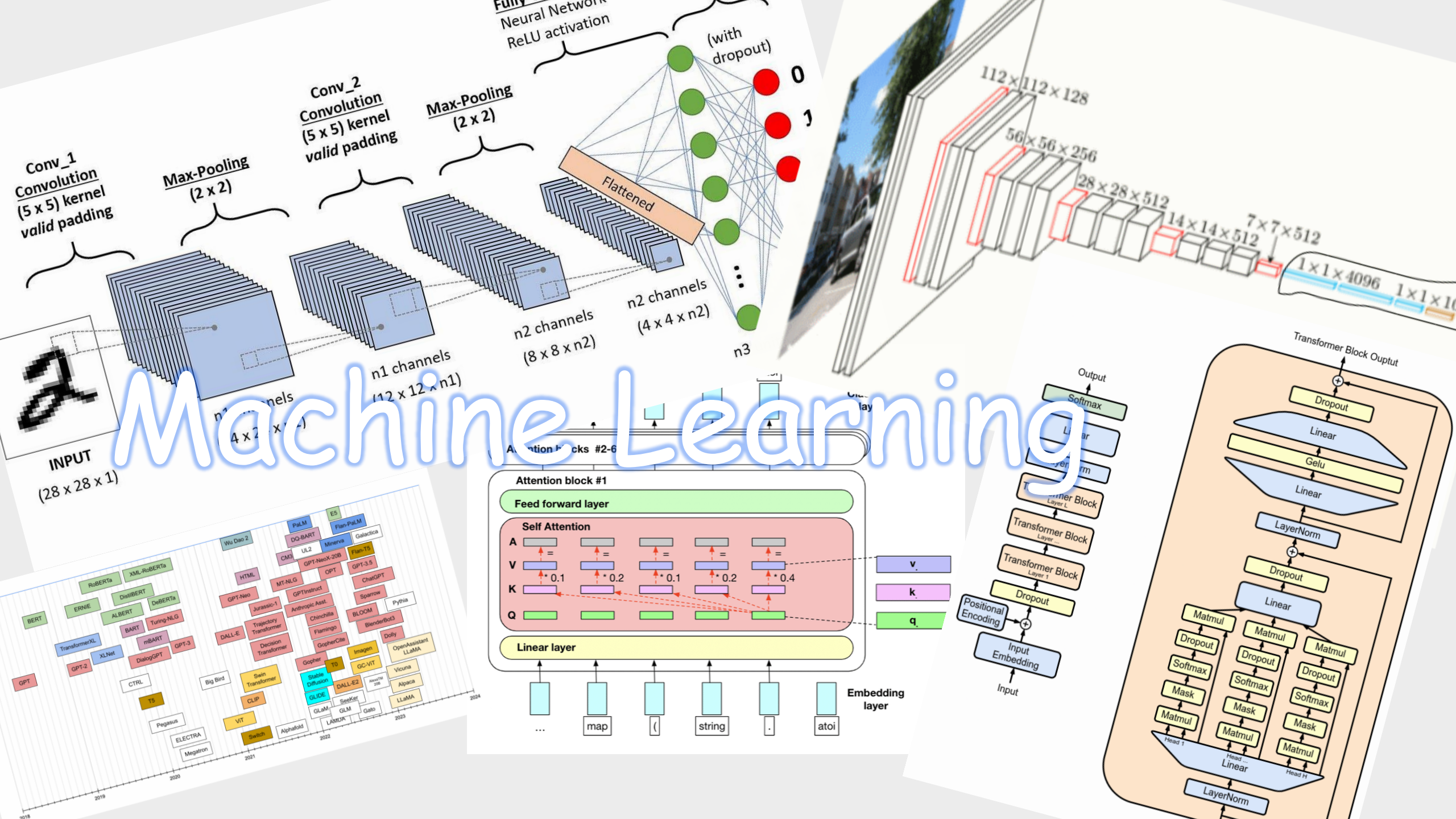
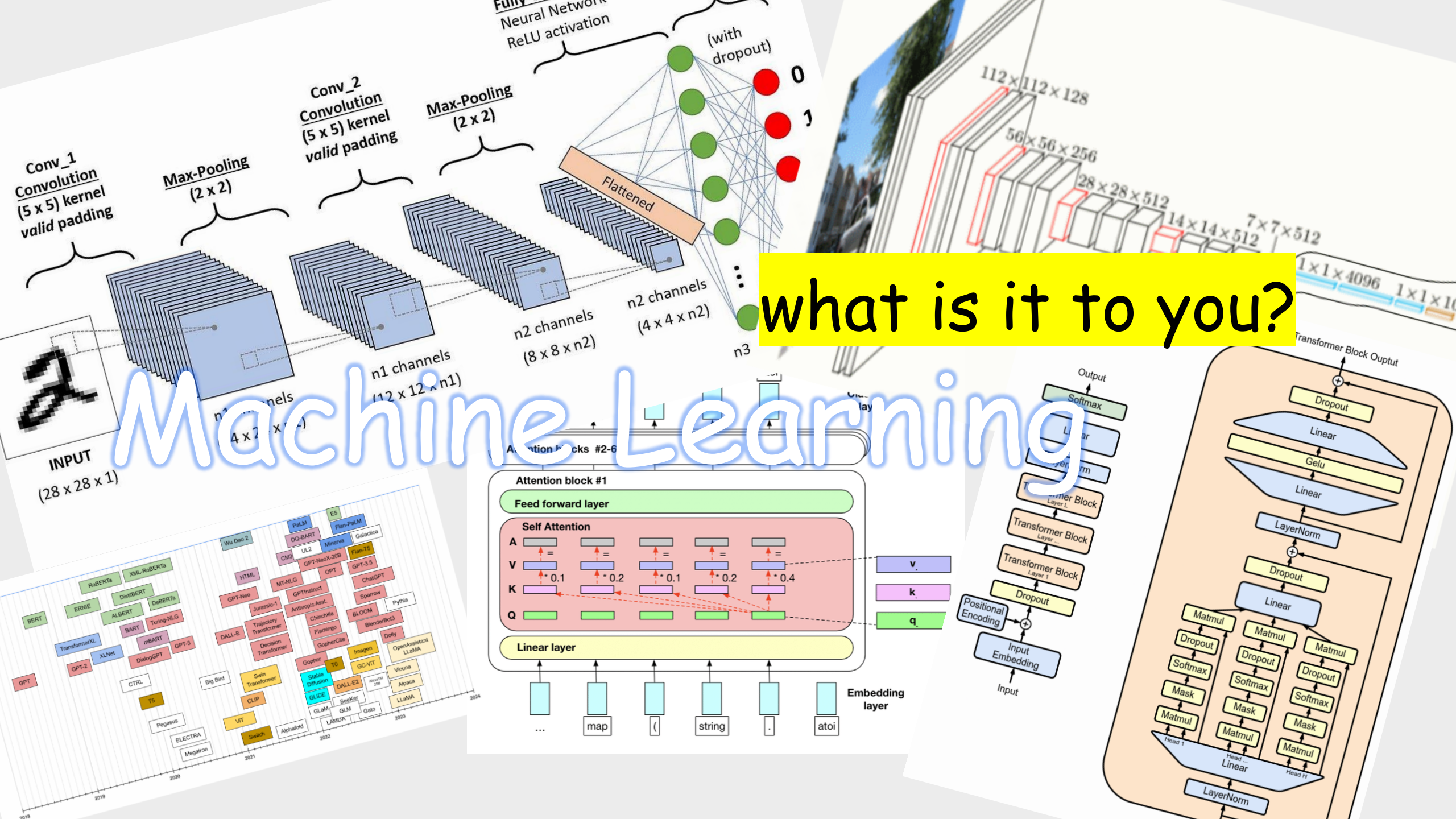

Machine Learning
Conv_1
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
Conv_2
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
Fully
Neural Network
ReLU activation
(with
dropout)
Flattened
INPUT
(28 x 28 x 1)
n1 channels
n2 channels
(8 x 8 x n2)
n2 channels
(4 x 4 x n2)
n3
112×112×128
56×56×256
28×28×512
14×14×512
7×7×512
1×1×4096
Attention block #1
Feed forward layer
Self Attention
Linear layer
Embedding
layer
Output
Softmax
Transformer Block
Positional
Encoding
Input
Embedding
Input
Transformer Block Ouptut
Dropout
Linear
Gelu
LayerNorm
Matmul
Softmax
Mask
Head 1
Head H
LayerNorm


Conv_1
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
Conv_2
Convolution
(5 x 5) kernel
valid padding
Max-Pooling
(2 x 2)
Fully
Neural Network
ReLU activation
(with
dropout)
Flattened
INPUT
(28 x 28 x 1)
n1 channels
(12 x 12 x n1)
n2 channels
(8 x 8 x n2)
n2 channels
(4 x 4 x n2)
n3
112×112×128
56×56×256
28×28×512
14×14×512
7×7×512
1×1×4096
what is it to you?
Machine Learning
Attention block #1
Feed forward layer
Self Attention
A
V
K
Q
* 0.1
* 0.2
* 0.1
* 0.2
* 0.4
v
k
q
Linear layer
Embedding
layer
...
map
(
string
.
atoi
Output
Softmax
Transformer Block
Transformer Block
Transformer Block
Dropout
Positional
Encoding
Input
Embedding
Input
Transformer Block Ouptut
Dropout
Linear
Gelu
Linear
LayerNorm
Dropout
Linear
Matmul
Dropout
Softmax
Mask
Matmul
Head 1
Head ...
Head H
Linear
LayerNorm
BERT
RoBERTa
XML-RoBERTa
ERNIE
DistilBERT
ALBERT
DeBERTa
BART
Turing-NLG
TransformerXL
XLNet
mBART
GPT-2
DialogGPT
GPT-3
GPT
CTRL
T5
Pegasus
ELECTRA
Megatron
Wu Dao 2
PaLM
E5
Flan-PaLM
DQ-BART
Galactica
CM3
UL2
Minerva
Flan-T5
GPT-NeoX-20B
OPT
GPT-3.5
ChatGPT
HTML
MT-NLG
GPT-Neo
GPTInstruct
Sparrow
Jurassic-1
Anthropic Asst.
Pythia
DALL-E
Trajectory
Transformer
Chinchilla
BLOOM
BlenderBot3
Decision
Transformer
Flamingo
Dolly
GopherCite
Imagen
OpenAssistant
LLaMA
Gopher
T0
GC-ViT
Vicuna
Big Bird
Swin
Transformer
Stable
Diffusion
DALL-E2
Alpaca
CLIP
GLIDE
SeeKer
LLaMA
ViT
GLaM
GLM
Gato
Switch
Alphafold
LaMDA
2018
2019
2020
2021
2022
2023
2024

lesson 1

Machine Learning

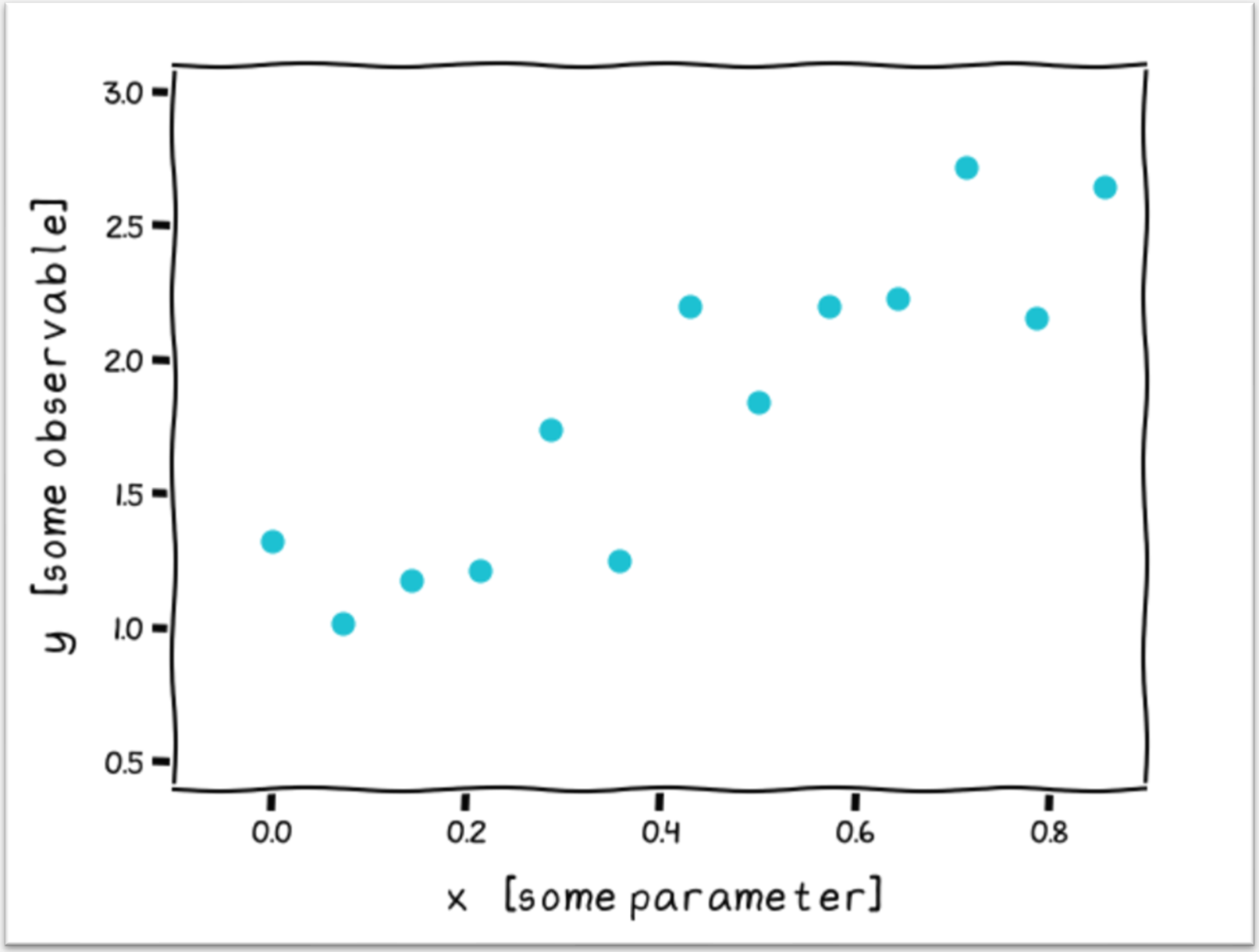
creating a model (with some parameters)

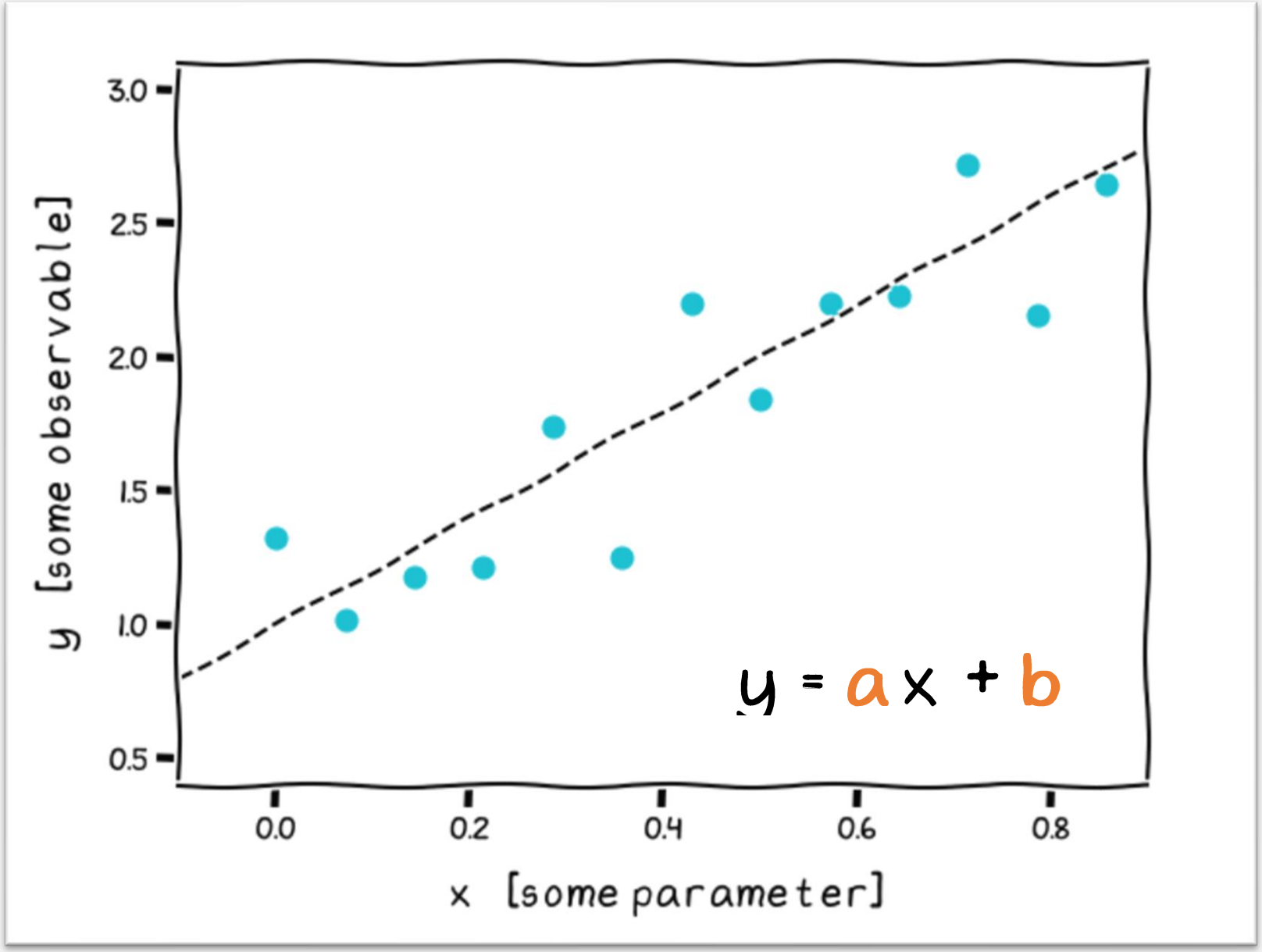
then optimizing it

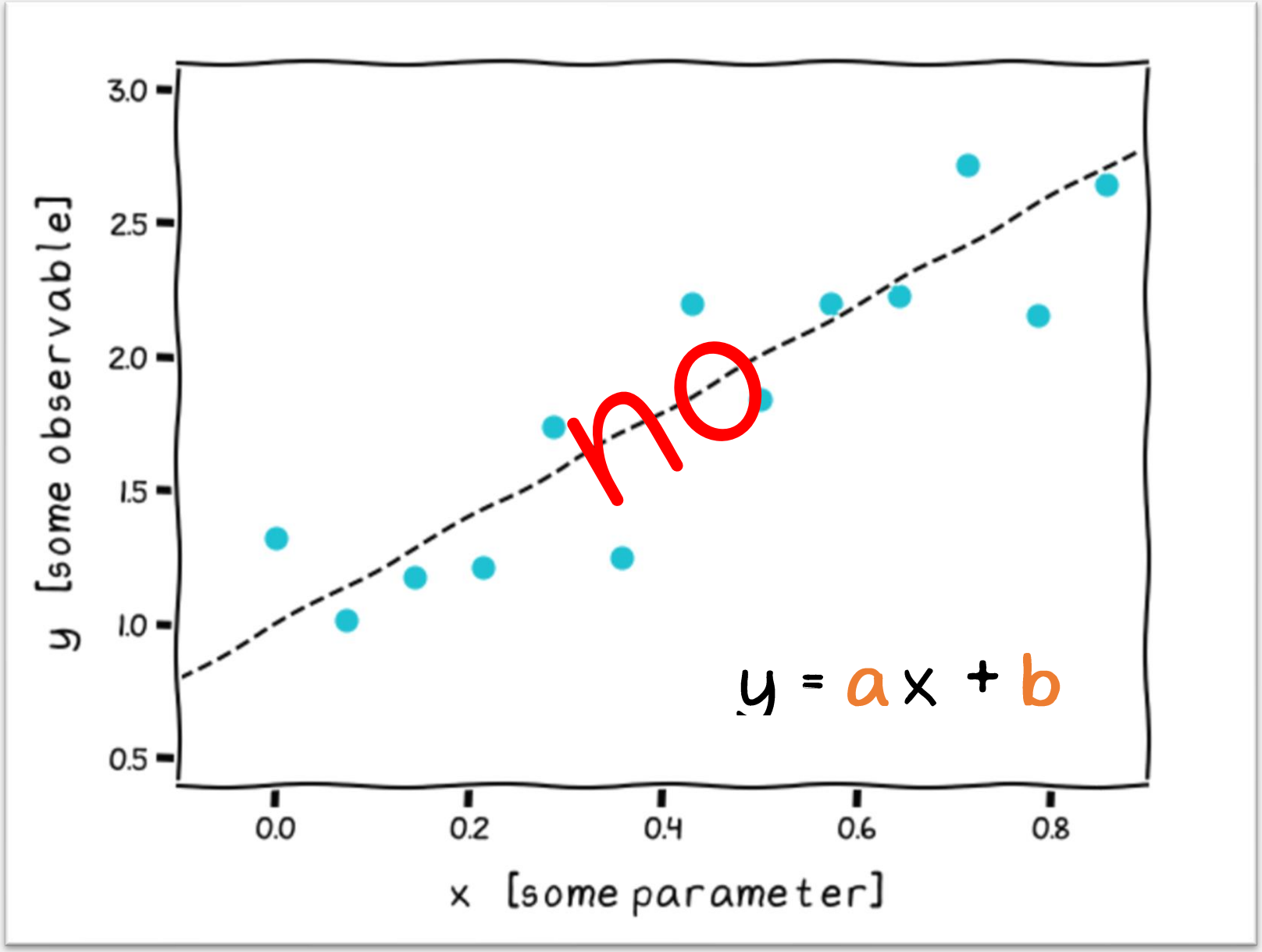
to predict something

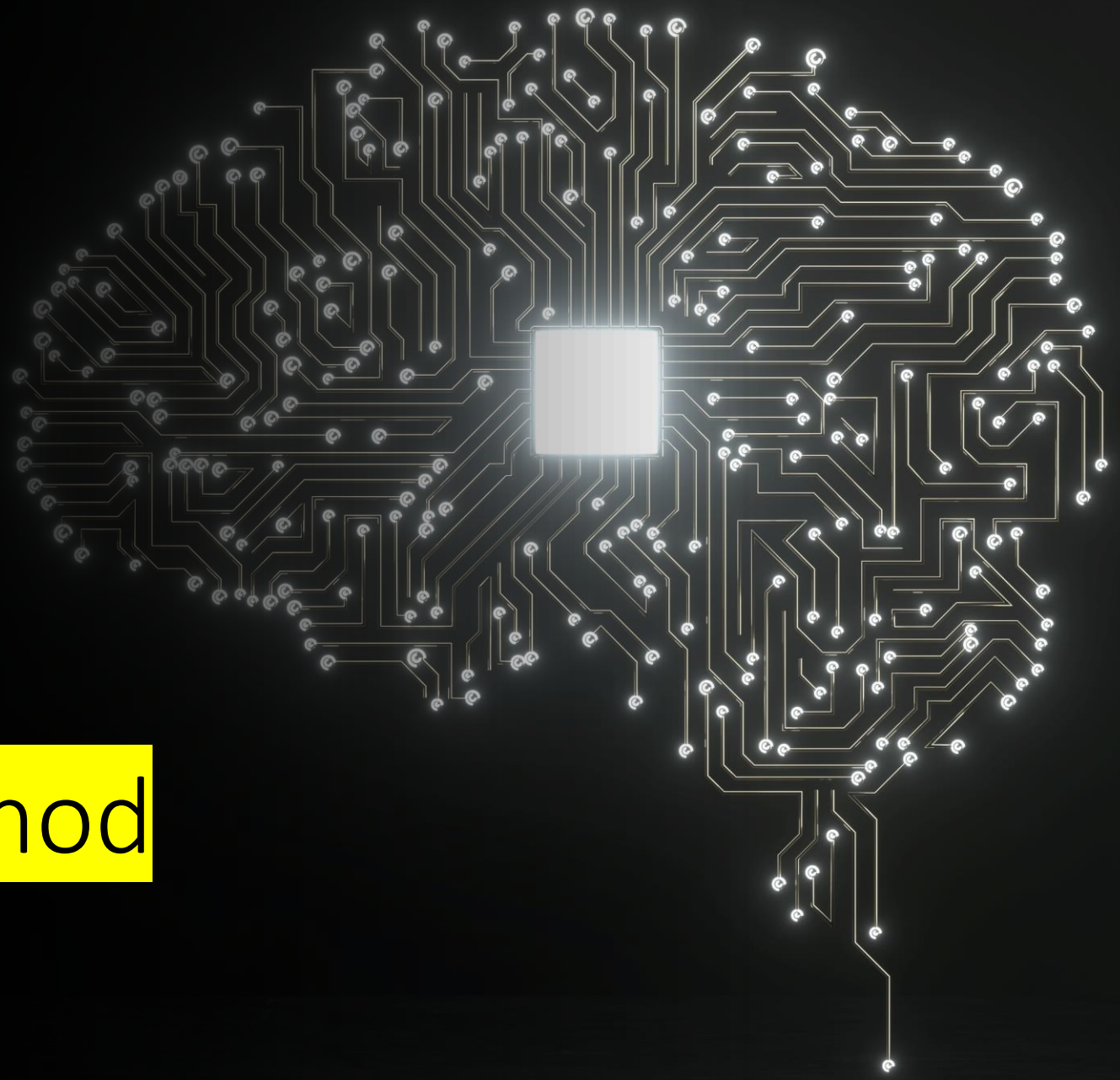
(with a computer algorithm)

matplotlib.pyplot.xkcd




3.0
2.5
2.0
1.5
1.0
0.5
y [some observable]
0.0
0.2
0.4
0.6
0.8
x [some parameter]
y = a x + b


3.0
2.5
2.0
1.5
1.0
0.5
y [some observable]
0.0
0.2
0.4
0.6
0.8
x [some parameter]
no
y = a x + b



the scientific method

and the philosophy of machine learning

1. Form a hypothesis
2. Collect data
3. Split the data
4. Train your model
5. Evaluate your model
6. Think about your model
7. Think of the next project

scientific method
In a nutshell
(ML edition)

1 Form a hypothesis

Before looking at any data, decide for yourself:

- What do you expect to find?
- What is your model?
- Do you plan to compare with other findings? If so, which?
- What do you consider “a good fit” (if modelling)?
- What do you consider “a discrepant result” or “a tension”?
  E.g., how many sigmas do you consider significant?

