigning each point to its closest centroid.
4:     Recompute the centroid of each cluster.
5: **until** Centroids do not change.

https://www.youtube.com/watch?v=_aWzGGNrcic

K-means clustering

Minimizes the sum of squared errors (SSE)

$$\mathrm{SSE} = \sum_{i=1}^{K} \sum_{\mathbf{x} \in C_i} dist(\mathbf{c}_i, \mathbf{x})^2$$

The centroid of the ith cluster is the mean of the points

$$\mathbf{c}_i = \frac{1}{m_i} \sum_{\mathbf{x} \in C_i} \mathbf{x}$$

Distance can be Euclidian, Manhattan, Minkowski,..

K-means clustering

Advantages

Easy to implement

Fast: $O(n)$

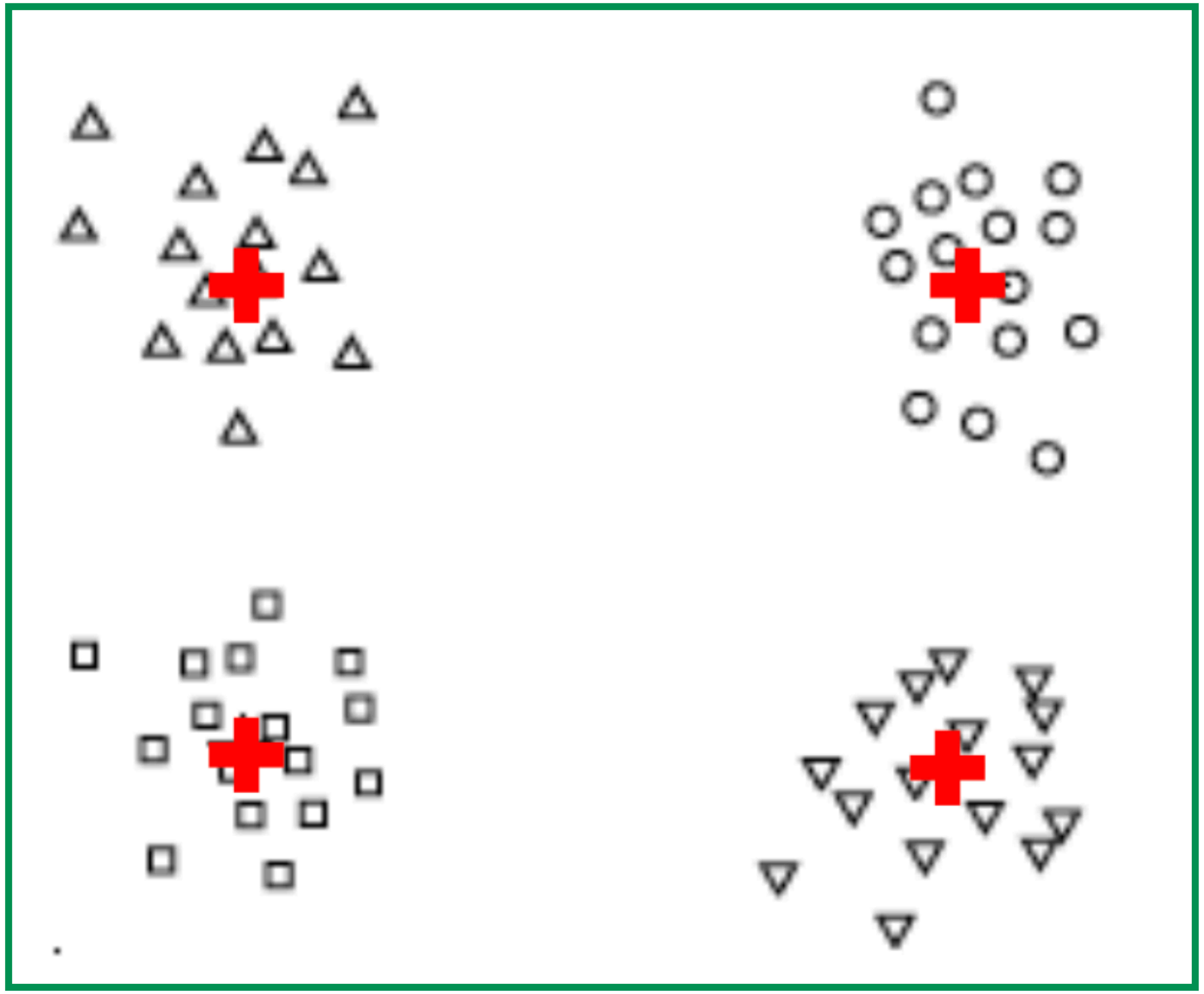