Looking at the data will bias you!

Example *The Black Cloud* (Hoyle, 1958). The Black Cloud appears to be heading for the Earth. The scientific team suggests that this proves the cloud has intelligence. Not so, says the dissenting team member.

Example *The Black Cloud* (Hoyle, 1958). The Black Cloud appears to be heading for the Earth. The scientific team suggests that this proves the cloud has intelligence. Not so, says the dissenting team member. Why? A golf ball lands on a golf course which contains 10^7 blades of grass; it stops on one blade; the chances are 1 in 10^7 of this event occurring by chance. This is not so amazing – the ball had to land somewhere. It would only be amazing if the experiment were re-run to test the newly formulated hypothesis (e.g. the blade being of special attractive character; the golfer of unusual skill) and the event was repeated. However, the importance of deciding if the Black Cloud knew about the Earth cannot await the next event or the sequence of events, and tempts the rush to judgement in which initial data, hypothesis and test data are combined; so in many instances in astronomy and cosmology.

1 Form a hypothesis

Before looking at any data, decide for yourself:

- What do you expect to find?
- What is your model?
- Do you plan to compare with other findings? If so, which?
- What do you consider “a good fit” (if modelling)?
- What do you consider “a discrepant result” or “a tension”?
  E.g., how many sigmas do you consider significant?

Looking at the data will bias you!

1 Form a hypothesis

Before looking at any data, decide for yourself:

- What do you expect to find?
- ~~What is your model?~~
- Do you plan to compare with other findings? If so, which?
- What do you consider “a good fit” (if modelling)?
- What do you consider “a discrepant result” or “a tension”?
  E.g., how many sigmas do you consider significant?

Looking at the data will bias you!

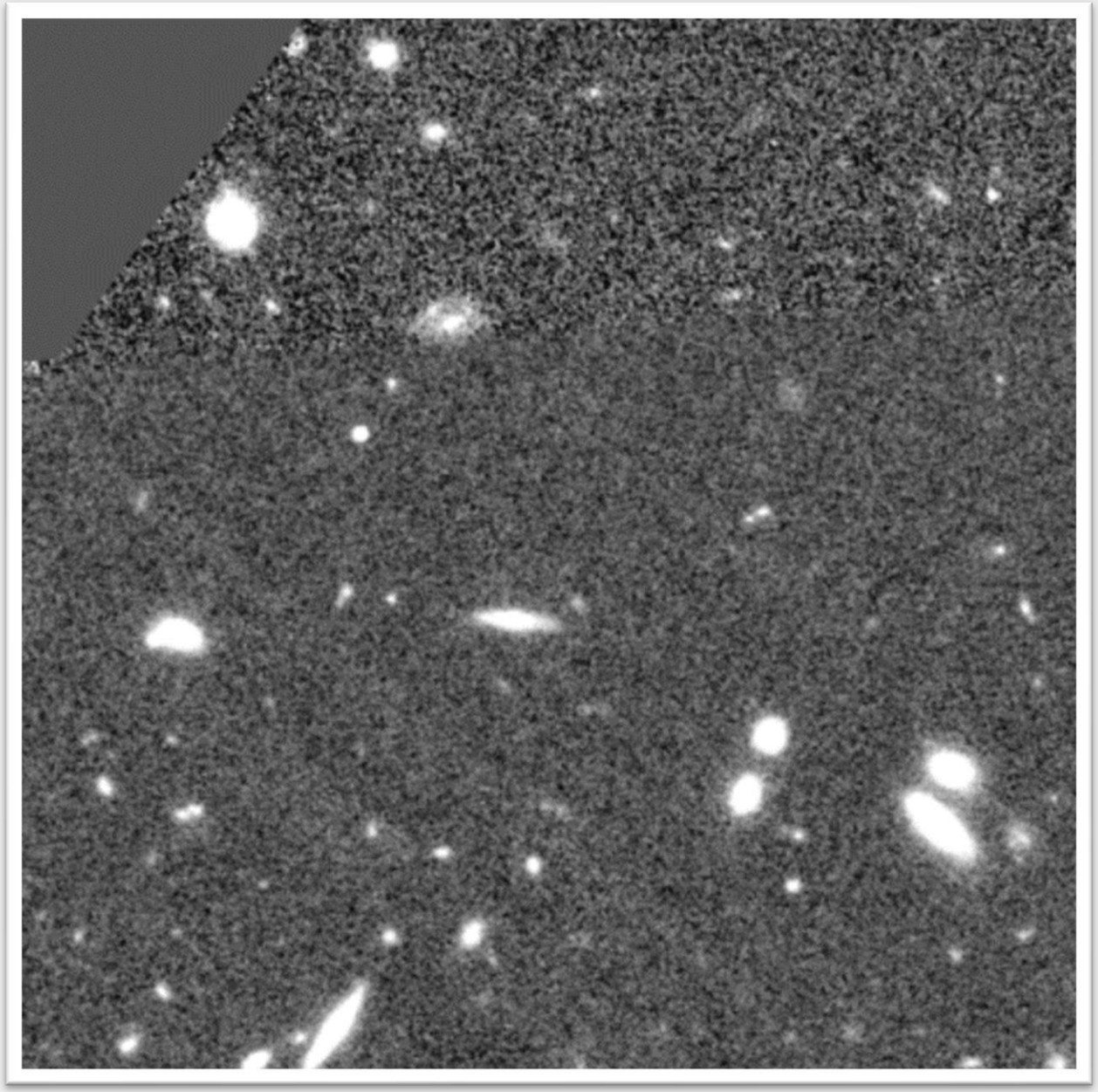
2 Collect data

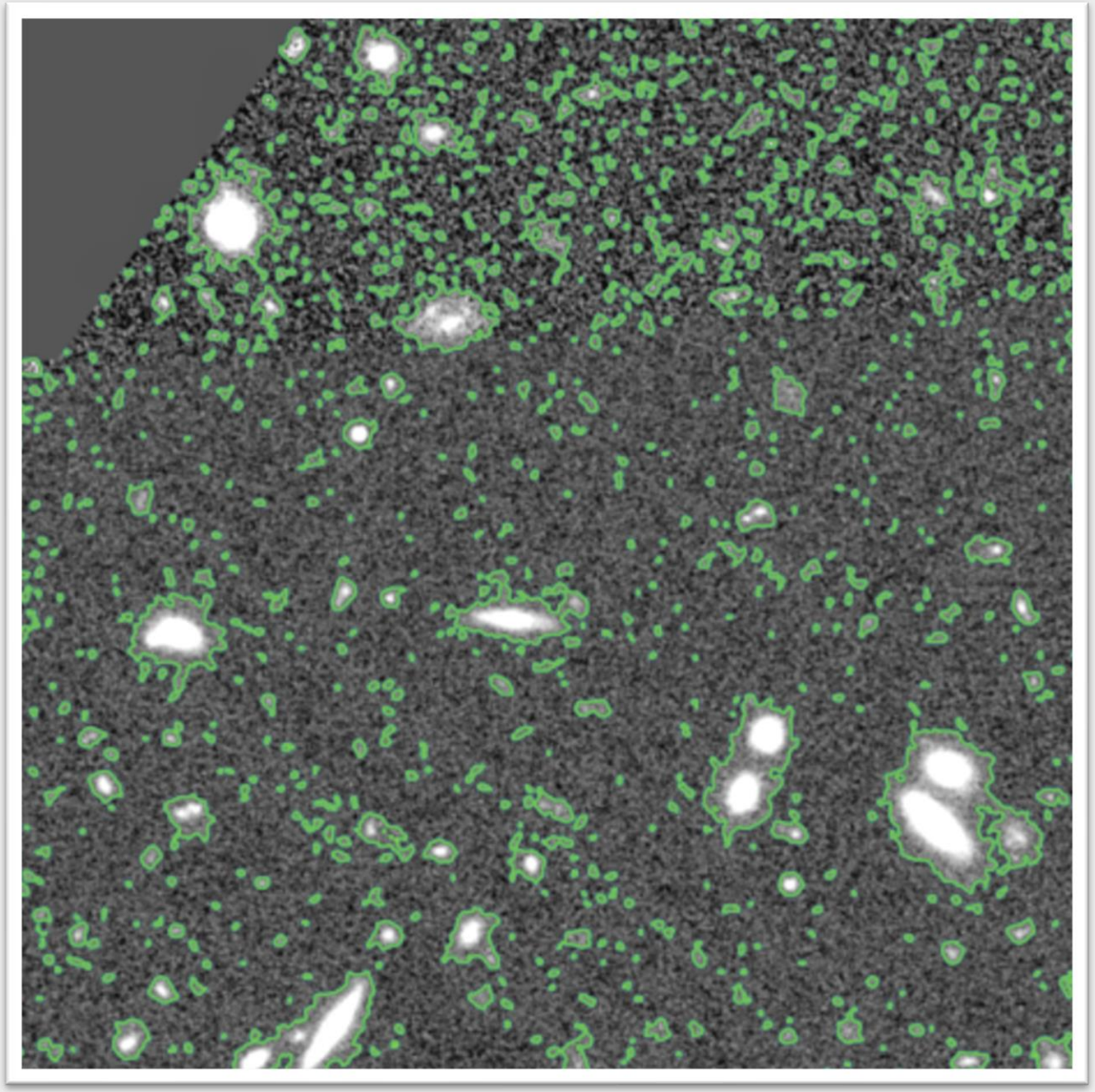
No turning on your fancy ML machine yet

- Run the experiment and preprocess data
- Preprocess collected data for analysis

  Cleaning?
  Outlier detection?
  Normalization?

Look at your data!





`sklearn.model_selection.train_test_split`

3 Split the data

The ultimate ML lifehack

- 70% training set

  The dataset you will fit everything on

  It should not change between re-runs of your code/model fitting/etc

  (set random_state to a number when sampling your training set)

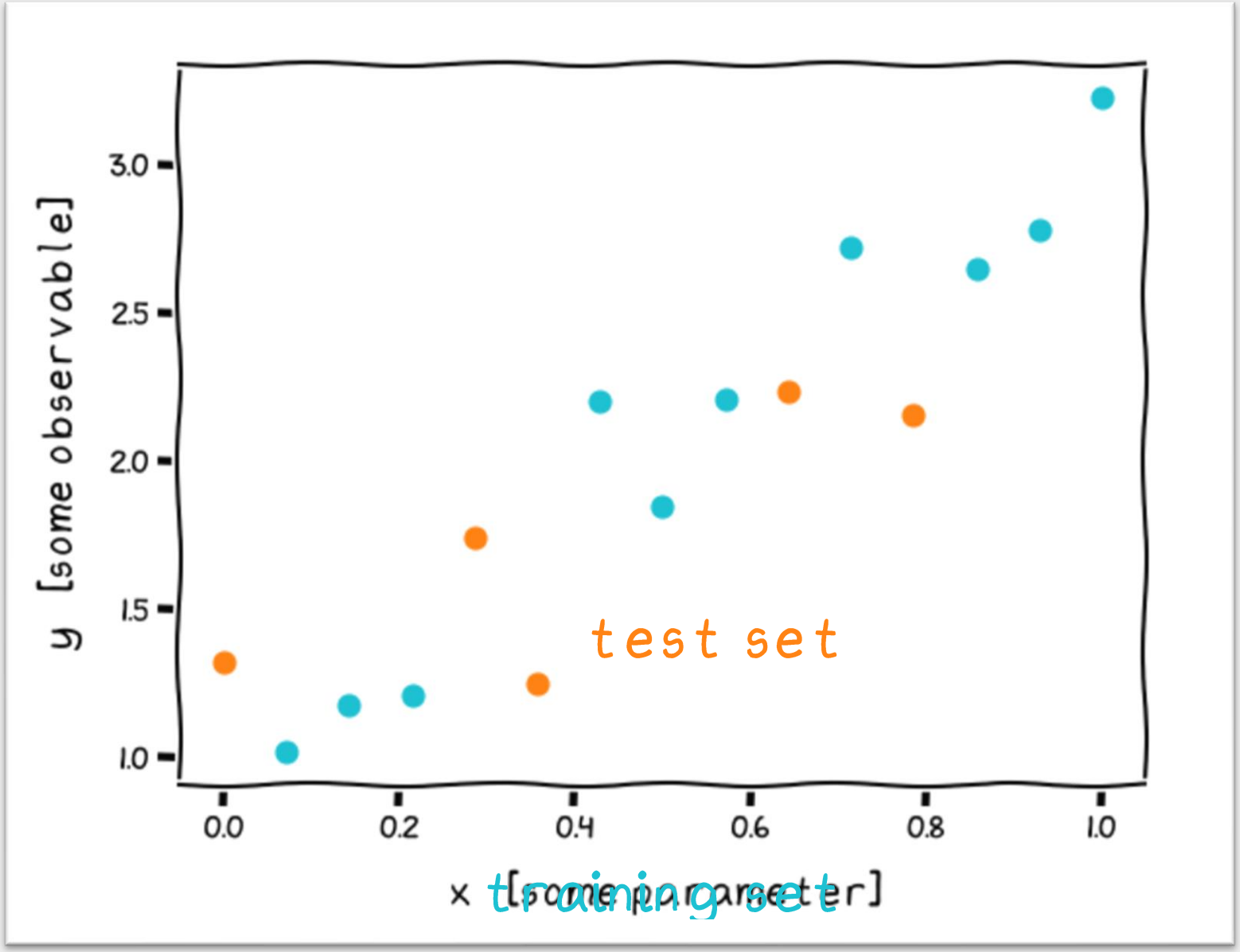
- 30% validation set

  The dataset you test the model's performance on

- 10% test set [optional]

  After the model is optimized & you are writing the paper, this is the final test

No need to re-run experiments!


y [some observable]
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0
x [some parameter]
test set
training set

4

“Train” the model

aka optimize your model parameters
using whatever approach you prefer

only look at your training set!

5

Evaluate the model

`sklearn.metrics`

loss functions
accuracy scores
precision/recall
confusion matrices

Goodness-of-fit, aka "loss"

The loss function can be anything you like

- Fitting a curve: χ^2, reduced χ^2, RMS loss
- Classification: binary cross-entropy
- More advanced techniques: regularization? Bayesian? Probabilities?

This is also done during training

More on gradient descent + parameter optimization next week

What data do we evaluate the model with?

5

Evaluate the model

`sklearn.metrics`

loss functions
accuracy scores
precision/recall
confusion matrices

Goodness-of-fit, aka "loss"

The loss function can be anything you li

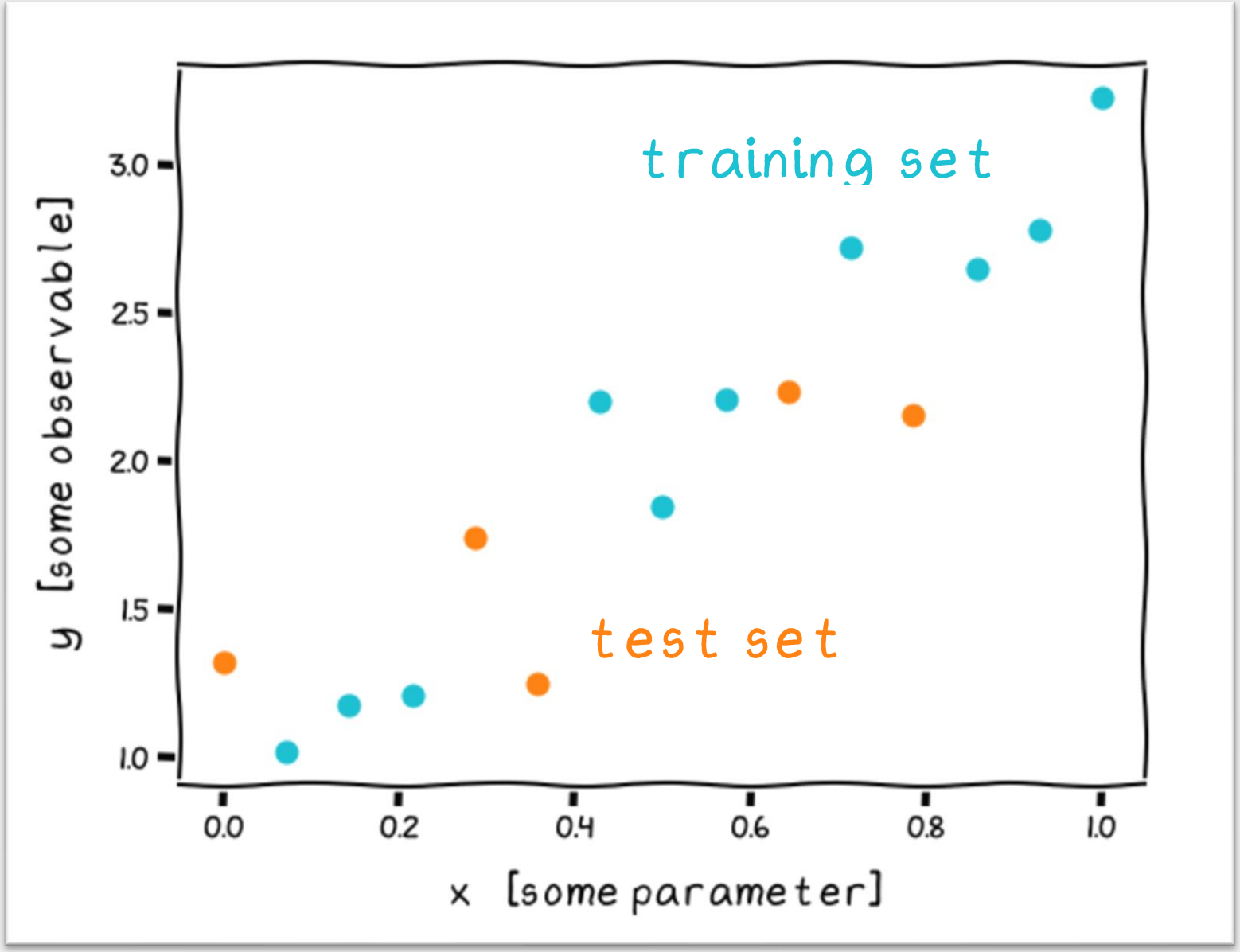
- Fitting a curve: χ^2 r
- Classificat
- More adva ian? Probabilities?

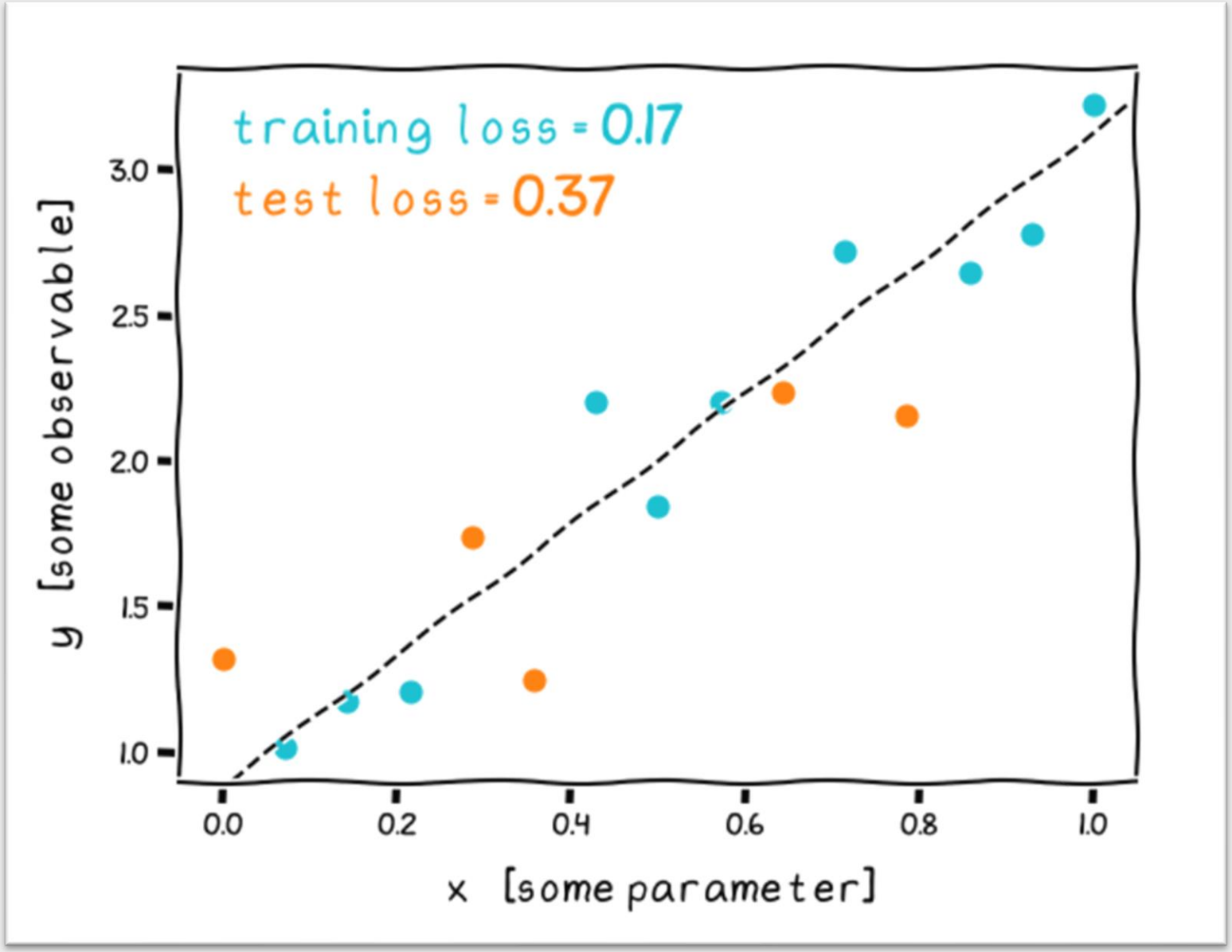
Both
(but mostly test set)

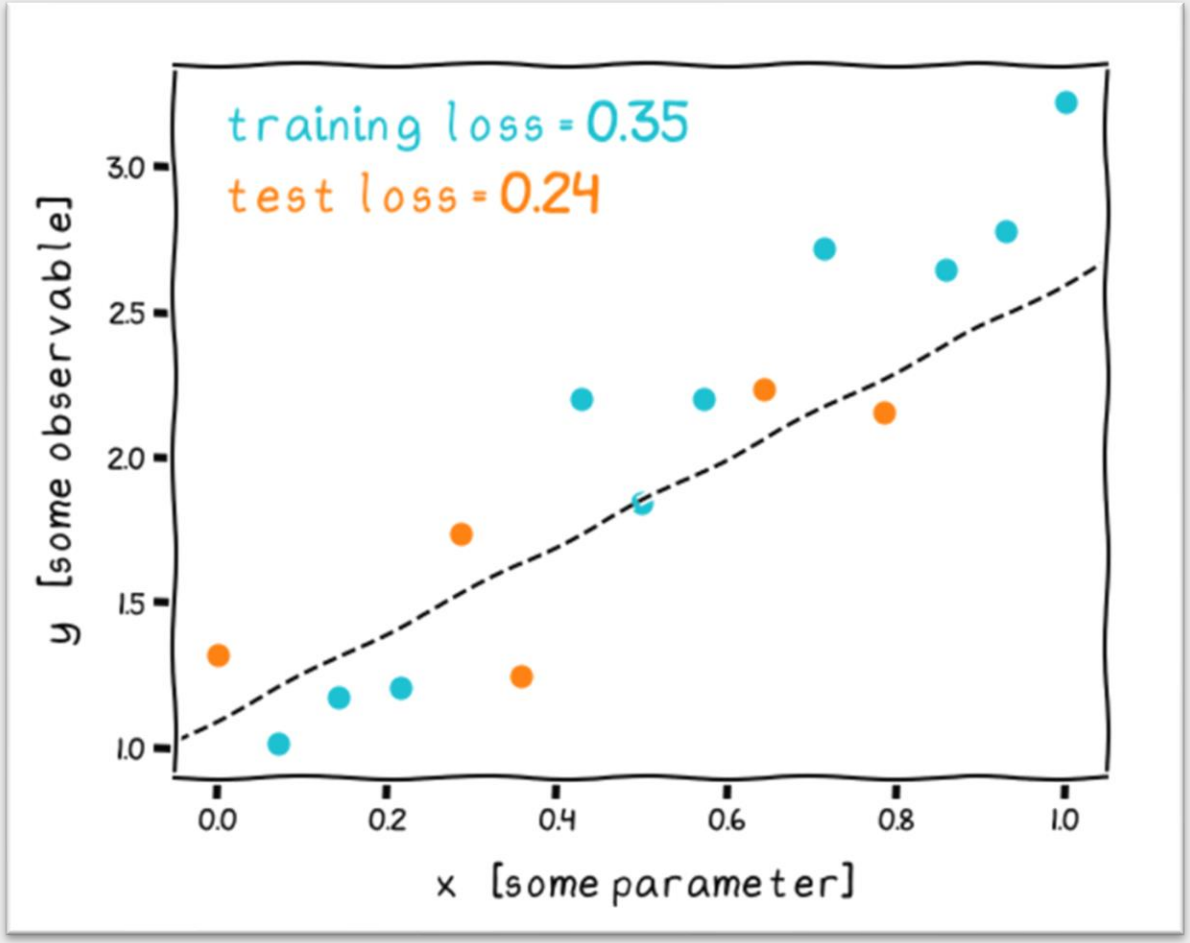
This is also done duri training

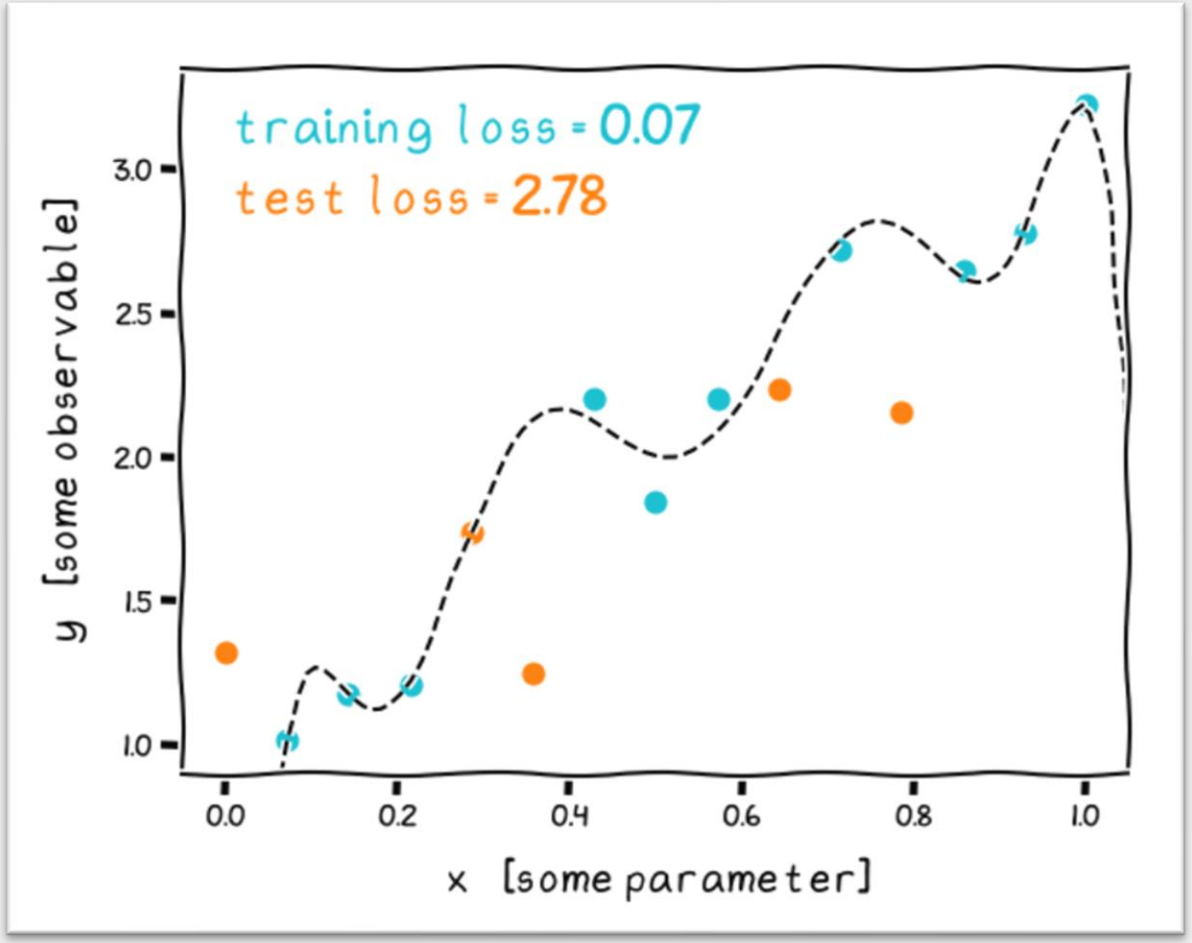
More on gradient descent + parameter optimization next week

What data do we evaluate the model with?

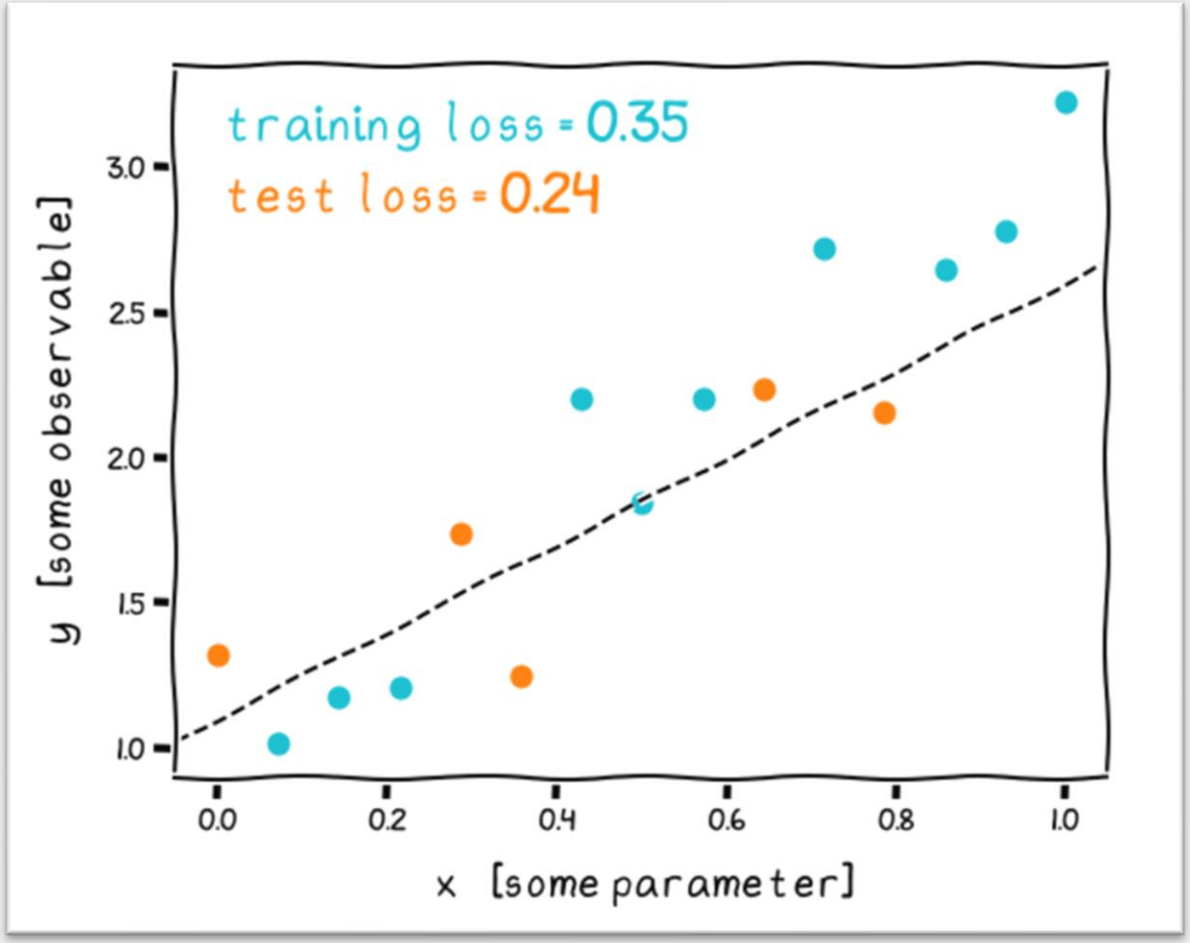

training set
test set
y [some observable]
x [some parameter]
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0


training loss = 0.17
test loss = 0.37
y [some observable]
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0
x [some parameter]


training loss = 0.35
test loss = 0.24
y [some observable]
x [some parameter]
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0


training loss = 0.07
test loss = 2.78
y [some observable]
x [some parameter]
3.0
2.5
2.0
1.5
1.0
0.0
0.2
0.4
0.6
0.8
1.0

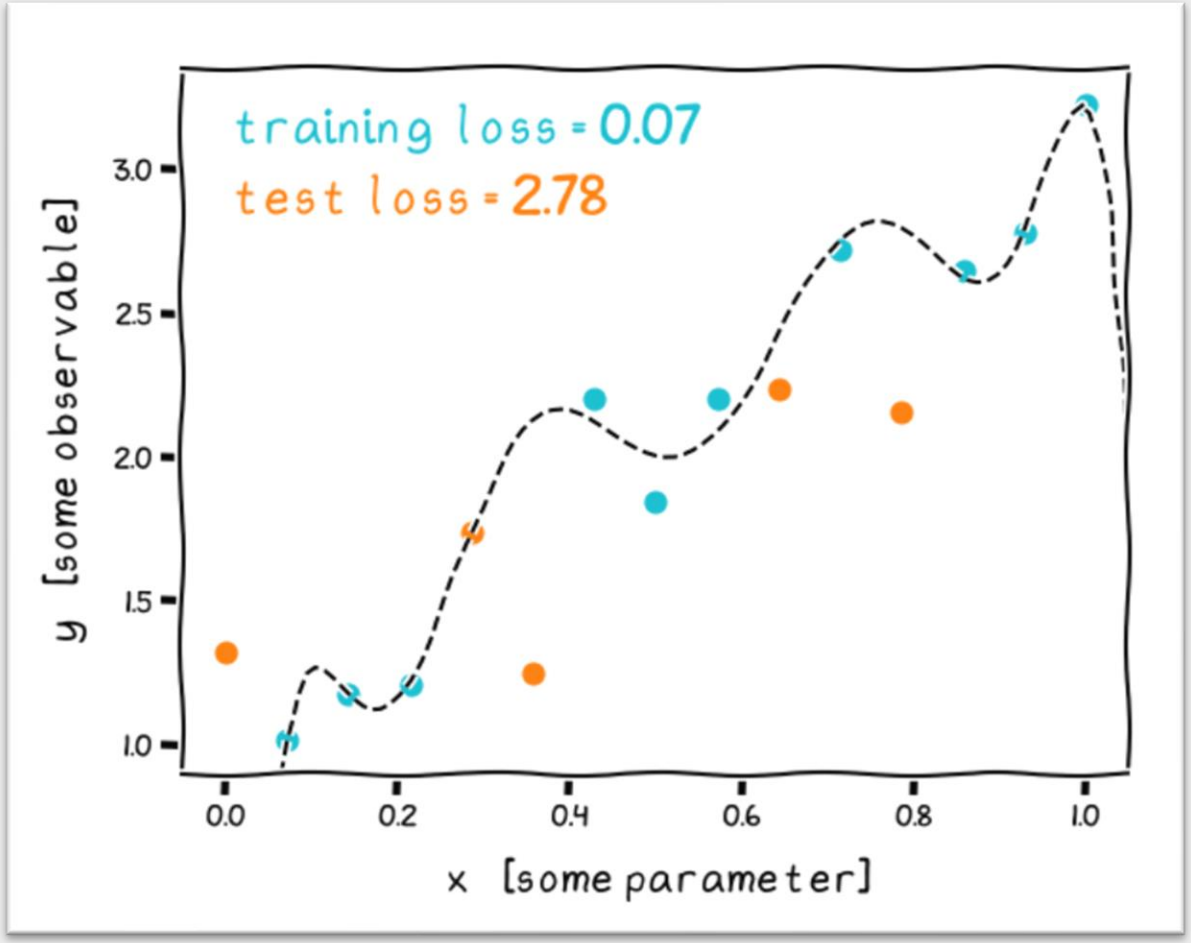
underfitting



training loss > test loss

or training loss is high / not convergent

overfitting



test loss > training loss

5

Evaluate the model

`sklearn.metrics`

loss functions
accuracy scores
precision/recall
confusion matrices

Goodness-of-fit, aka "loss"

The loss function can be anything you like

- Fitting a curve: χ^2, reduced χ^2, RMS loss
- Classification: binary cross-entropy
- More advanced techniques: regularization? Bayesian? Probabilities?

This is also done during training

More on gradient descent + parameter optimization next week

What data do we evaluate the model with?

5

Evaluate the model

cheatsheet

`sklearn.metrics`

loss functions
accuracy scores
precision/recall
confusion matrices

Training your model	training loss
Deciding between models	validation loss
Presenting your model	validation loss or even better, a new test set loss

What data do we evaluate the model with?

6 Final considerations

What are the limitations of your model?

- What was in your training set?
- What was not included?
- What was under / over-represented?
- When would your assumptions break down?

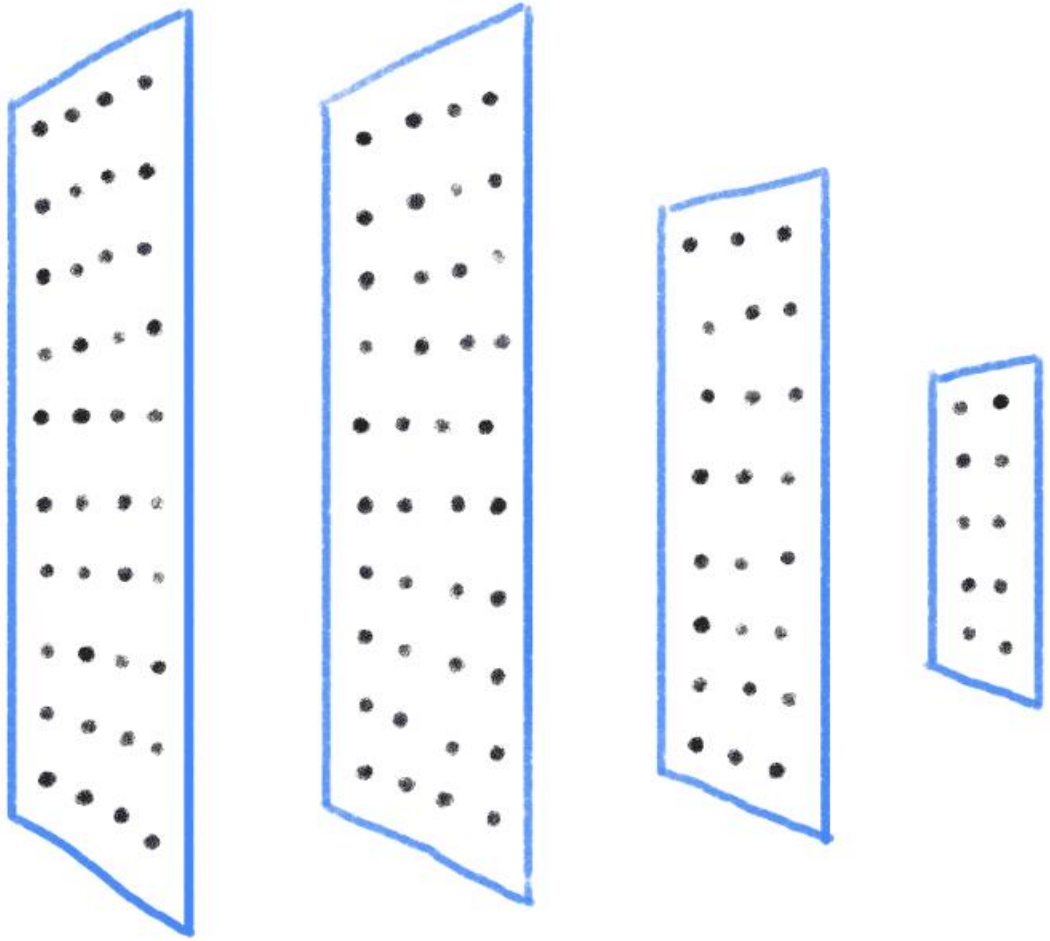
Do not extrapolate!

CAT

(LABELED PHOTOS)

DOG

is this a cat?

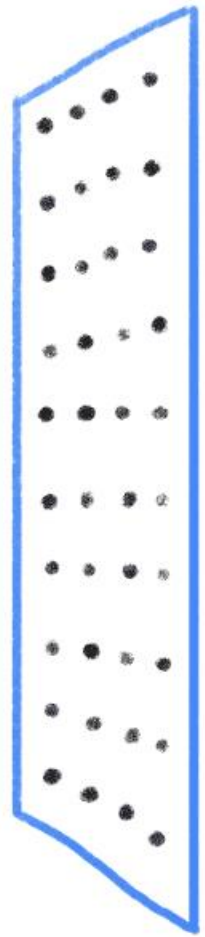


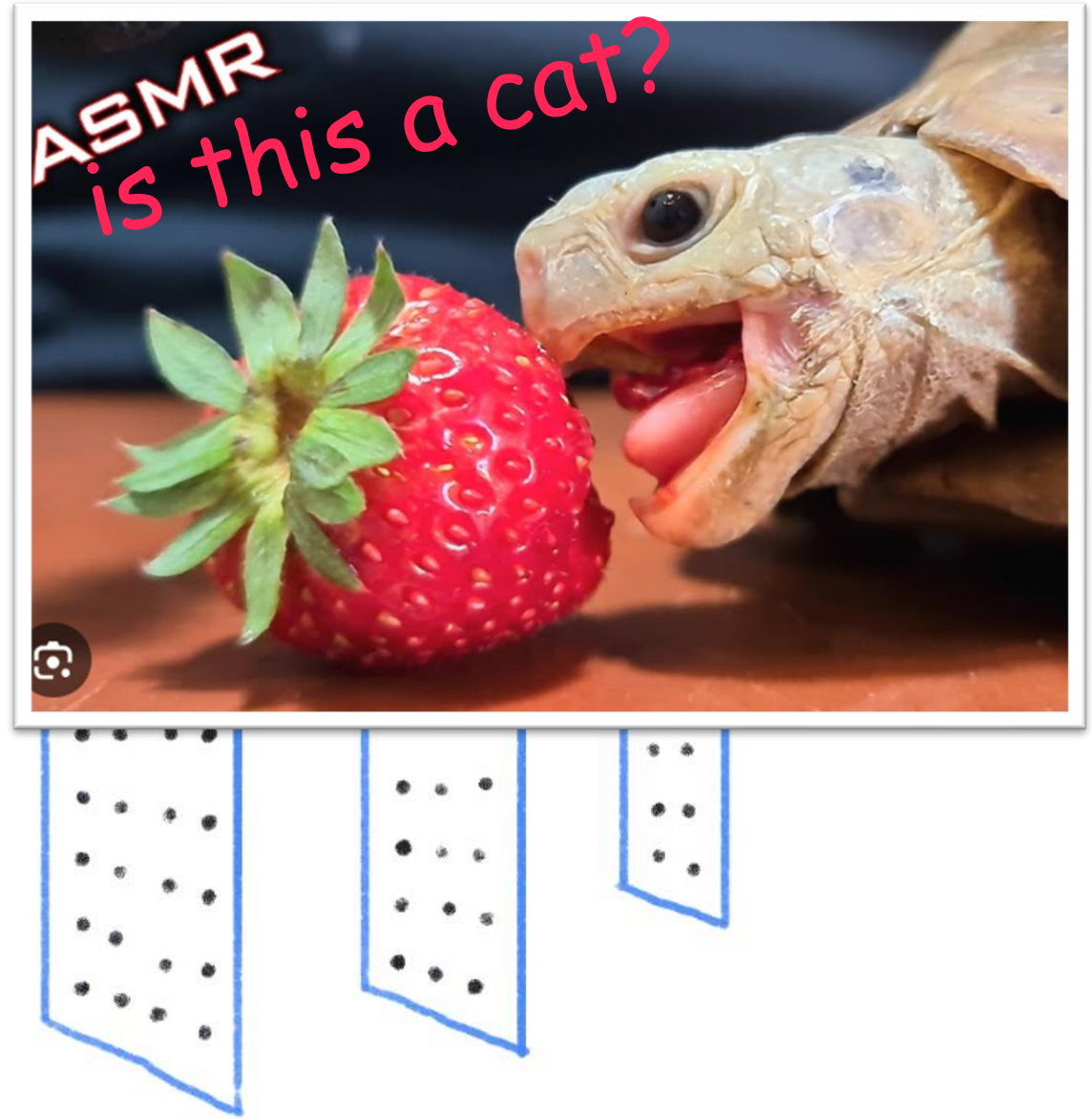
OUTPUT

CAT

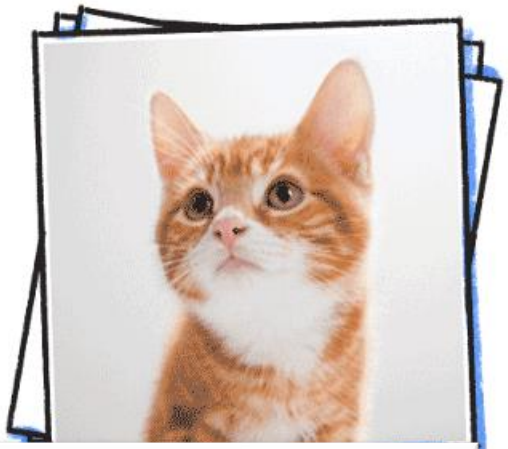
(LABELED PHOTOS)

DOG





CAT



is this a cat?

7 Next steps

Congratulations, you have a model!

- Now you have a new hypothesis – test it
- How does it fit in with other models / what we already know?
- What have you learnt?
- Did you get any cool scientific results? (write a paper)
- Did you provide a cool new dataset for people? (write a paper)

Write a paper, put your Code on GitHub, make your data public :)

1. Form a hypothesis
2. Collect data
3. Split the data
4. Train your model
5. Evaluate your model
6. Think about your model
7. Think of the next project

scientific method
In a nutshell
(ML edition)

Lessons to remember!

1. Always look at your data before doing things
2. Never tell people your model is awesome

   using training set metrics
3. Never extrapolate! and don't trust people who do
4. Scikit-learn is your best friend

Types of Machine Learning

Types of Machine Learning

supervised

unsupervised

contrastive

active

reinforcement

semi-supervised

self-supervised

Types of Machine Learning

supervised

unsupervised

contrastive

active

semi-supervised

reinforcement

self-supervised

Types of Machine Learning

supervised

you know the
right answer
(for some of your data)

labelled data

unsupervised

you don't know the
right answer

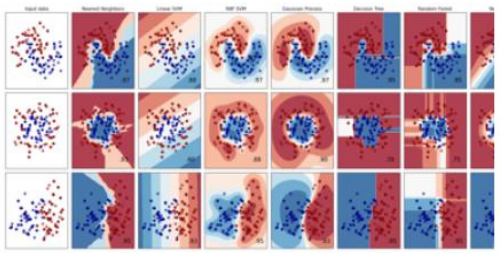
unlabelled data

supervised

Classification

Identifying which category an object belongs to.

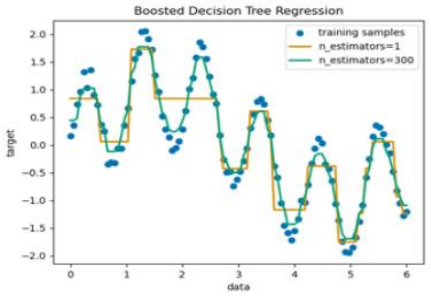
Applications: Spam detection, image recognition.
Algorithms: Gradient boosting, nearest neighbors, random forest, logistic regression, and more...



Regression

Predicting a continuous-valued attribute associated with an object.

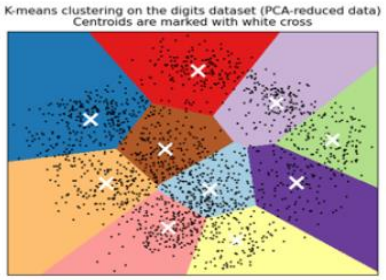
Applications: Drug response, Stock prices.
Algorithms: Gradient boosting, nearest neighbors, random forest, ridge, and more...



Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, HDBSCAN, hierarchical clustering, and more...

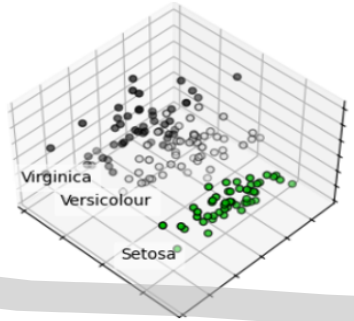


unsupervised

Dimensionality reduction

Reducing the number of random variables to consider.

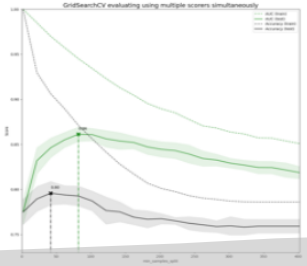
Applications: Visualization, Increased efficiency
Algorithms: PCA, feature selection, non-negative matrix factorization, and more...



Model selection

Comparing, validating and choosing parameters and models.

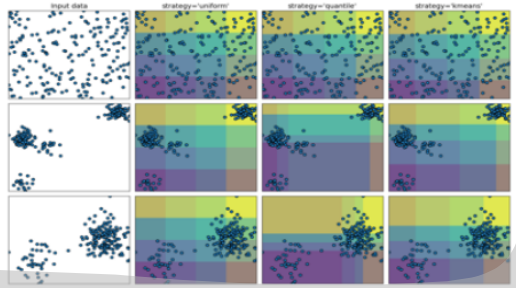
Applications: Improved accuracy via parameter tuning
Algorithms: grid search, cross validation, metrics, and more...



Preprocessing

Feature extraction and normalization.

Applications: Transforming input data such as text for use with machine learning algorithms.
Algorithms: preprocessing, feature extraction, and more...



not ML (this take is a bit hot and probably untrue)

unsupervised learning


no need for train/test split
try lots of models
explore your data
see what works
unsupervised learning

data exploration

what type of data am I working with?

how many types of penguins are there?

finding outliers

which points are unusual? why / how?

can I remove artifacts from my galaxy catalog?

unsupervised learning

similarity search

which things are similar, and why?

which TV shows should I watch if I like Bojack Horseman?

finding patterns

what are the patterns or groups in my data?