Disadvantages

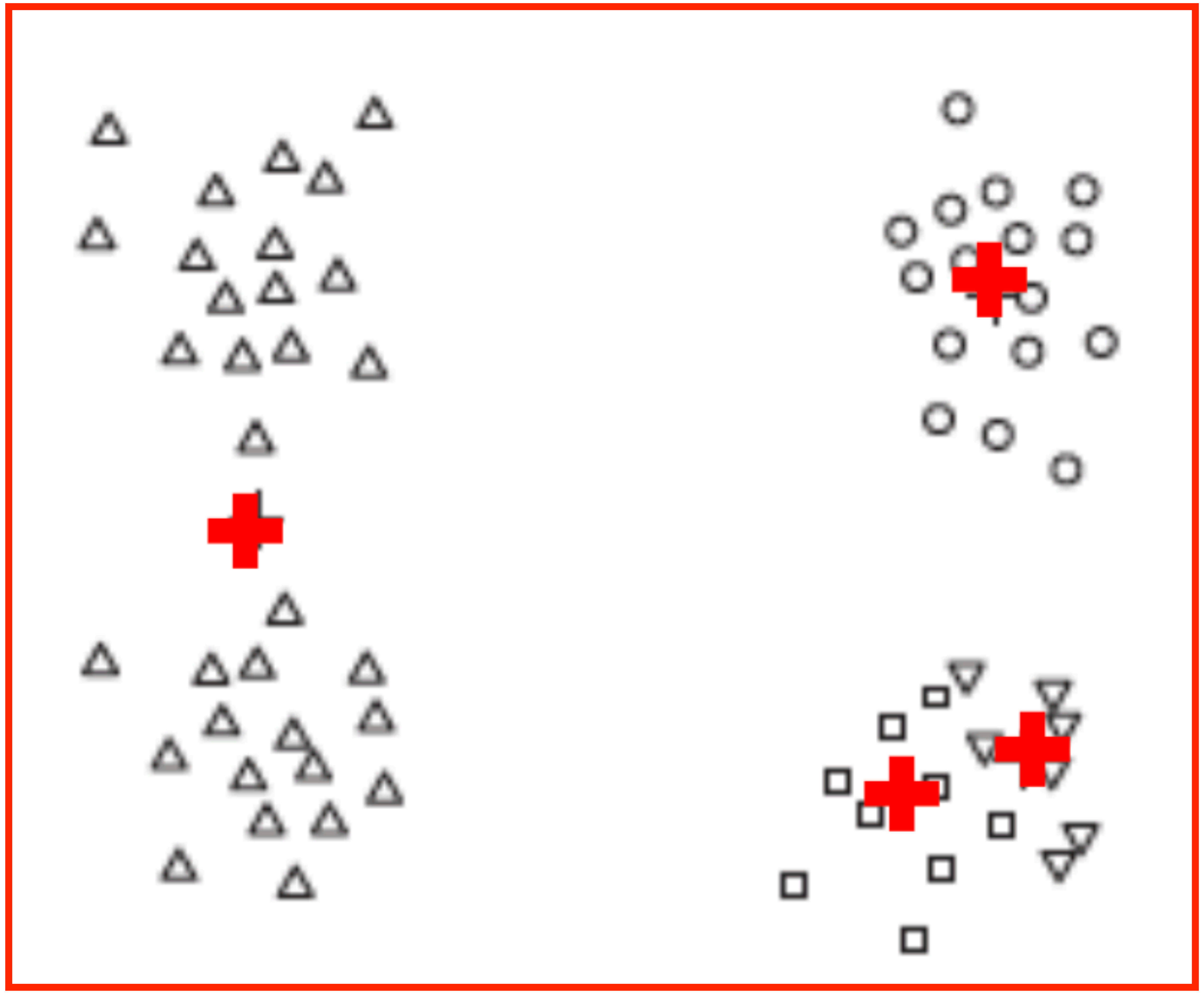
Number of partitions needs to be known

Sensitivity to initial conditions

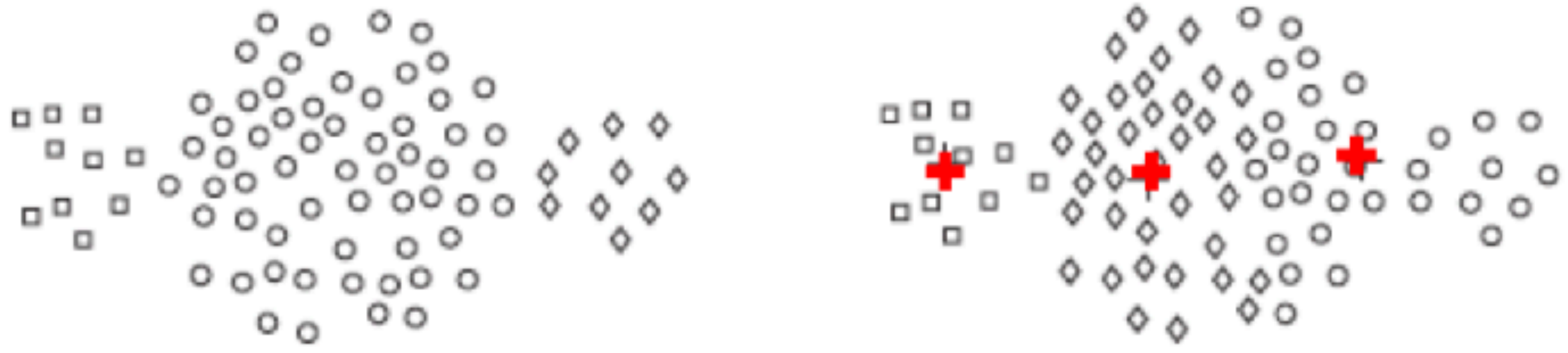
Not effective under several conditions

K-means can get stuck



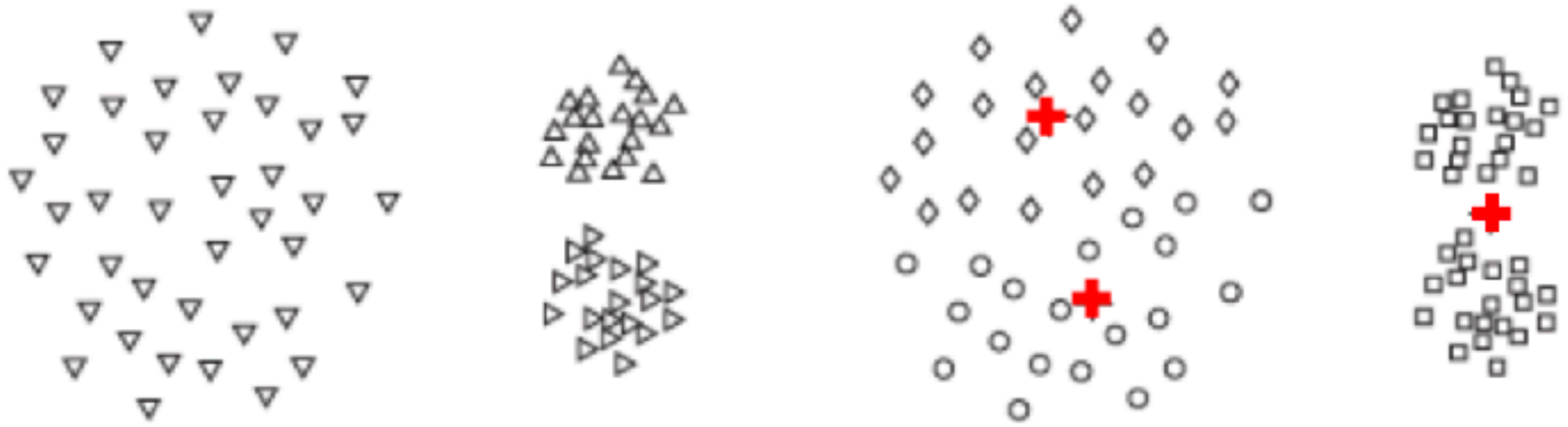


K-means is often not working well



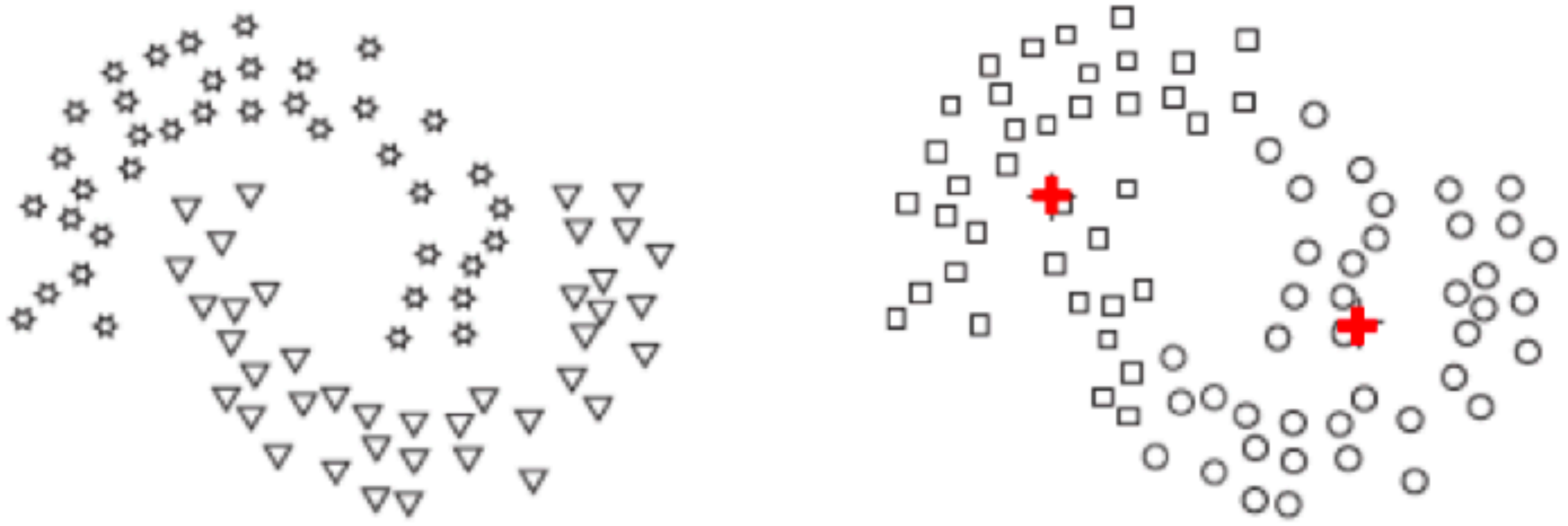
(a) Original points. (b) Three K-means clusters.

Figure 8.9. Clusters of different size



(a) Original points. (b) Three K-means clusters.

Figure 8.10 Clusters of different density



(a) Original points. (b) Two K-means clusters.