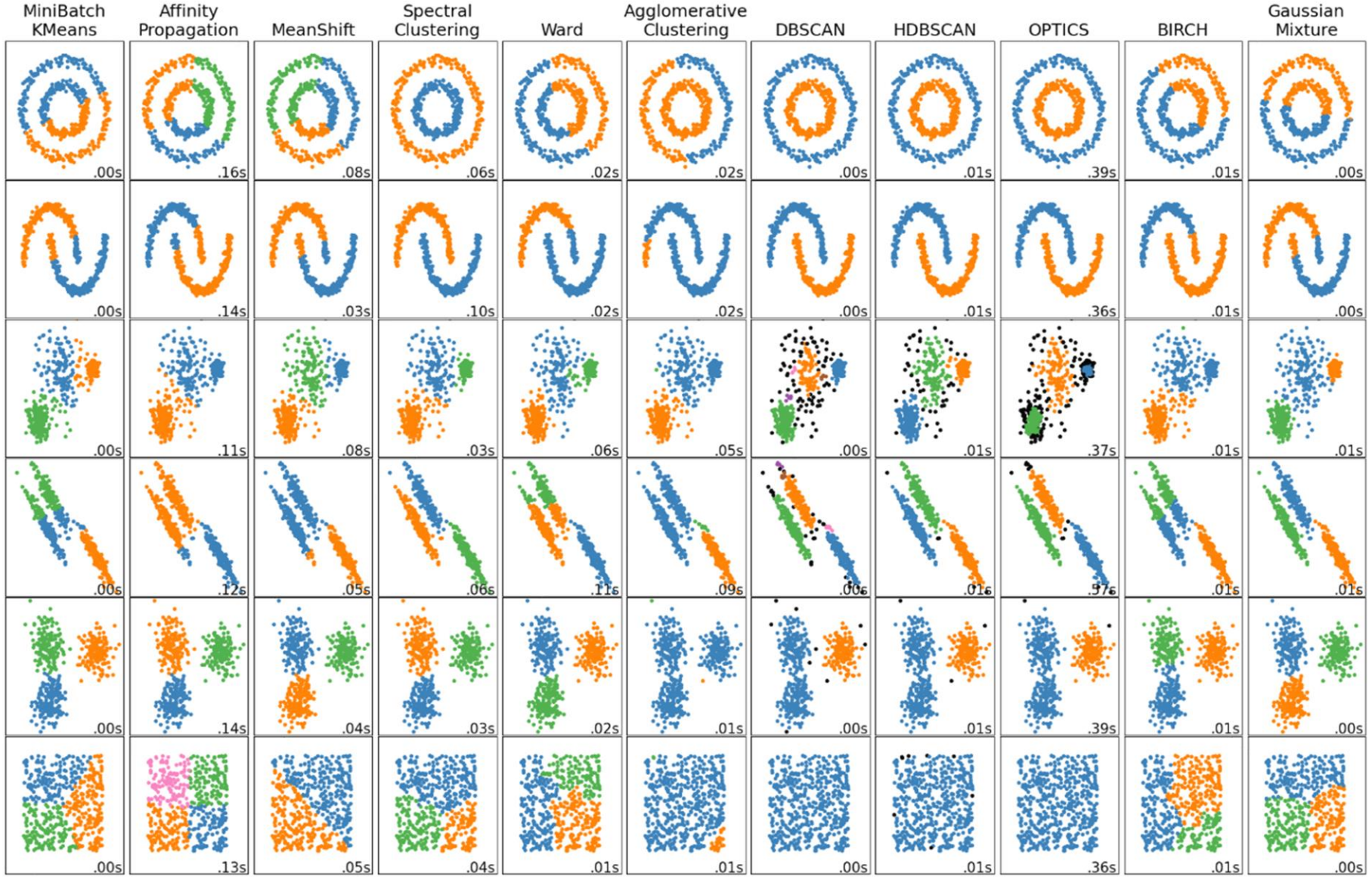
how do I detect REM sleep from EEG?

unsupervised learning

look at your data

choose an algorithm

2.3.1. Overview of clustering methods

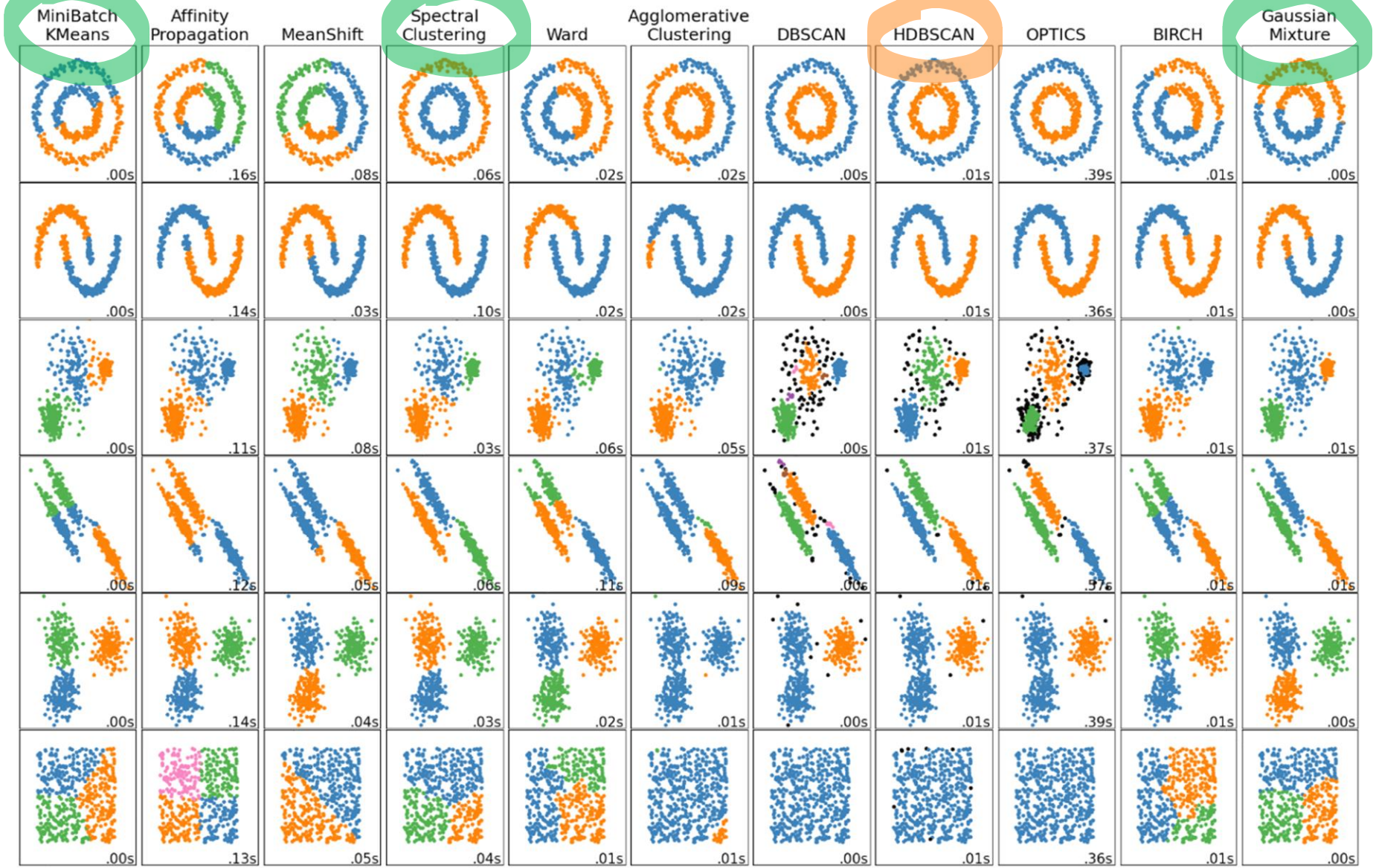


A comparison of the clustering algorithms in scikit-learn

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large `n_samples`, medium `n_clusters` with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters, inductive	Distances between points
Affinity propagation	damping, sample preference	Not scalable with n_samples	Many clusters, uneven cluster size, non-flat geometry, inductive	Graph distance (e.g. nearest-neighbor graph)
Mean-shift	bandwidth	Not scalable with `n_samples`	Many clusters, uneven cluster size, non-flat geometry, inductive	Distances between points
Spectral clustering	number of clusters	Medium `n_samples`, small `n_clusters`	Few clusters, even cluster size, non-flat geometry, transductive	Graph distance (e.g. nearest-neighbor graph)
Ward hierarchical clustering	number of clusters or distance threshold	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, transductive	Distances between points
Agglomerative clustering	number of clusters or distance threshold, linkage type, distance	Large `n_samples` and `n_clusters`	Many clusters, possibly connectivity constraints, non Euclidean distances, transductive	Any pairwise distance
DBSCAN	neighborhood size	Very large `n_samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes, outlier removal, transductive	Distances between nearest points
HDBSCAN	minimum cluster membership, minimum point neighbors	large `n_samples`, medium `n_clusters`	Non-flat geometry, uneven cluster sizes, outlier removal, transductive, hierarchical, variable cluster density	Distances between nearest points
OPTICS	minimum cluster membership	Very large `n_samples`, large `n_clusters`	Non-flat geometry, uneven cluster sizes, variable cluster density, outlier removal, transductive	Distances between points
Gaussian mixtures	many	Not scalable	Flat geometry, good for density estimation, inductive	Mahalanobis distances to centers
BIRCH	branching factor, threshold, optional global clusterer.	Large `n_clusters` and `n_samples`	Large dataset, outlier removal, data reduction, inductive	Euclidean distance between points
Bisecting K-Means	number of clusters	Very large `n_samples`, medium `n_clusters`	General-purpose, even cluster size, flat geometry, no empty clusters, inductive, hierarchical	Distances between points

lesson 2

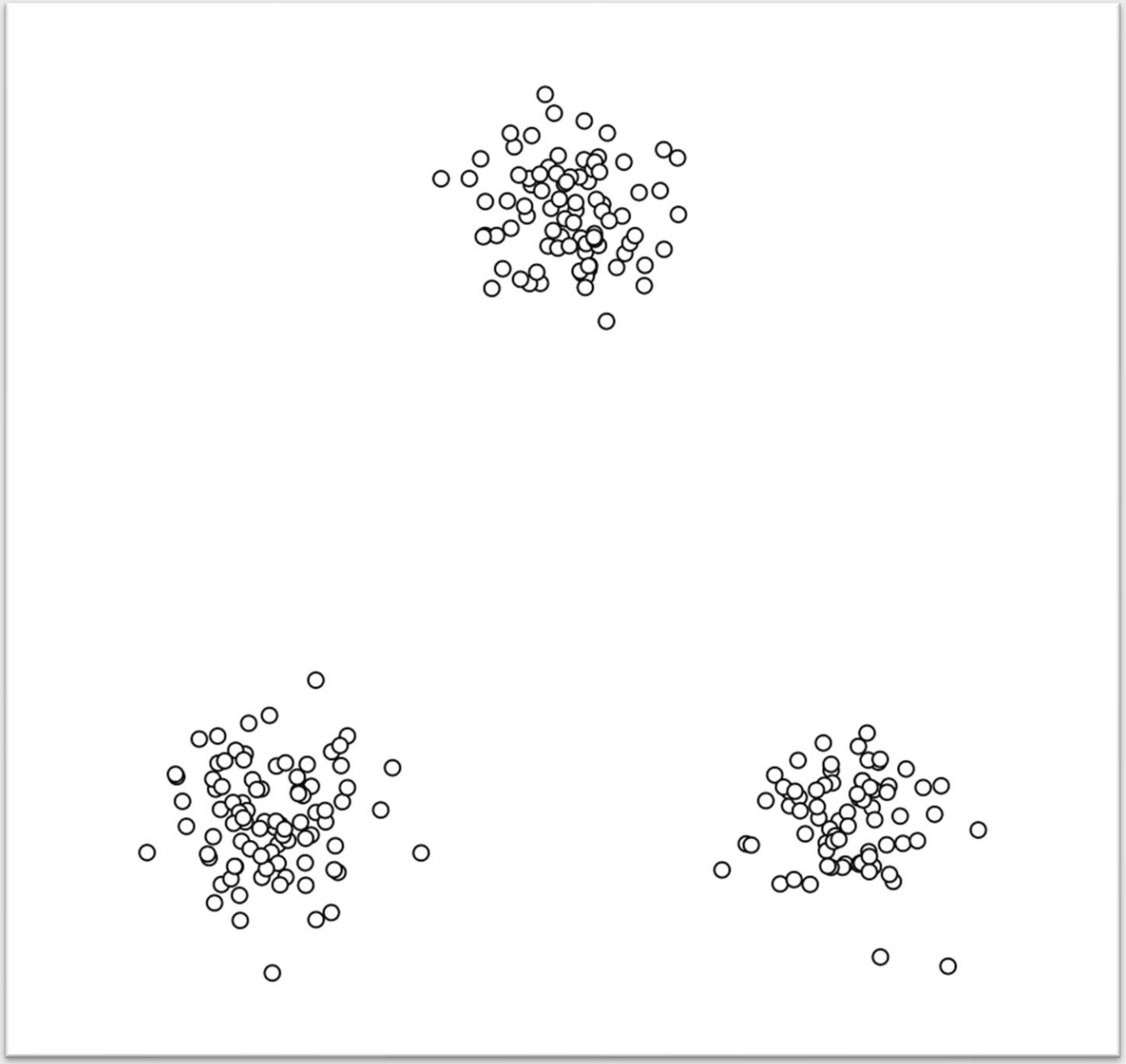
2.3.1. Overview of clustering methods



A comparison of the clustering algorithms in scikit-learn

K-means

How many groups are there?

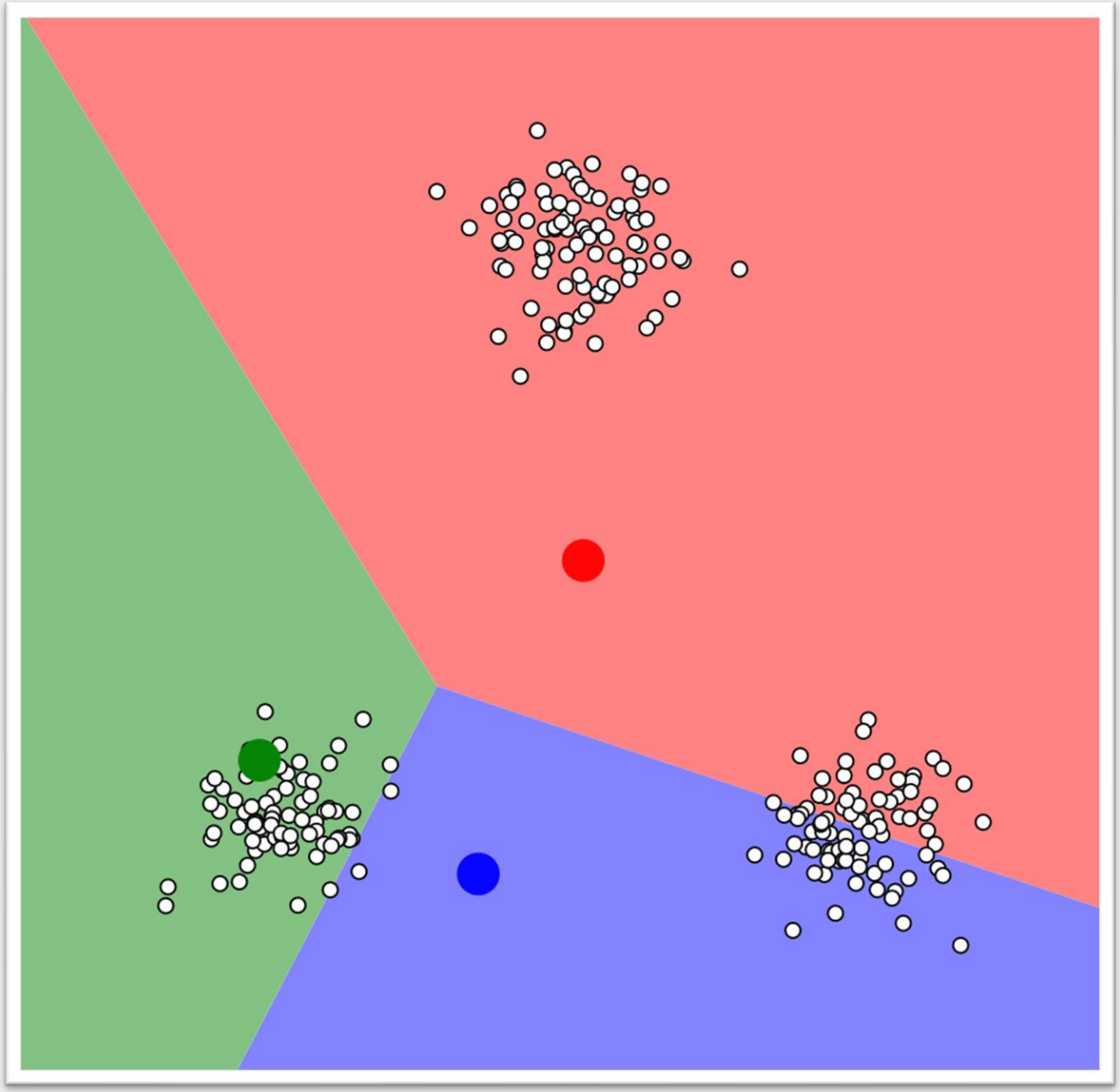


https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

`sklearn.cluster.KMeans`

K-means

Take a guess
on where the cluster
centers are



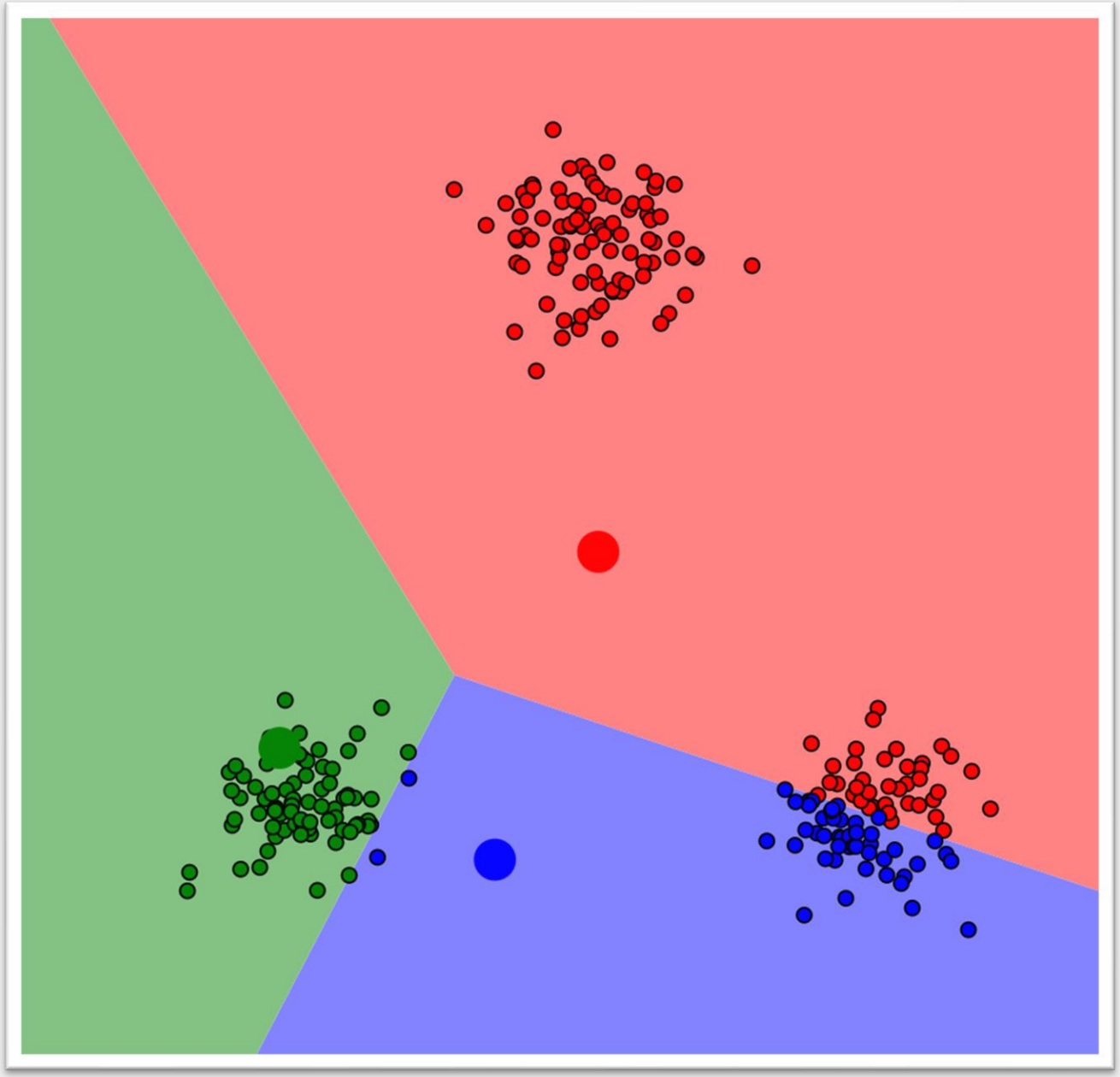
Options

- Random
- User-defined
- Optimal (kmeans++)

`sklearn.cluster`.KMeans

K-means

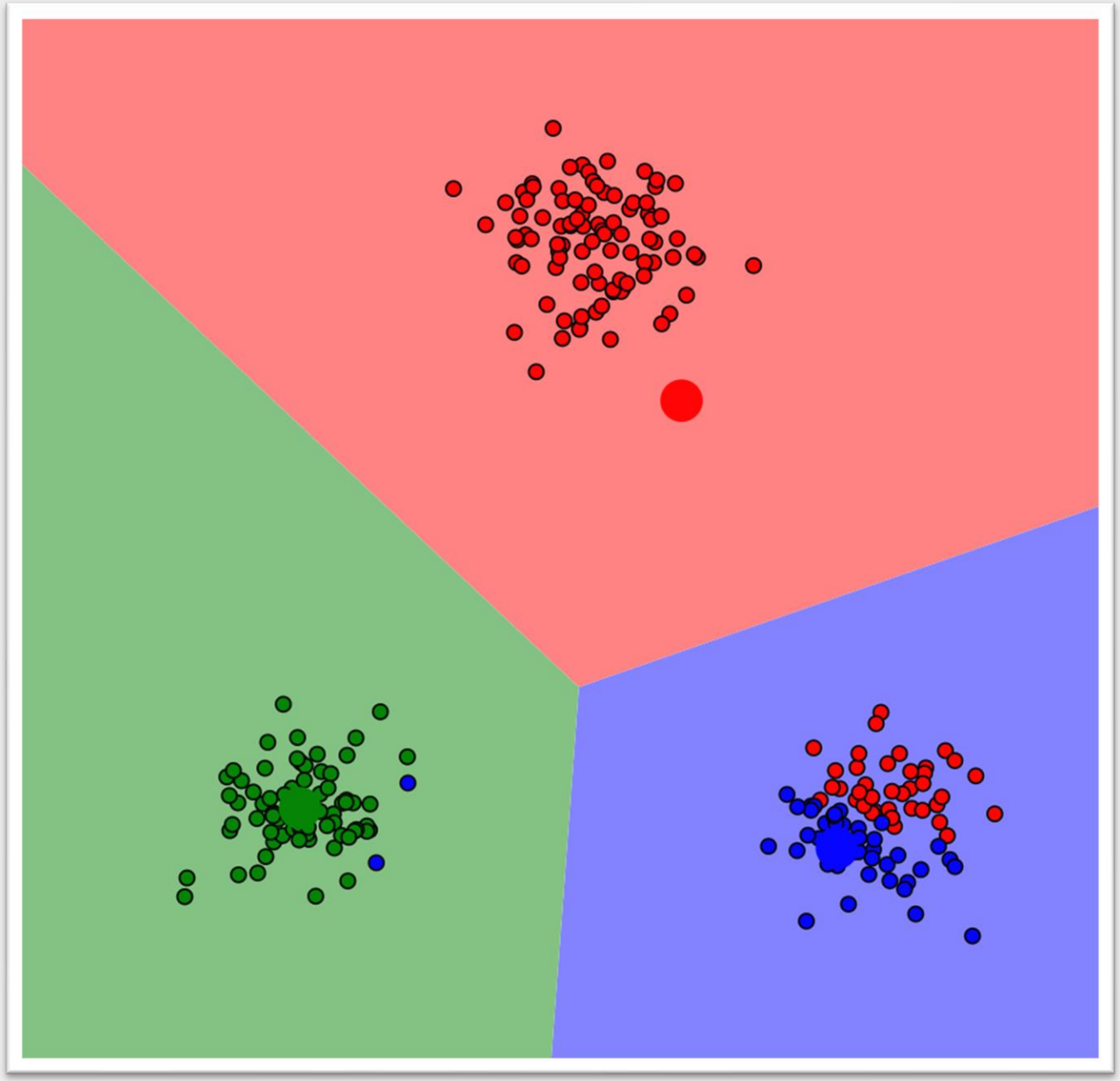
Assign points
based on their
distance to the cluster
center