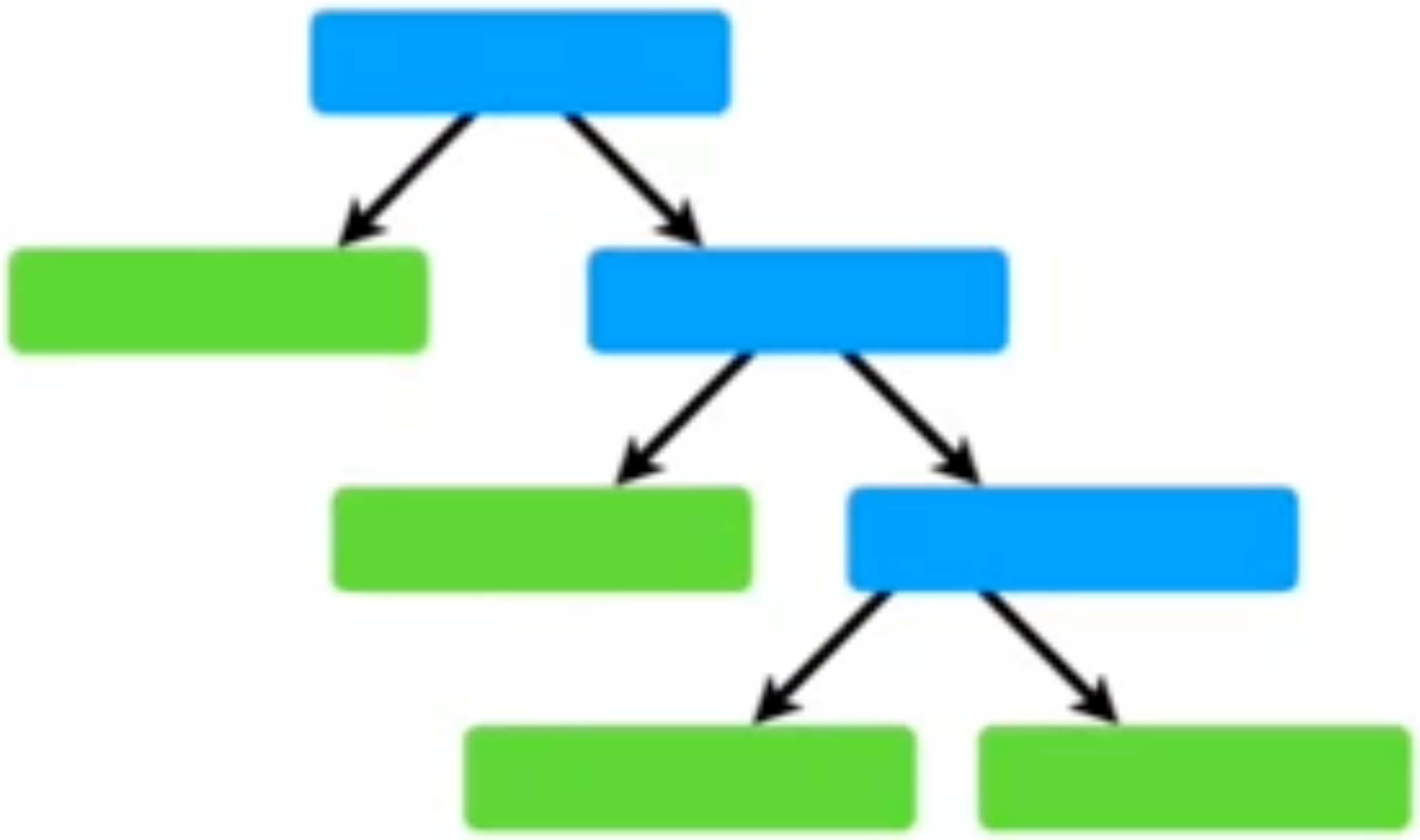
Figure 8.11. Non globular clusters

Today you learned about information theory

Entropy is the
expected surprise

$$H(X) = \mathbb{E}(I(X))$$

..useful for:
Decision trees



Data Science setup

git

Clustering