`sklearn.cluster`.KMeans

K-means

Update centroids
based on the average
position of all
the points in the cluster

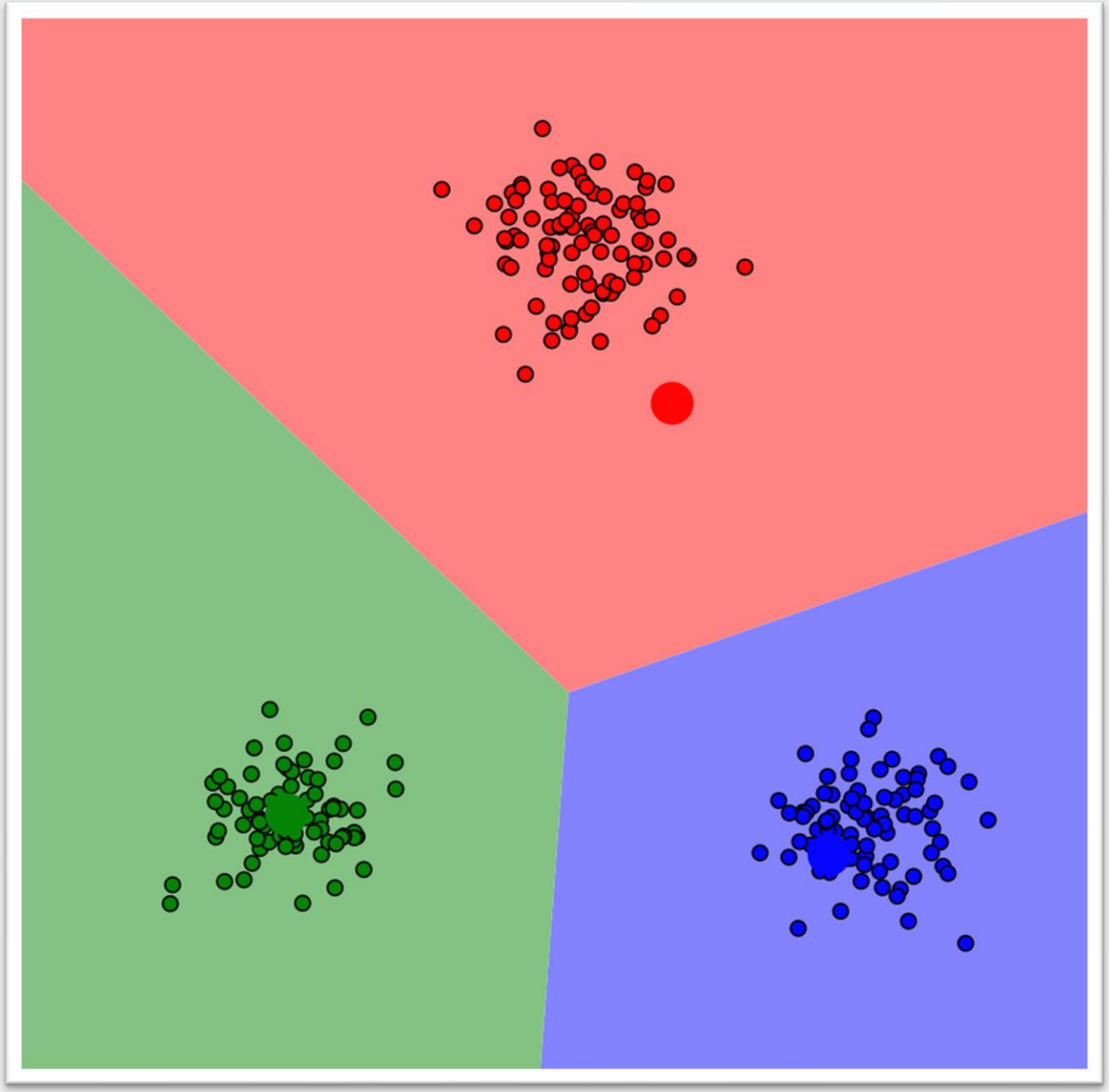


`sklearn.cluster`.KMeans

K-means

Reassign points…

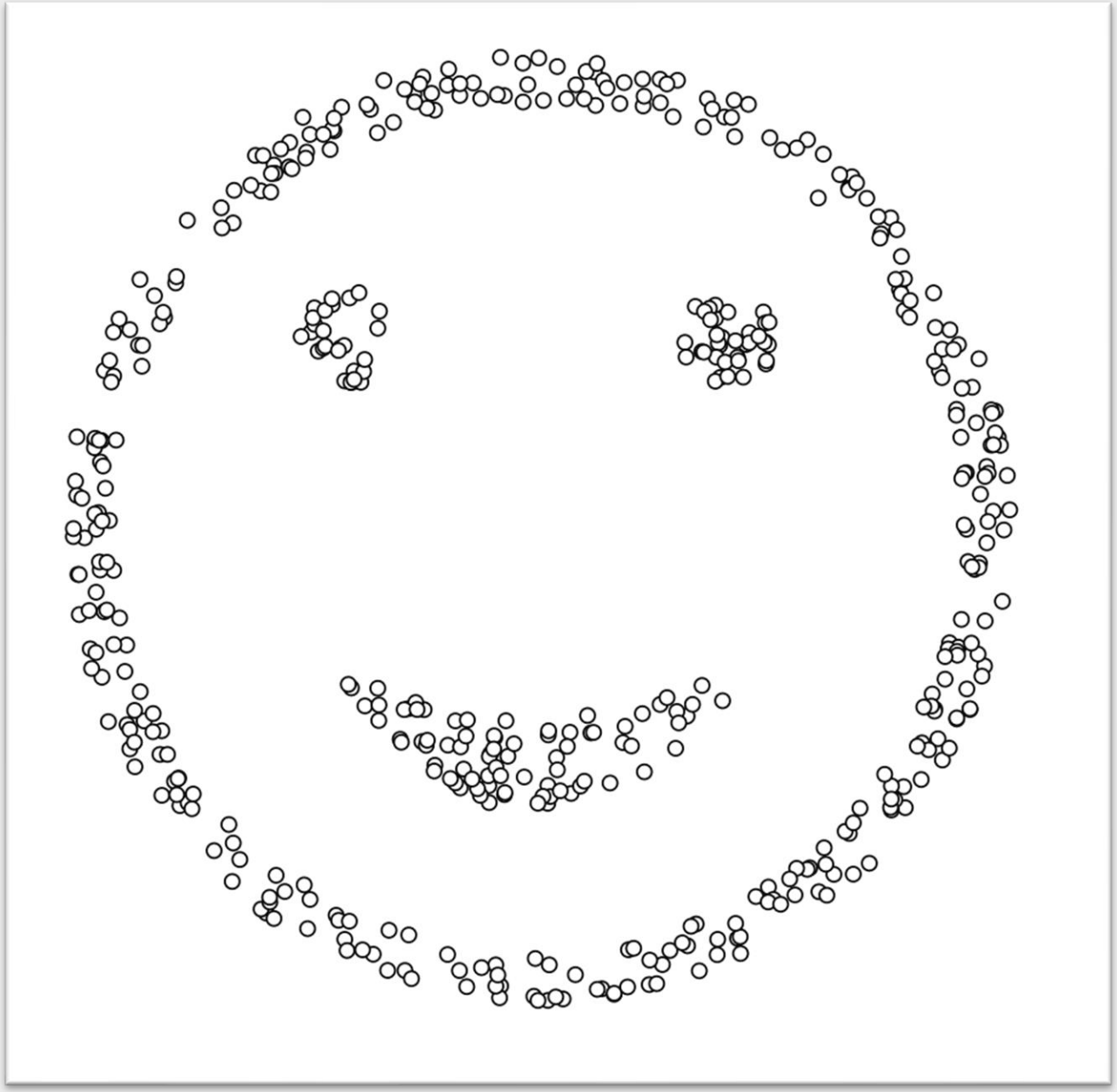
Rinse and repeat N times…



`sklearn.cluster`.KMeans

K-means

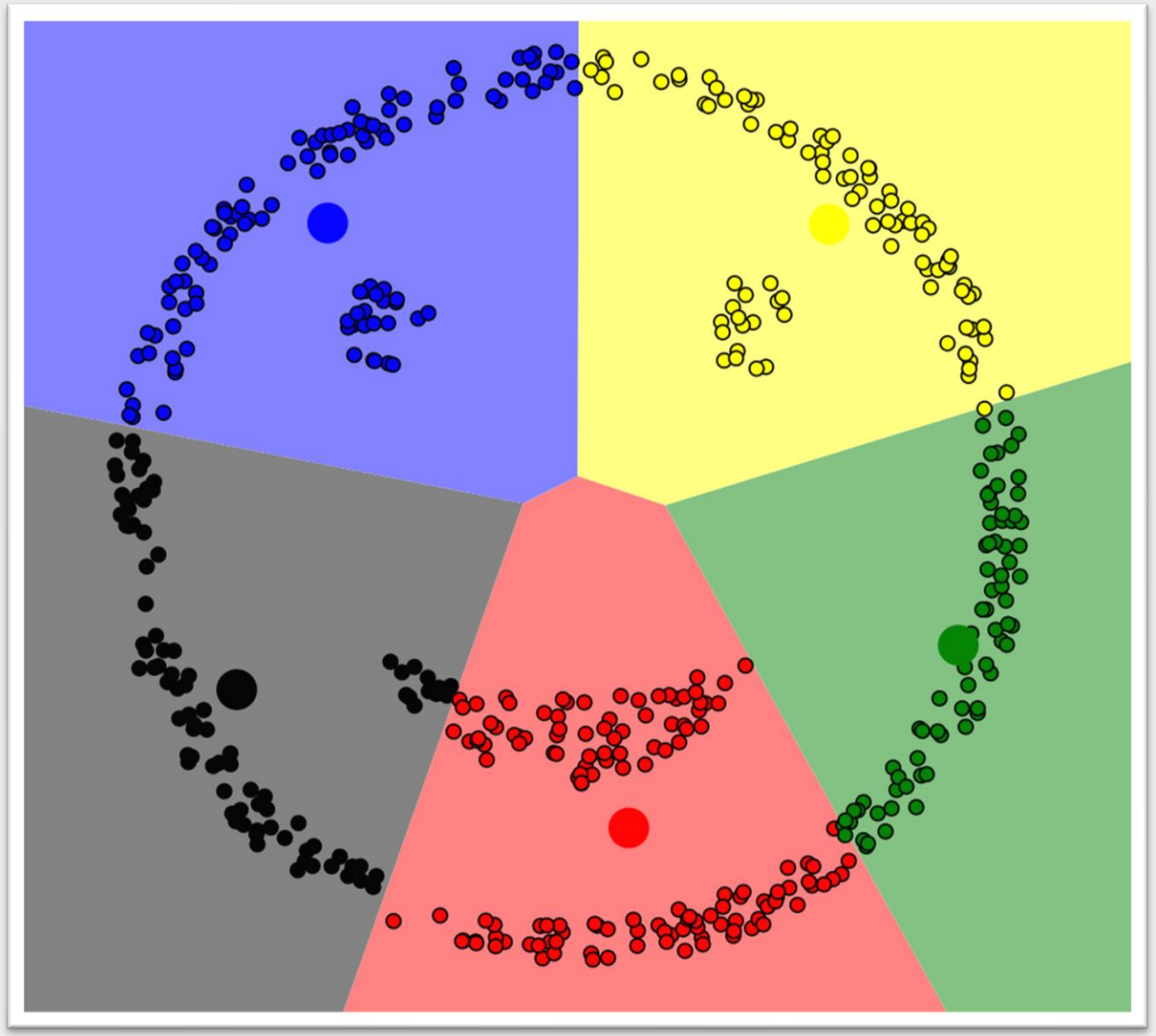
How many groups are there?



`sklearn.cluster.KMeans`

K-means

What about this data?



`sklearn.cluster.KMeans`

K-means

pros

easy

really fast!

easy to interpret

great in many dims

no assumption

of cluster shape**

cons

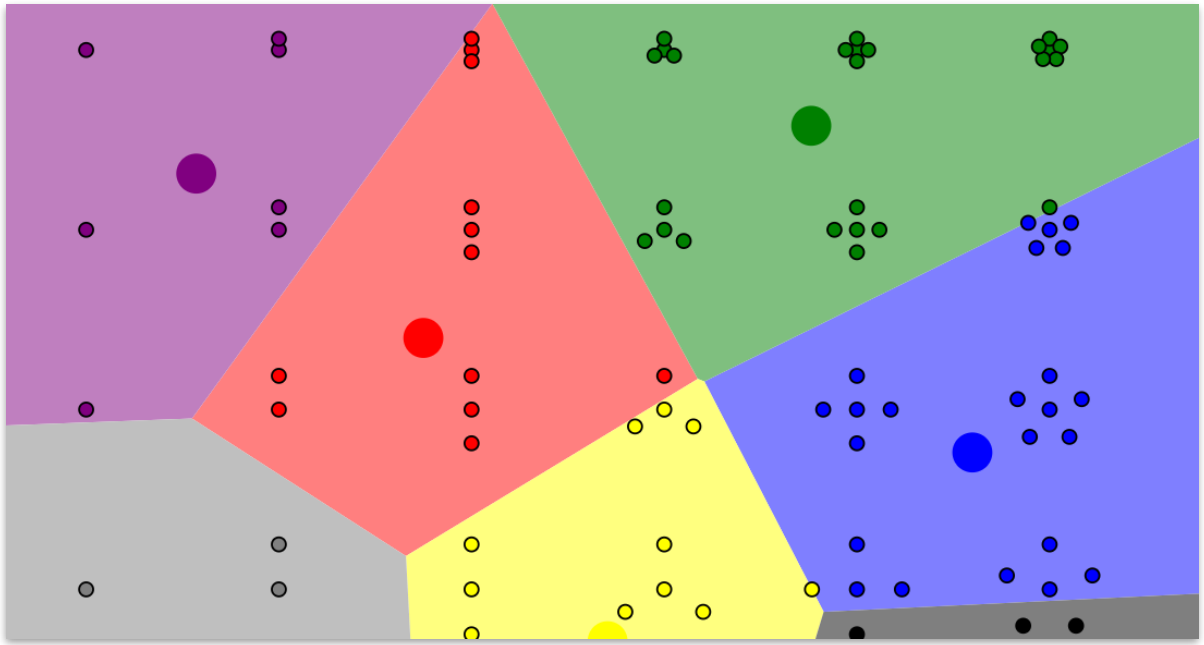
need to decide on k

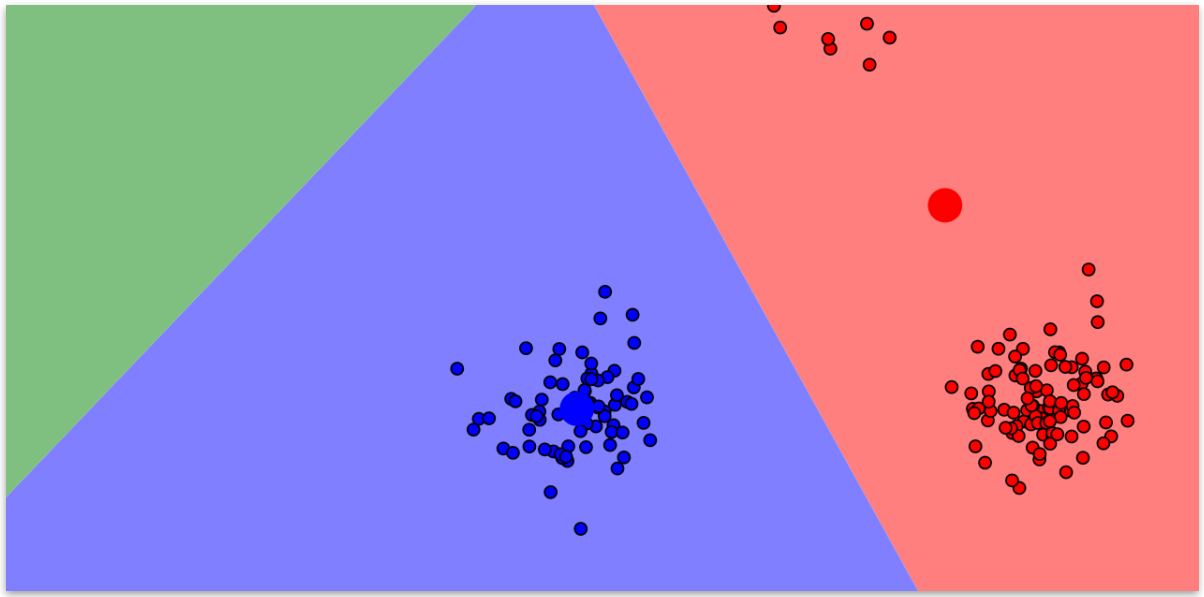
depends on initial values

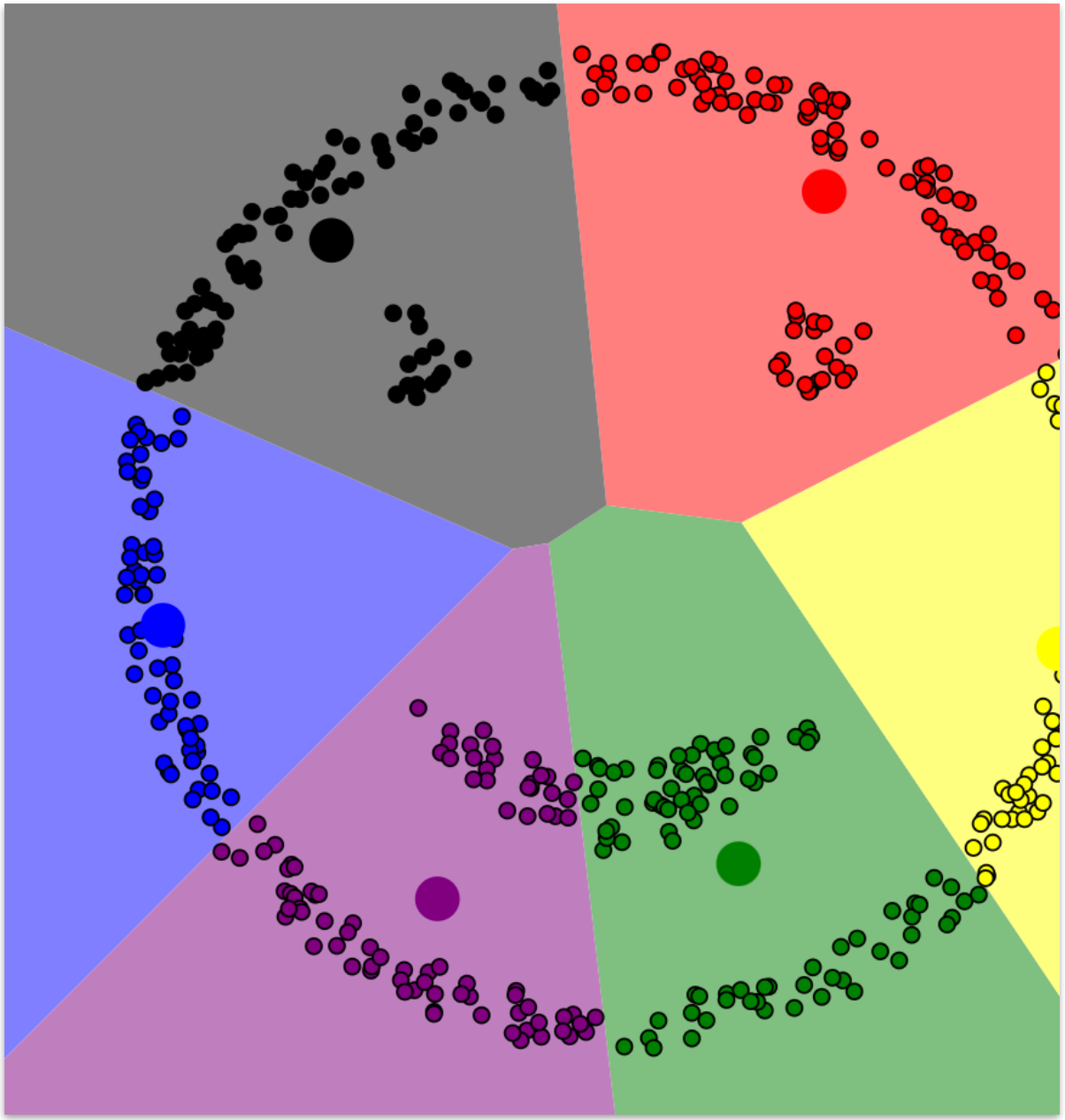
not great for uneven clusters

outliers

complex patterns

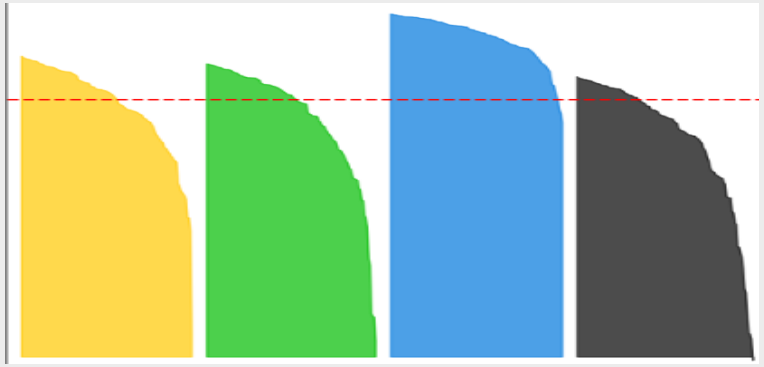






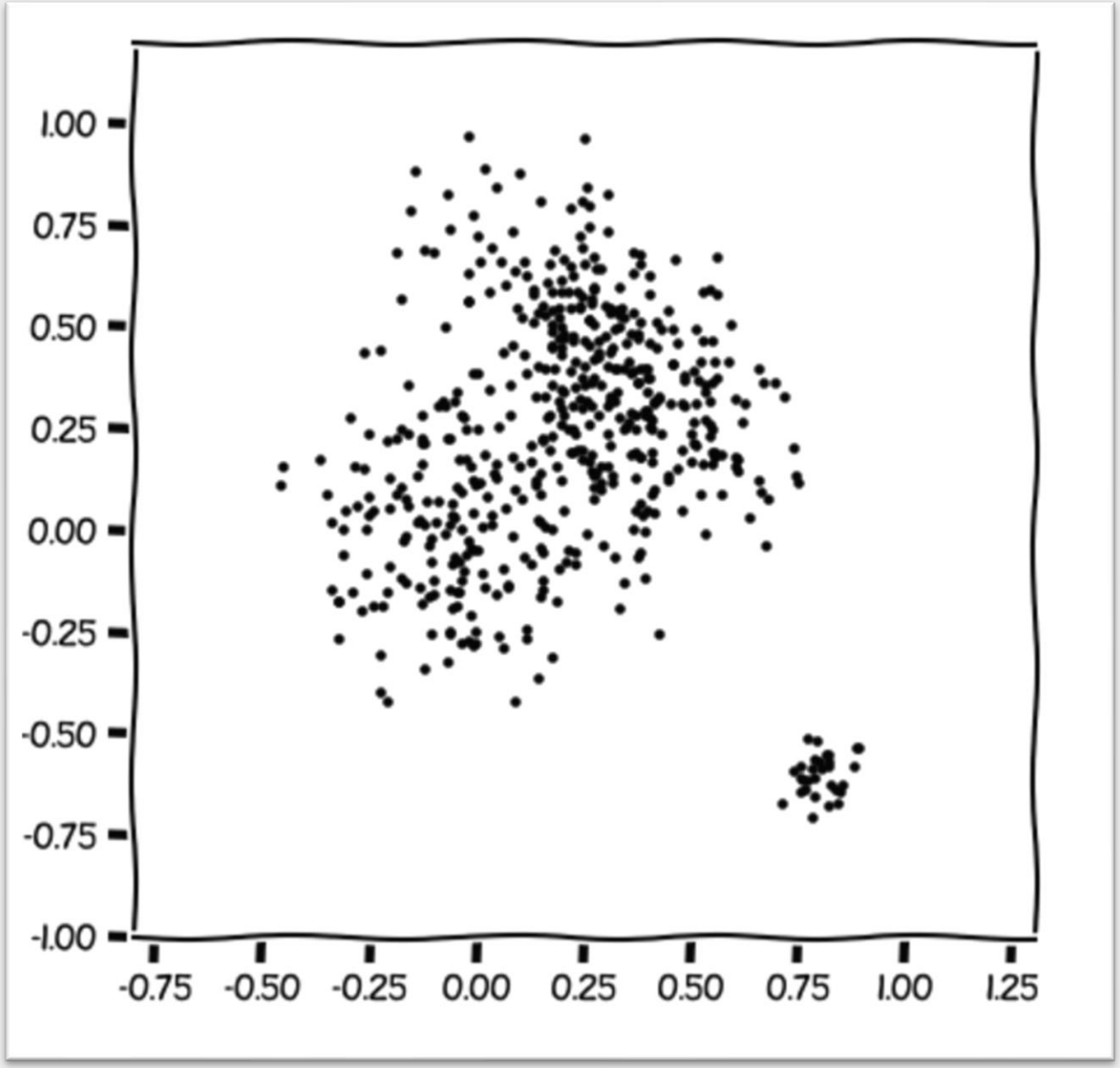
k-means

silhouette score



K-means

How many groups are there?



numpy.random.multivariate_normal

K-means

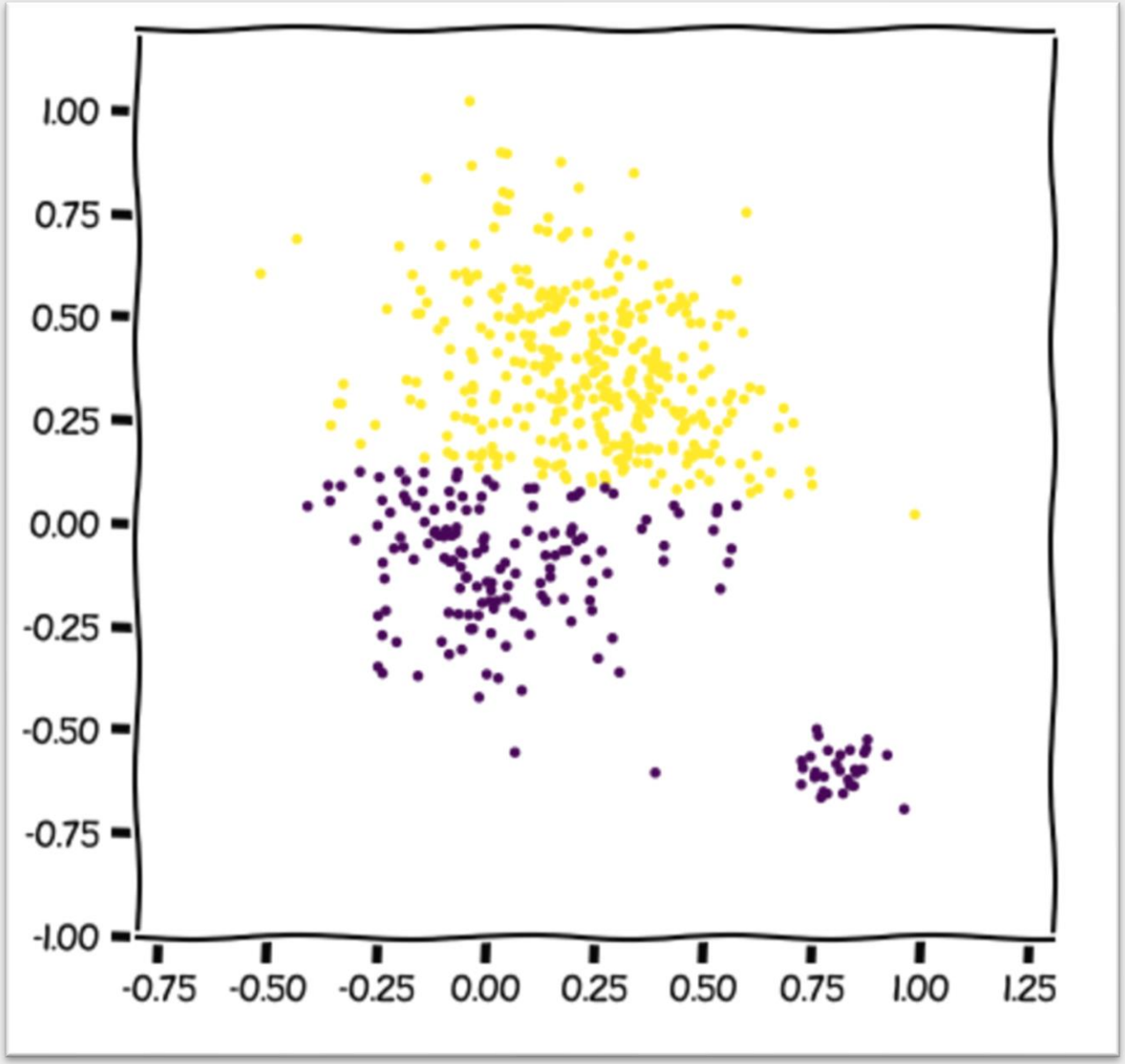
Fit results

Parameters

- Two clusters
- Automatic centroids (kmeans++)

Lesson learnt

Do not blindly trust your kmeans results when making a lesson plan…



numpy.random.multivariate_normal

K-means

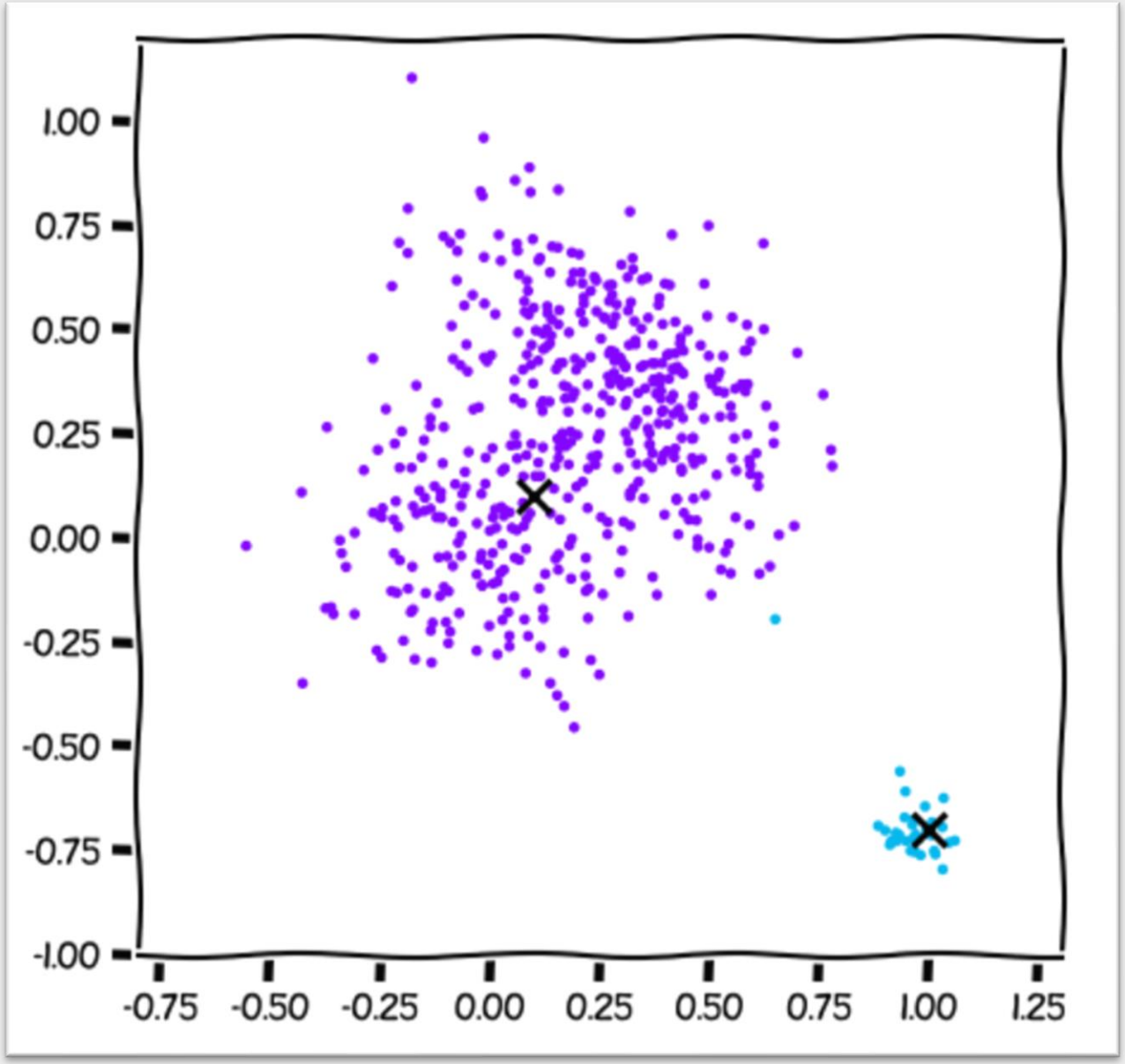
Fit results

Parameters

- Two clusters
- Custom centroids
- Moved the small cluster a little further away to help K-means do its thing

Lesson learnt

It really doesn't like having uneven clusters…

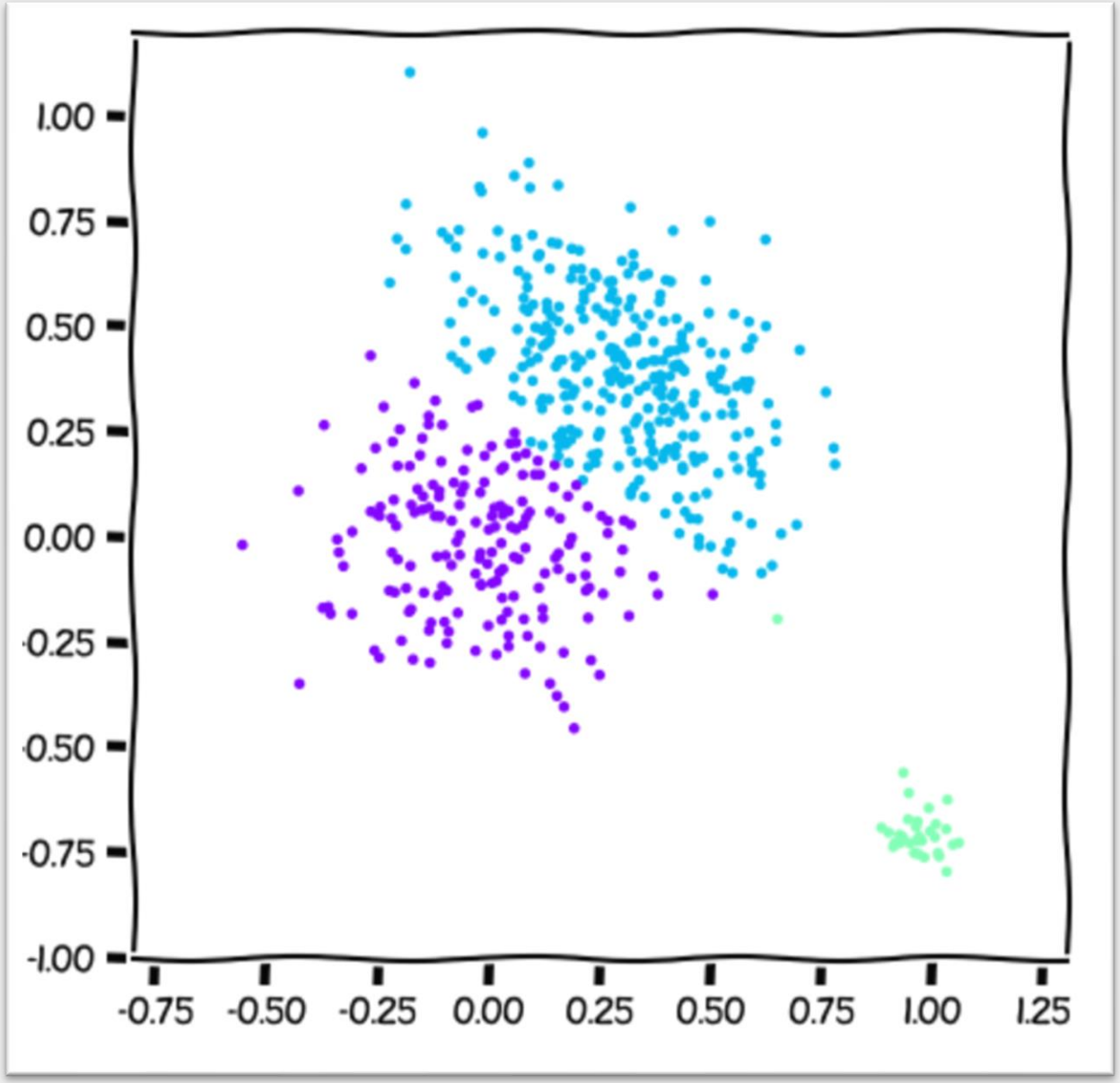


K-means

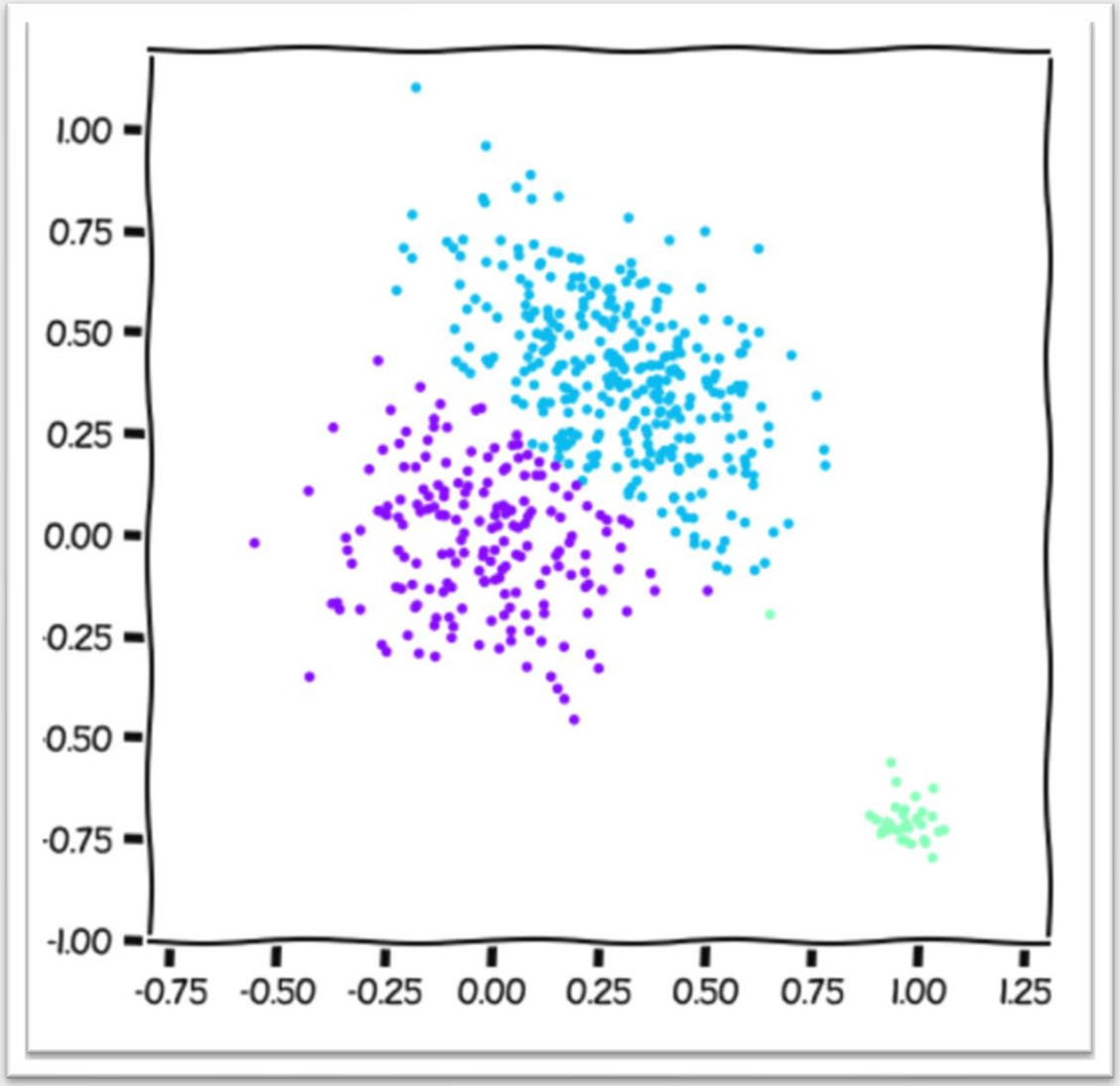
Fit results

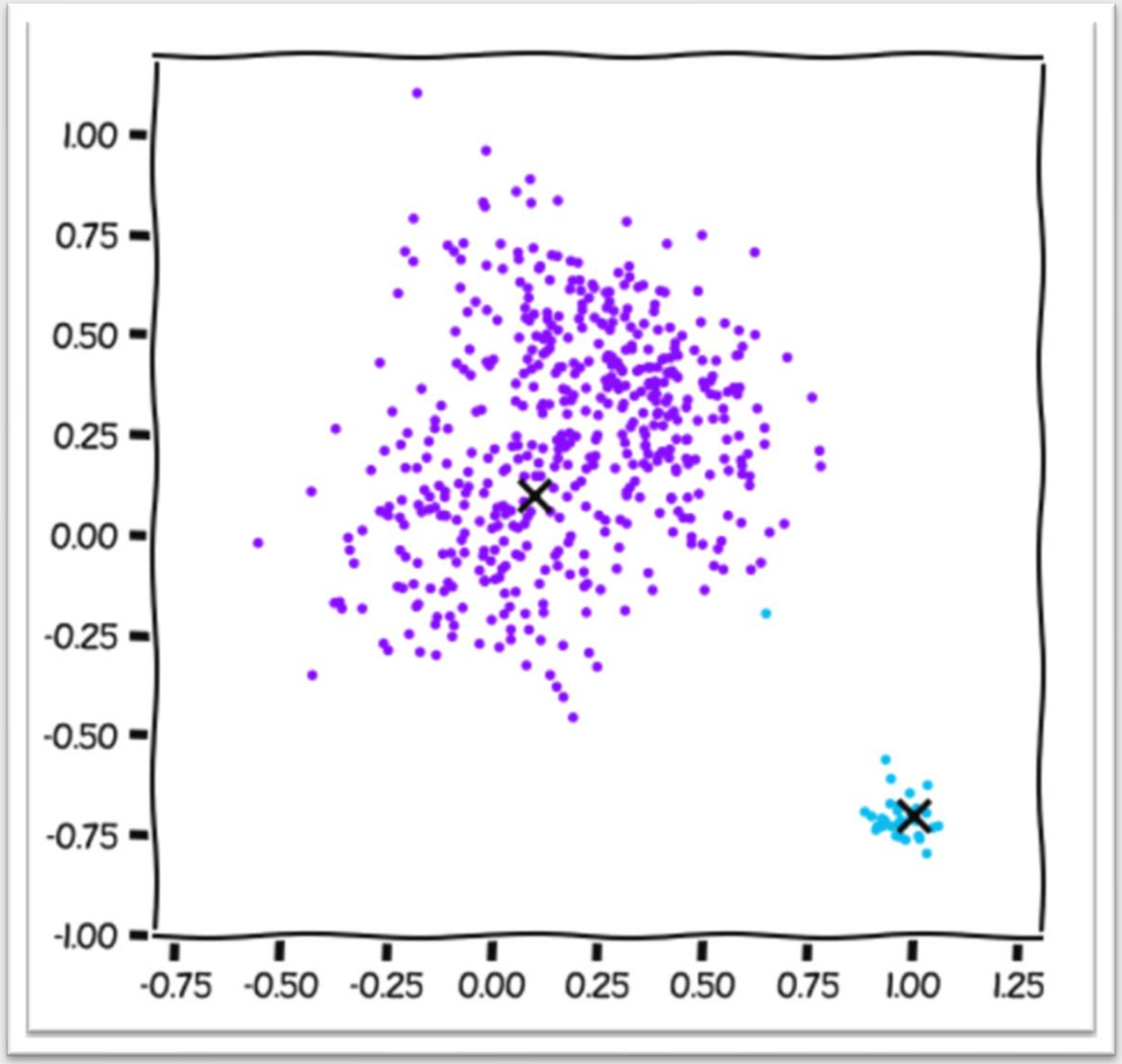
Parameters

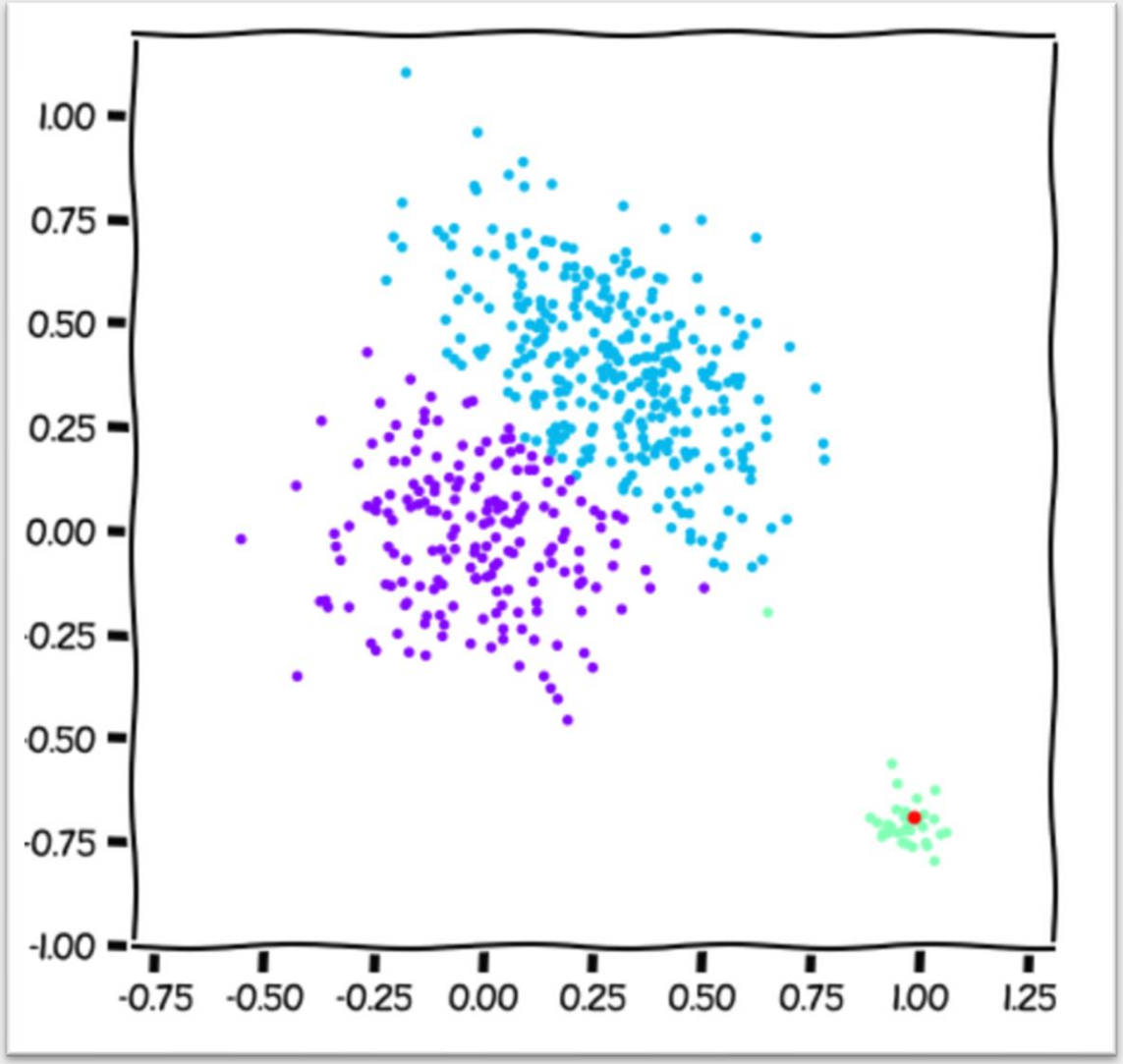
- Three clusters
- kmeans++ centroids



which fit is better?

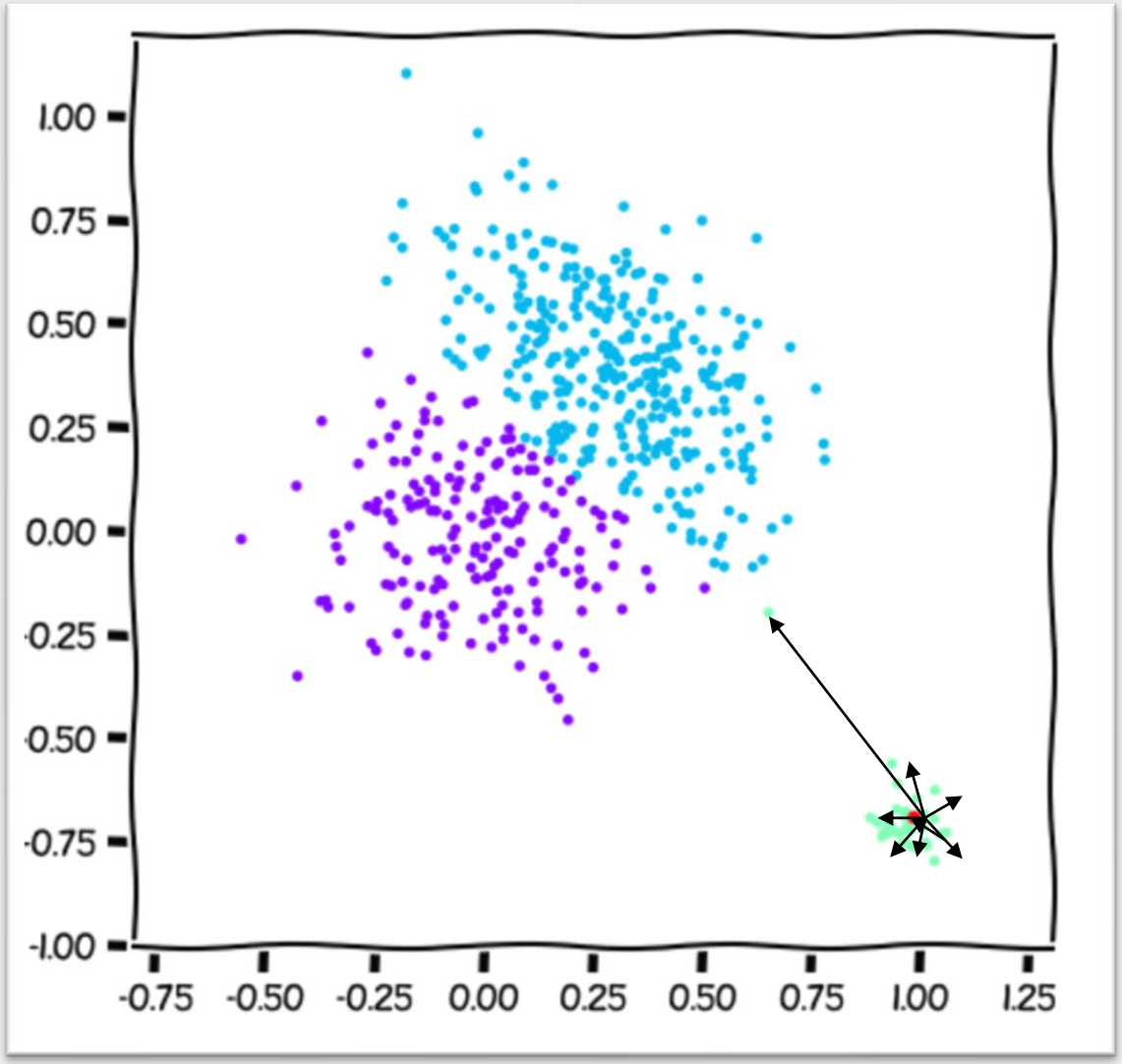






Silhouette score

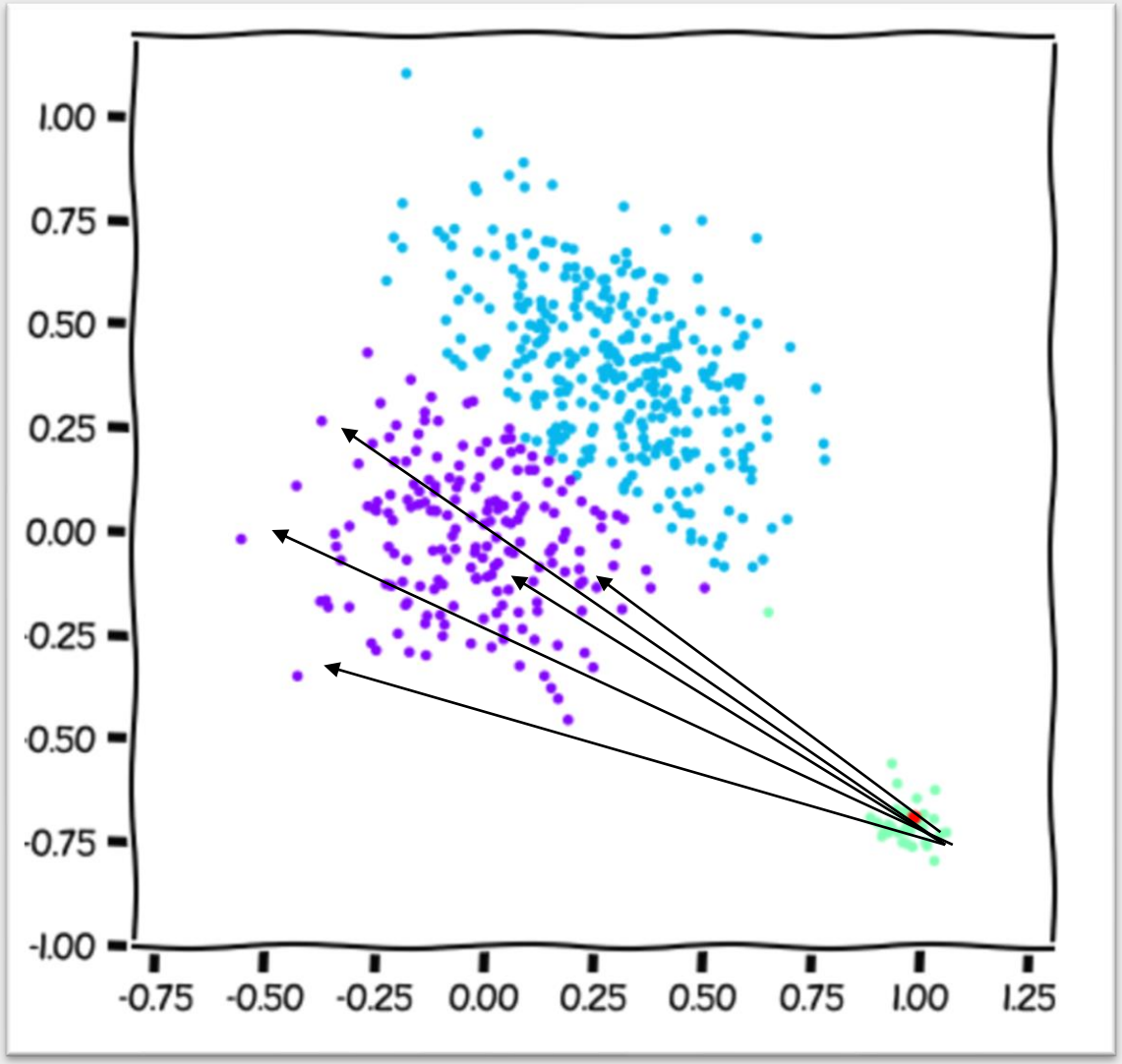
Calculated for each point in each cluster



Silhouette score

Calculated for each point in each cluster

A average distance from the point to each other point in its cluster

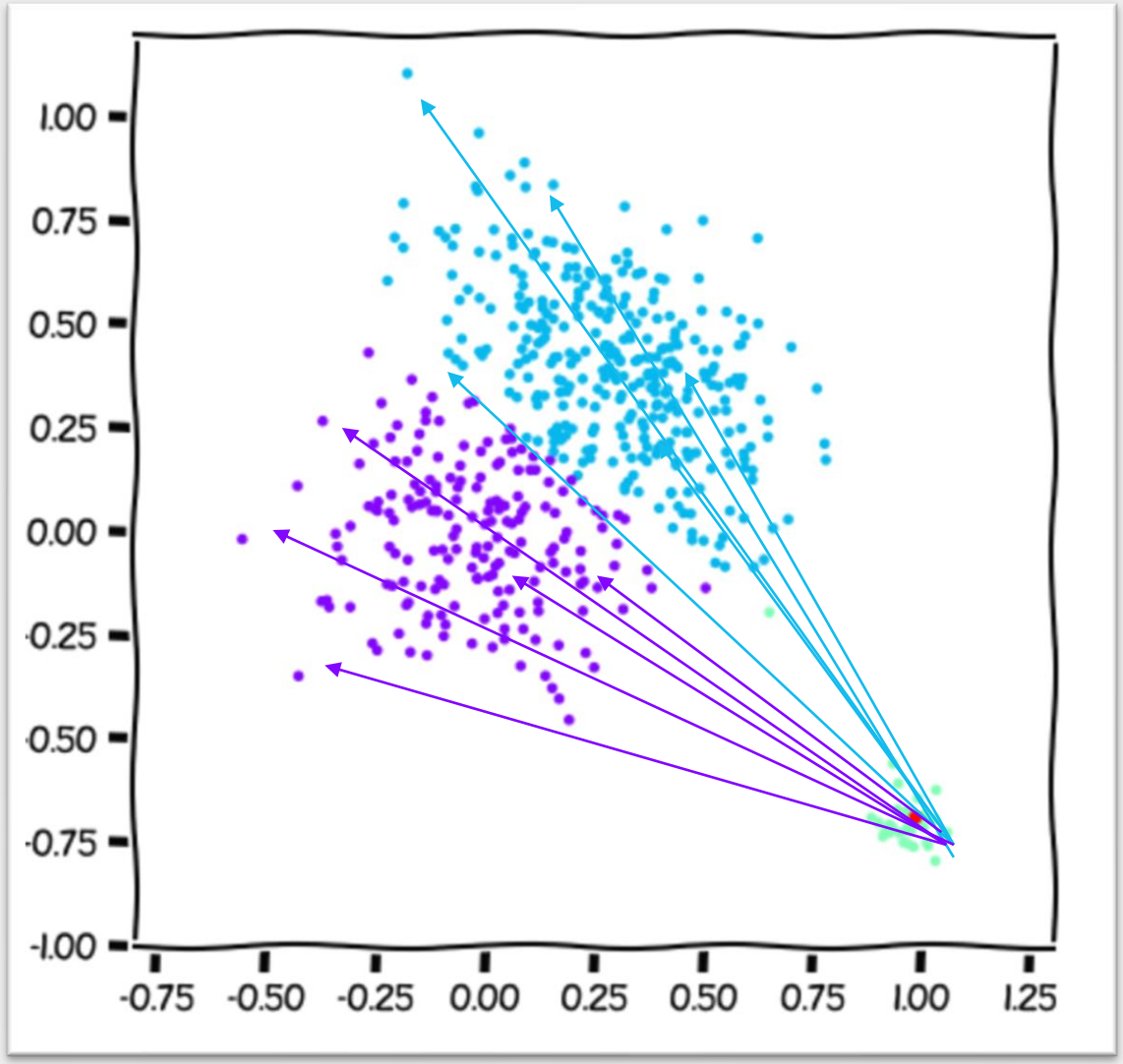


Silhouette score

Calculated for each point in each cluster

A — average distance from the point to each other point in its cluster

B — average distance from the point to each point of another cluster

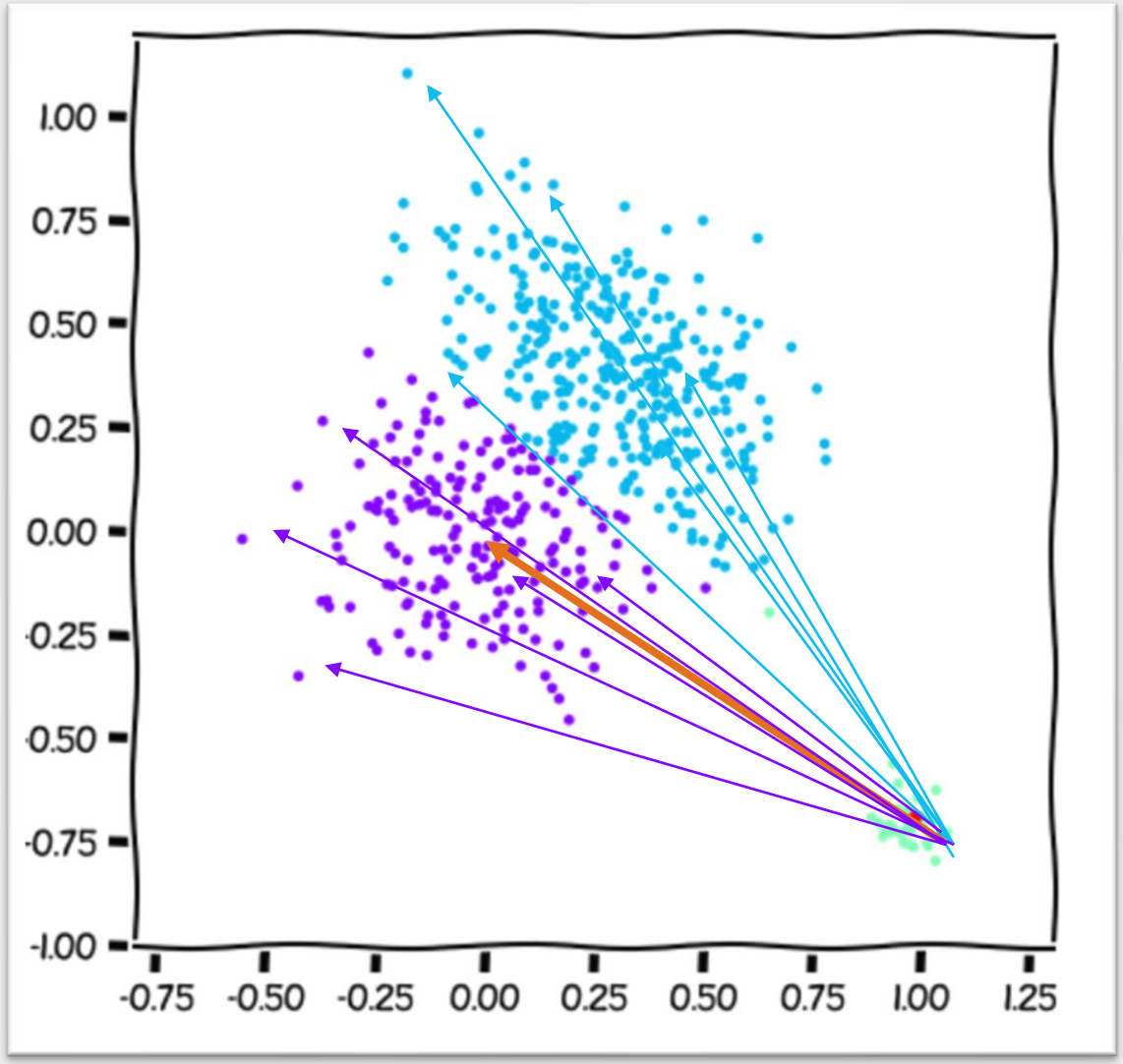


Silhouette score

Calculated for each point in each cluster

A — average distance from the point to each other point in its cluster

B_1, B_2 — average distance from the point to each point of another cluster



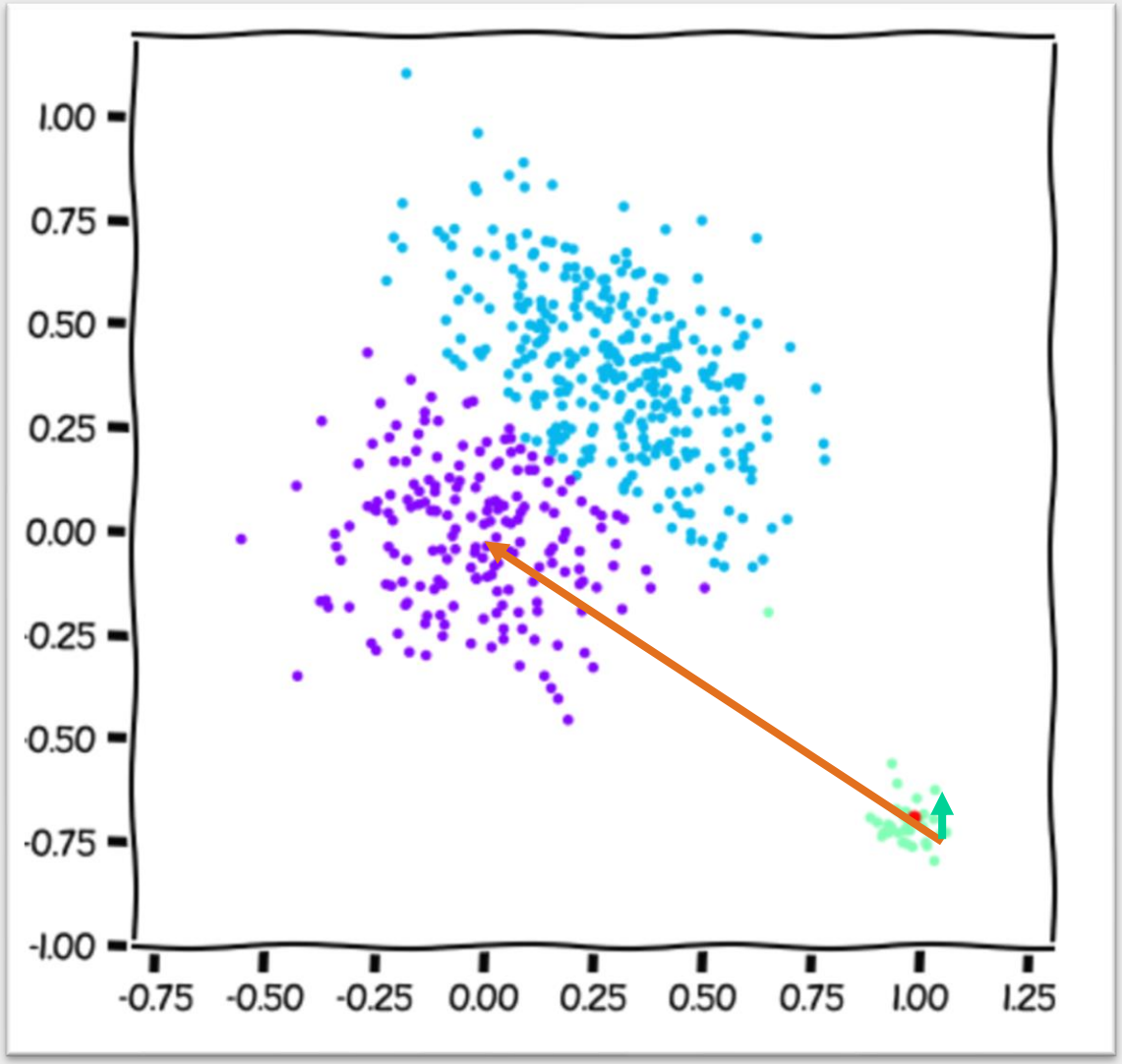
Silhouette score

Calculated for each point in each cluster

A average distance from the point to each other point in its cluster

B_1, B_2 average distance from the point to each point of another cluster

B $\min(B_1, B_2, \ldots)$
average distance from the point to the closest cluster
here $B = B_1$



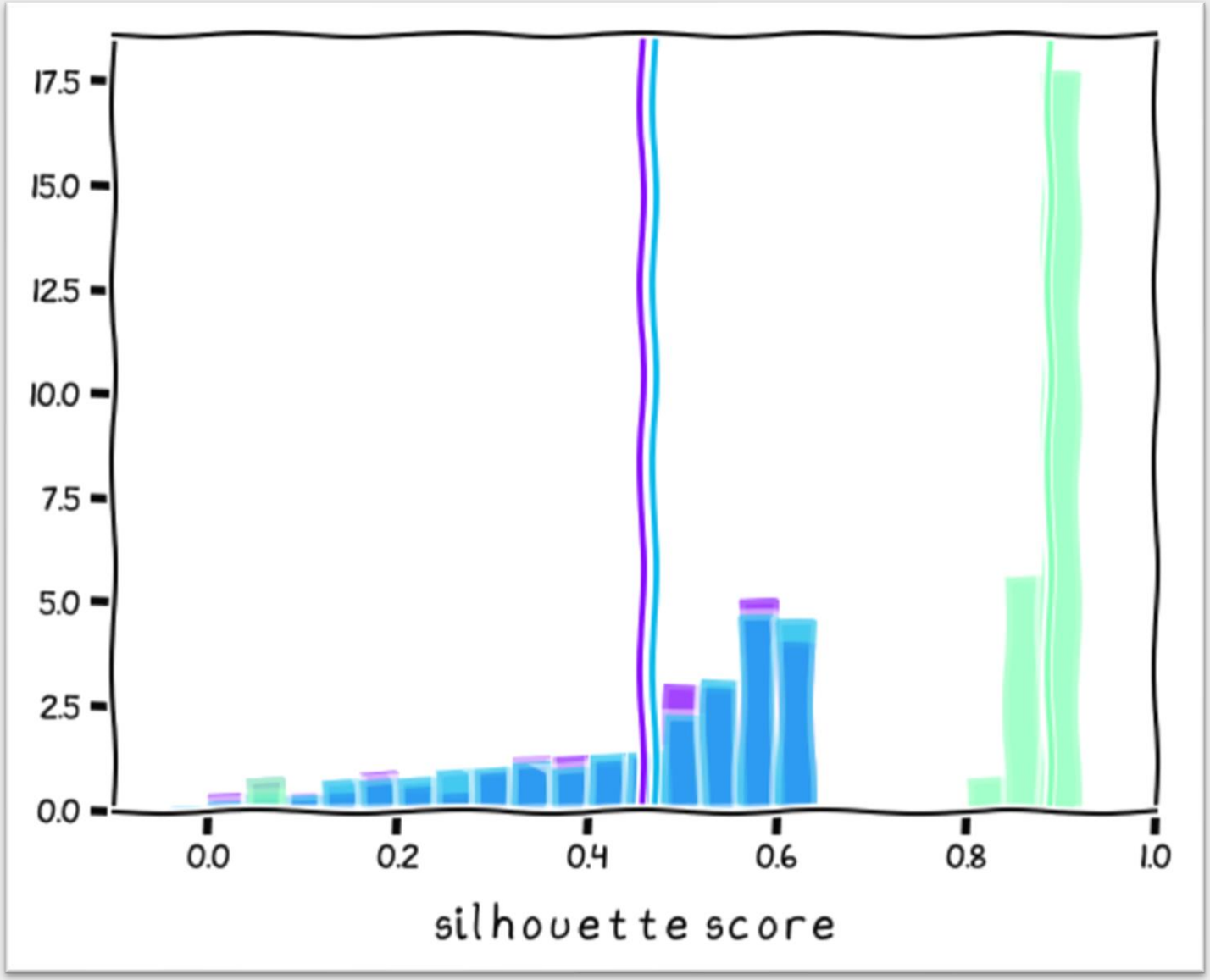
Silhouette score

Calculated for each point in each cluster

A average distance from the point to each other point in its cluster

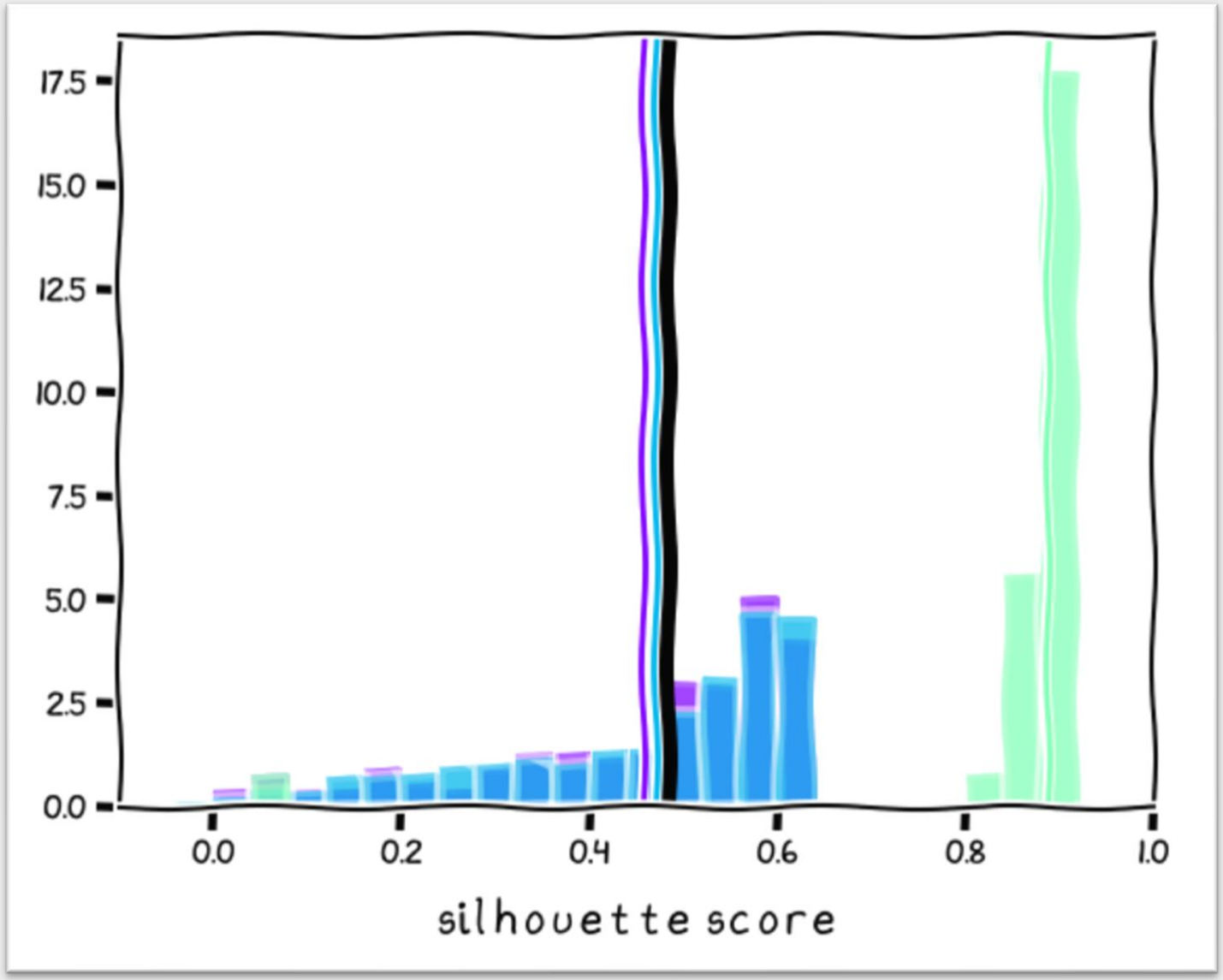
B $\min(B_1, B_2, \ldots)$
average distance from the point to the closest cluster
here $B = B_1$

$$\text{score} = \frac{B - A}{\max[B, A]} = 0.9$$



Do this for every point in one cluster

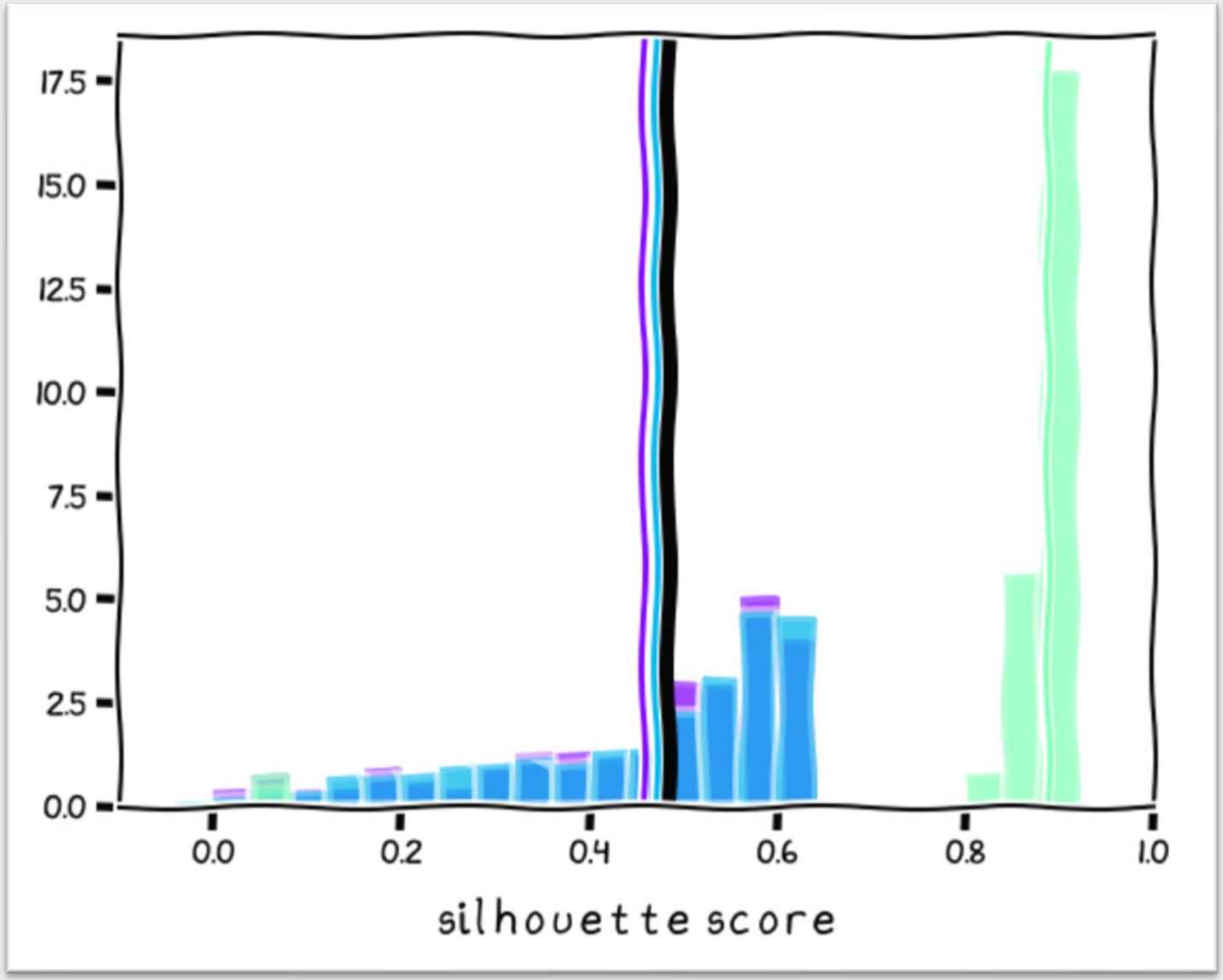
…and then every point in the other clusters…



Do this for every point in one cluster

…and then every point in the other clusters…

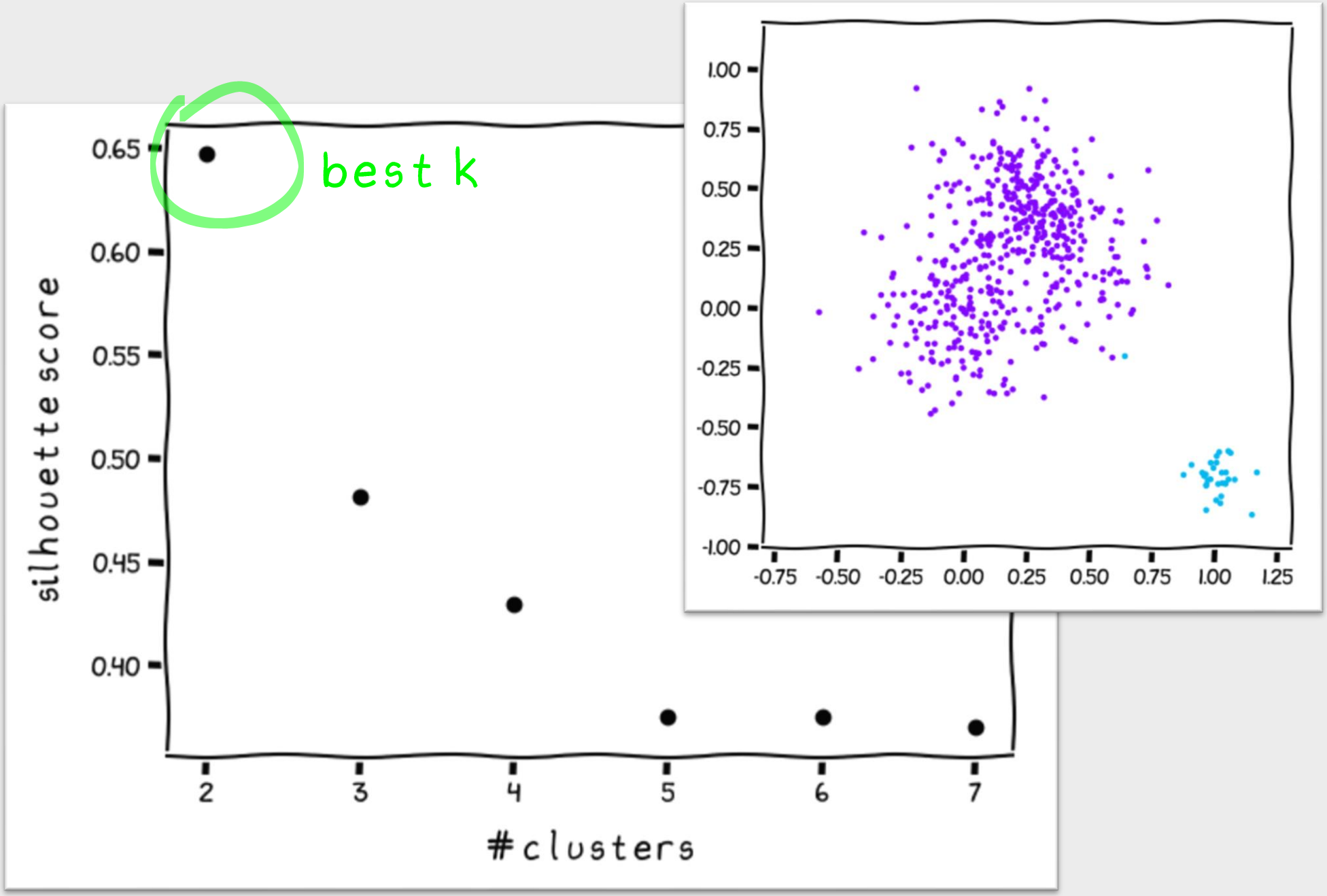
and find the **mean score for the entire dataset**



Do this for every point in one cluster

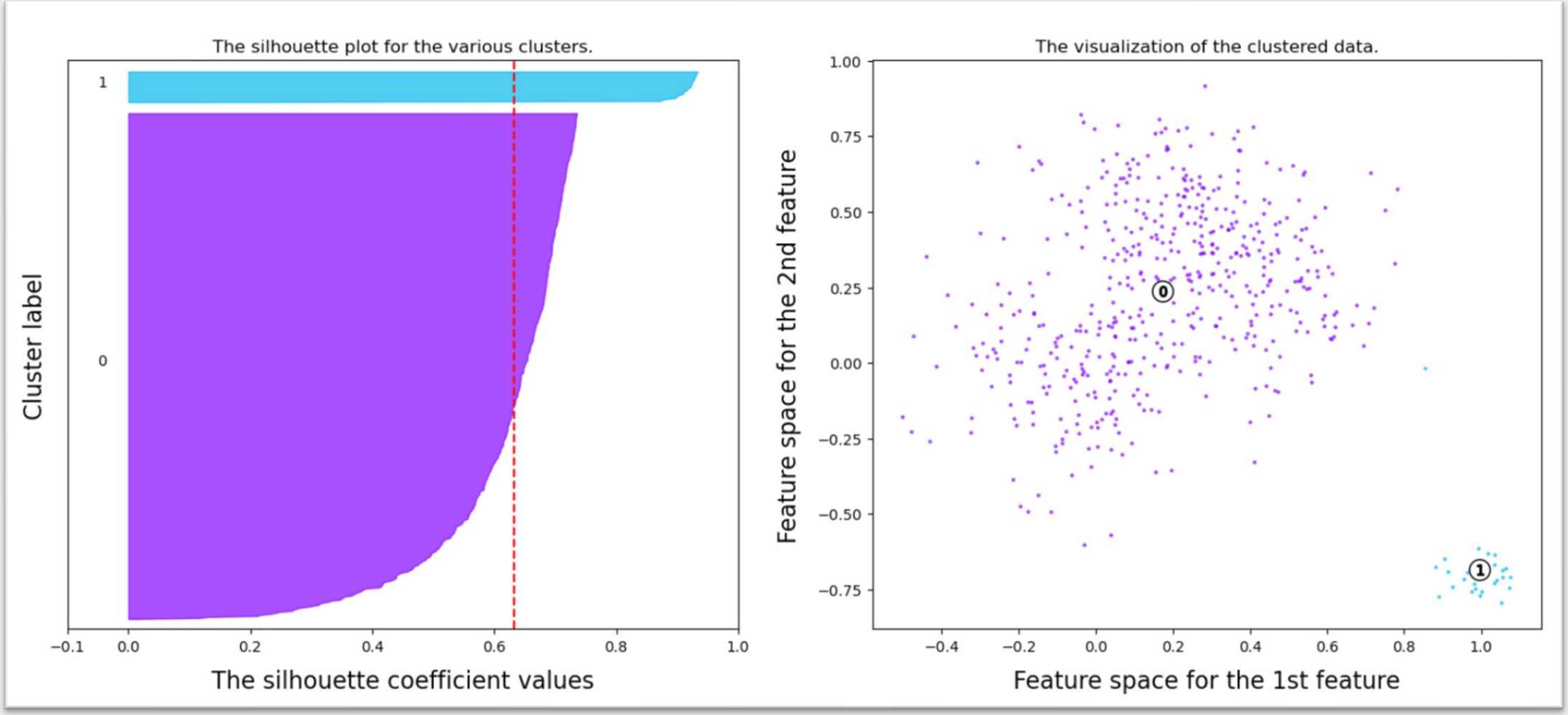
…and then every point in the other clusters…

and find the **mean score for the entire dataset**


best k
silhouette score
0.65
0.60
0.55
0.50
0.45
0.40
2
3
4
5
6
7
#clusters
1.00
0.75
0.50
0.25
0.00
-0.25
-0.50
-0.75
-1.00
-0.75
-0.50
-0.25
0.00
0.25
0.50
0.75
1.00
1.25

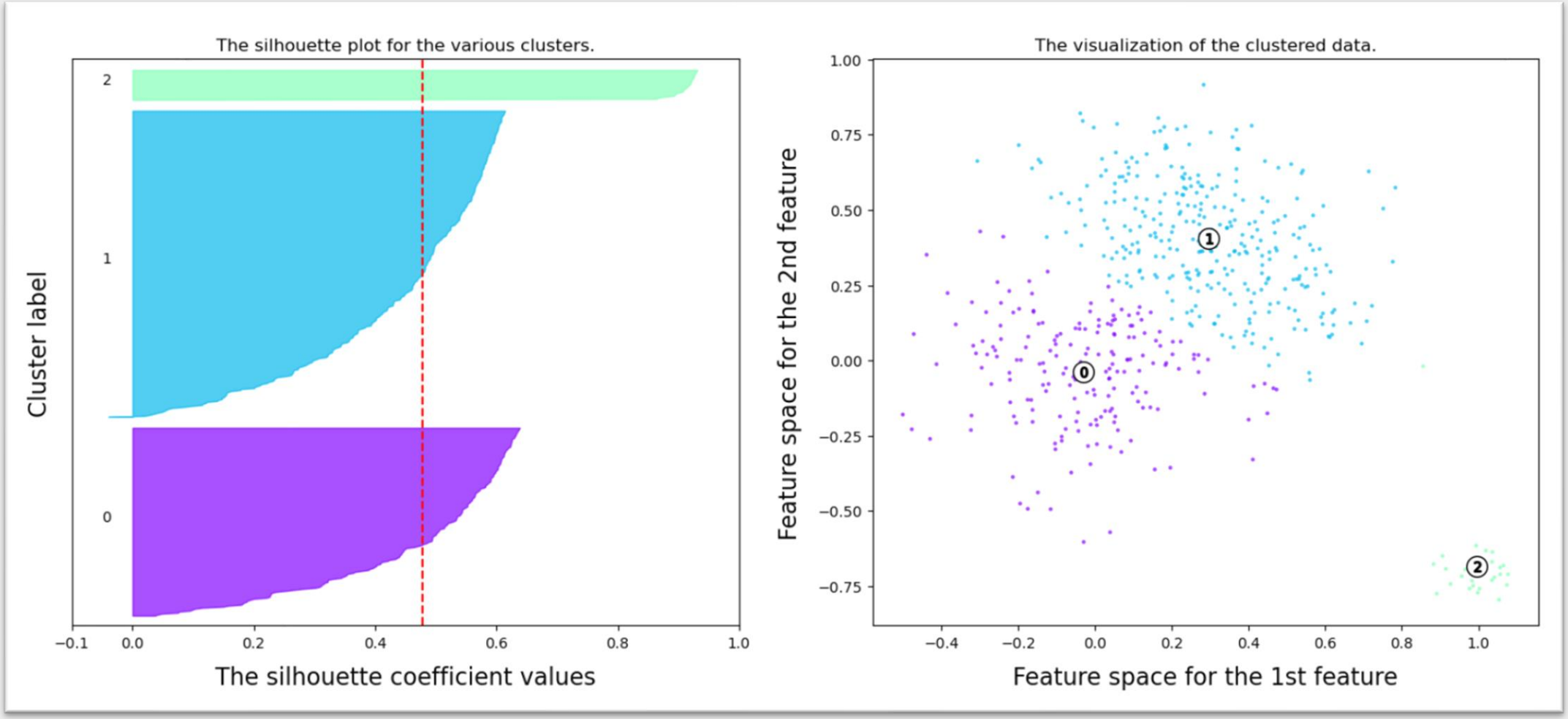
K-means silhouette score

sklearn.metrics.silhouette_score



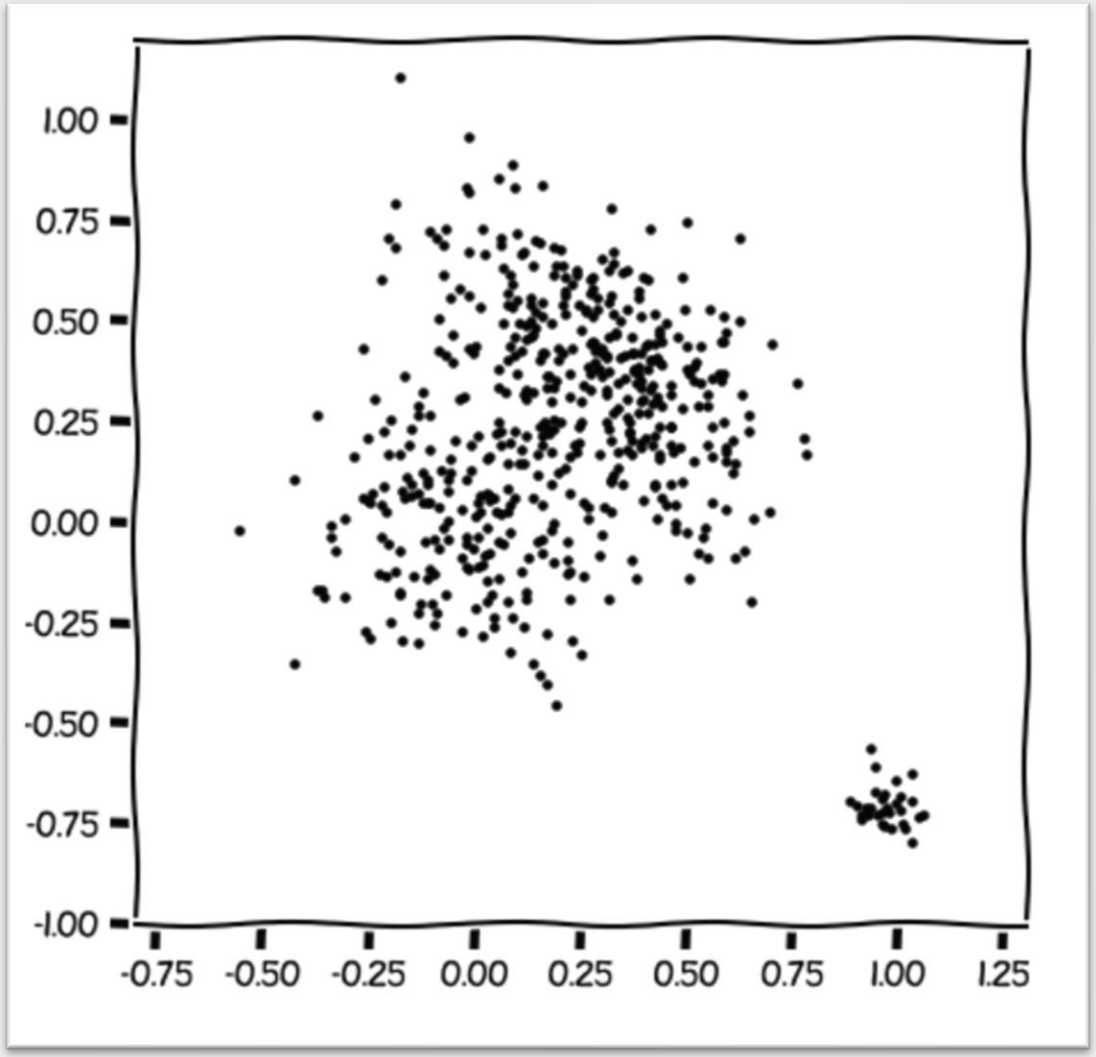
K-means silhouette score

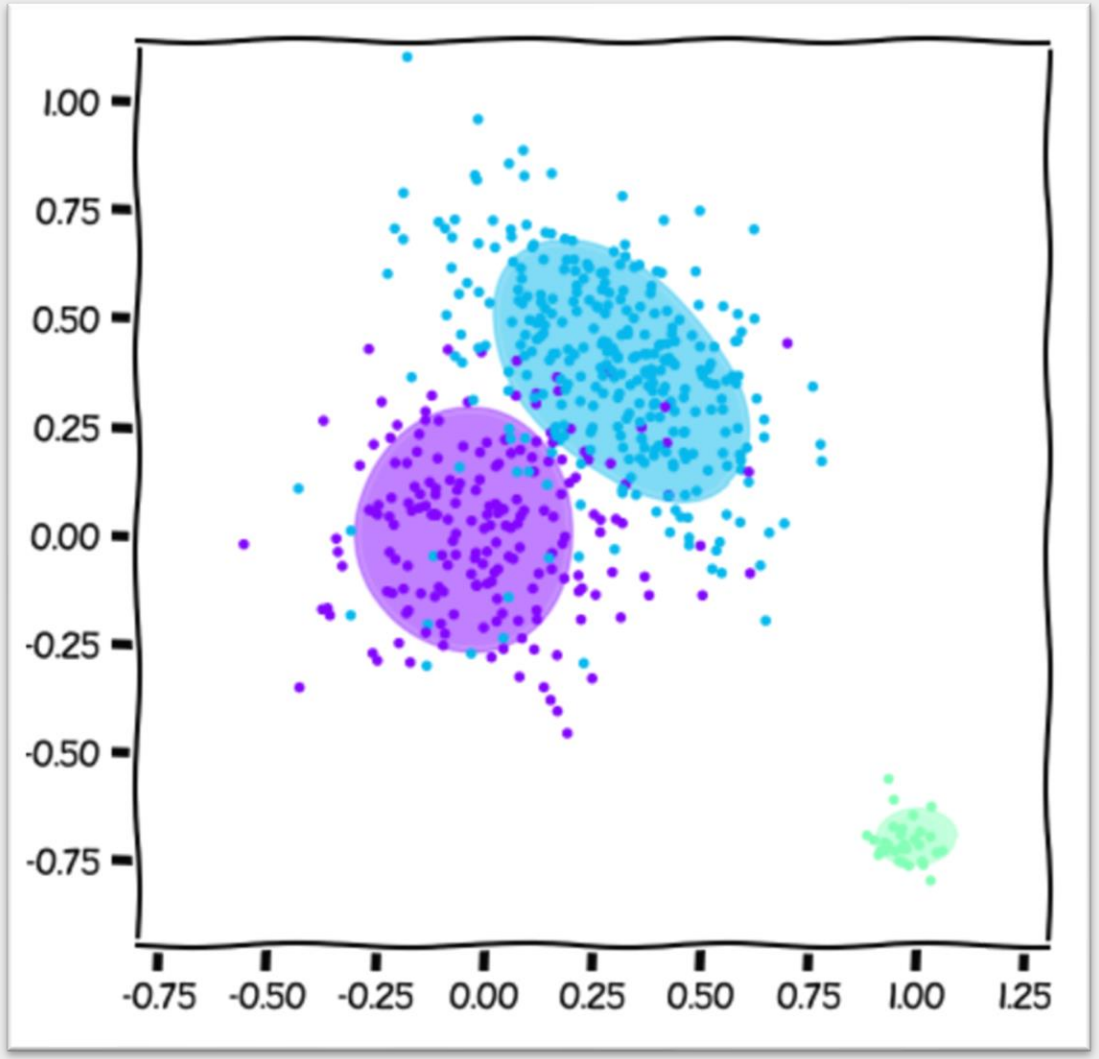
`sklearn.metrics.silhouette_score`

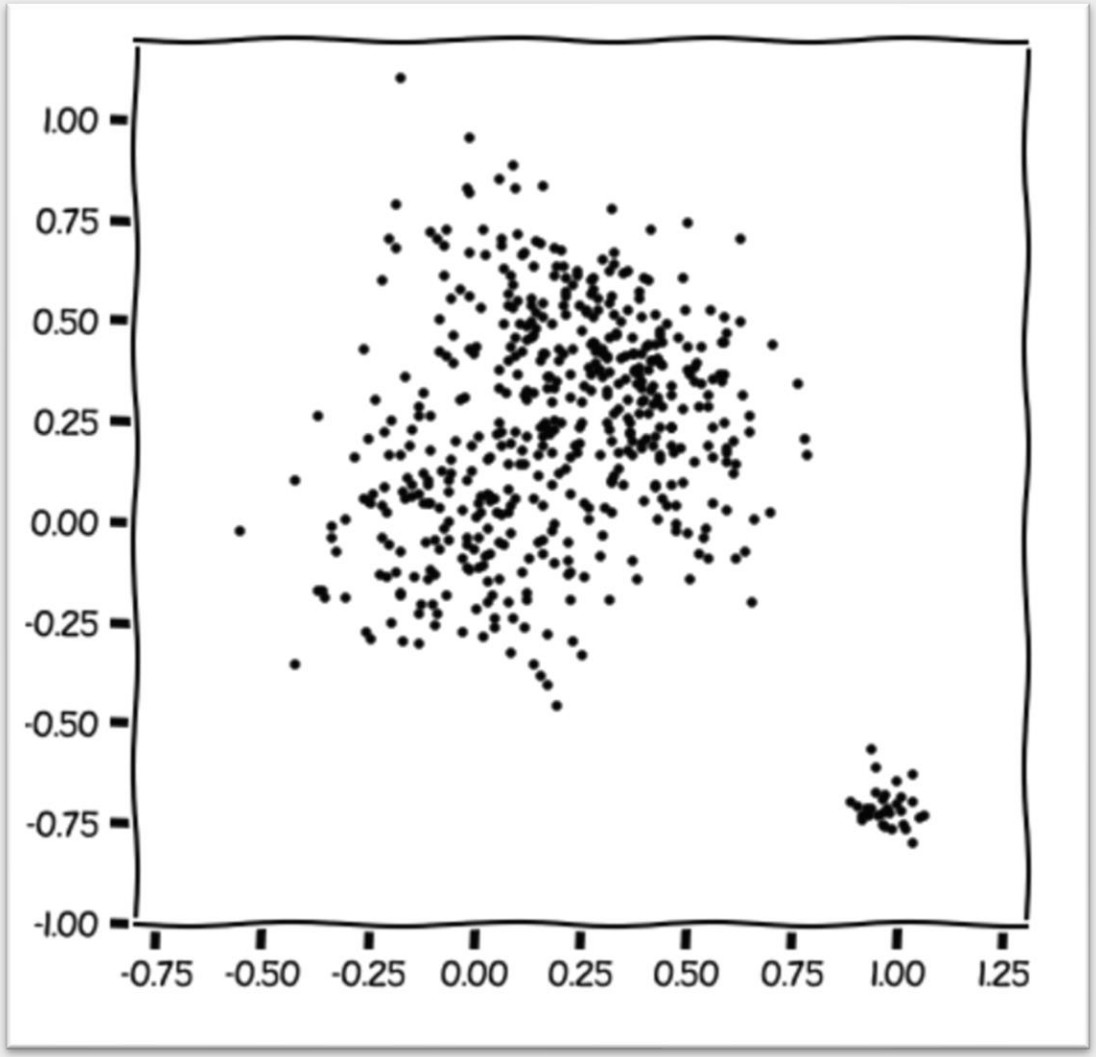


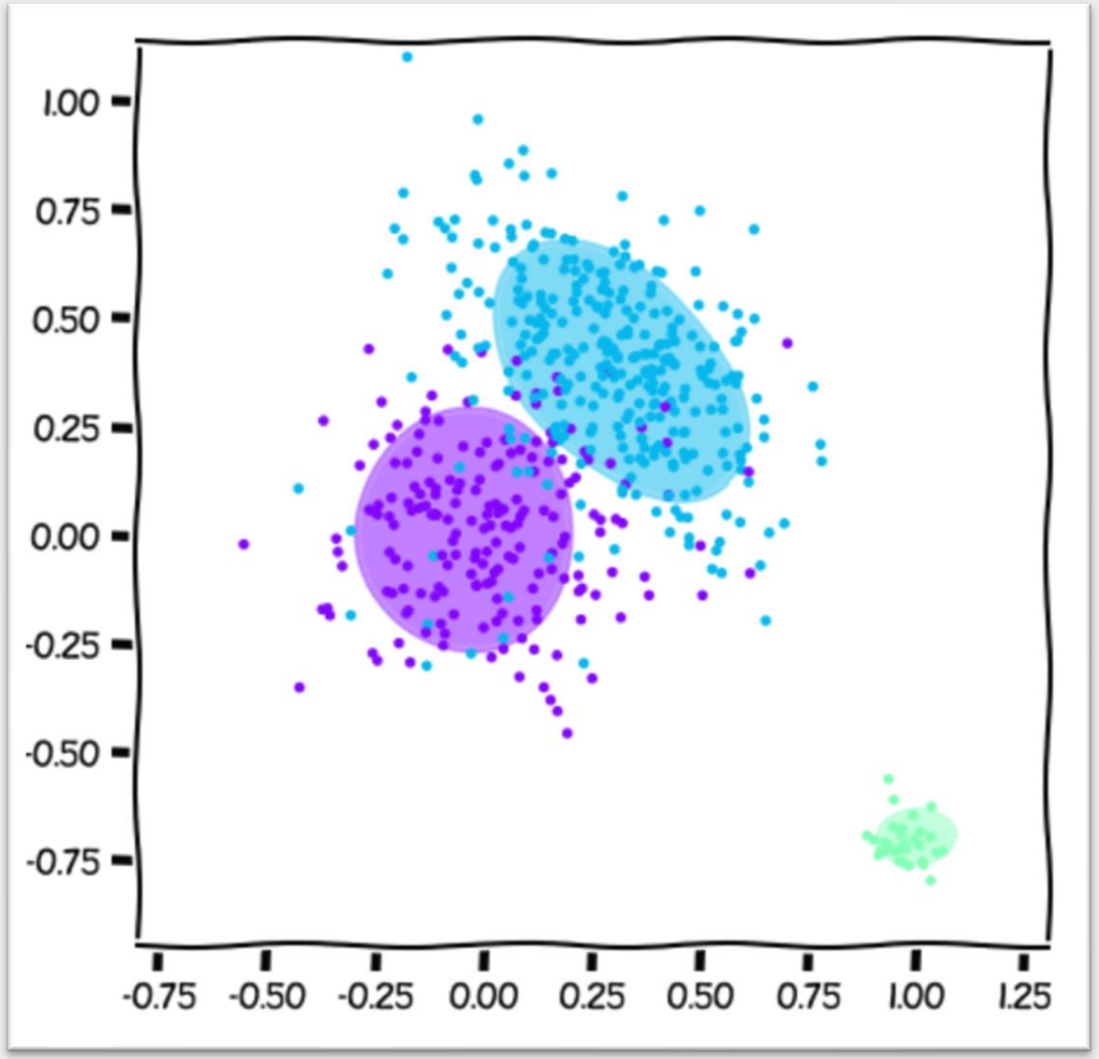
Gaussian Mixture Models

`sklearn.mixture`.GaussianMixture

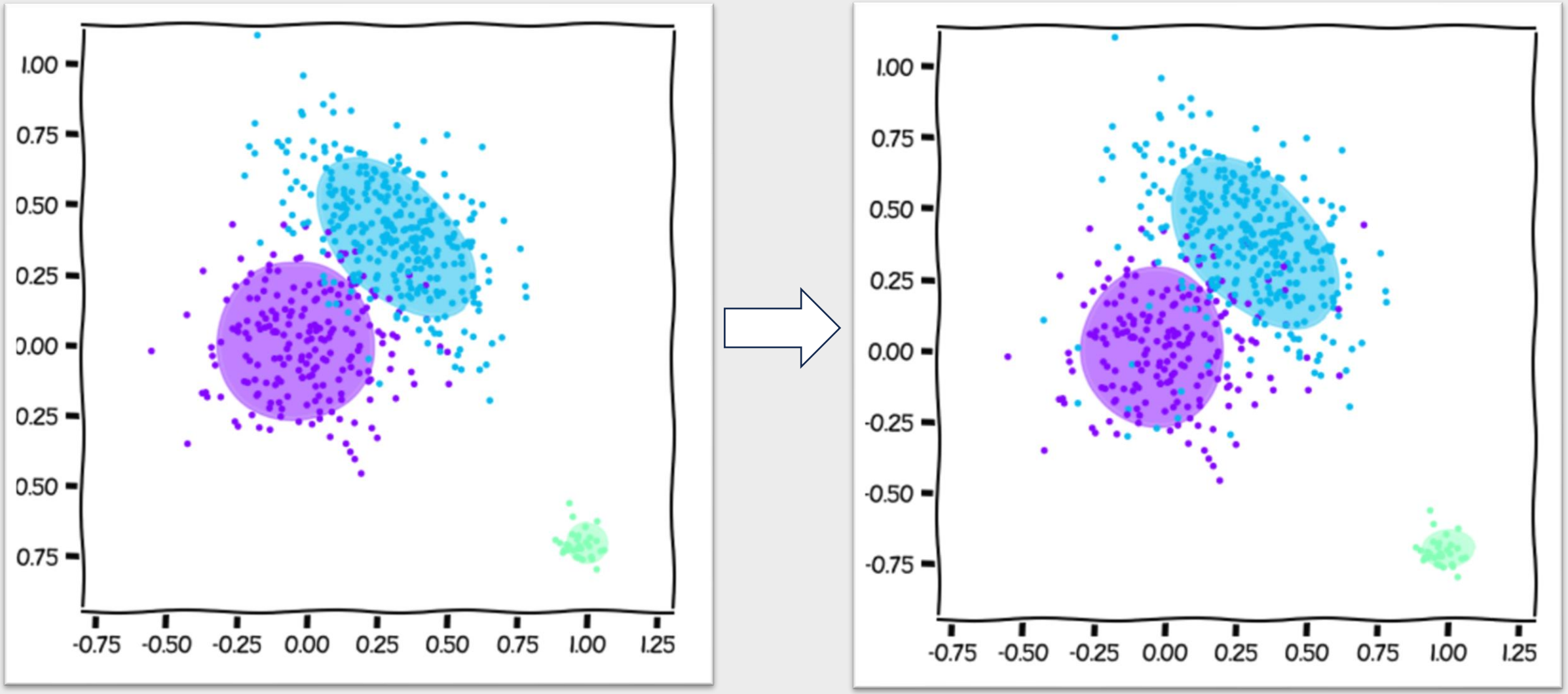

1.00
0.75
0.50
0.25
0.00
-0.25
-0.50
-0.75
-1.00
-0.75 -0.50 -0.25 0.00 0.25 0.50 0.75 1.00 1.25


1.00
0.75
0.50
0.25
0.00
-0.25
-0.50
-0.75
-0.75 -0.50 -0.25 0.00 0.25 0.50 0.75 1.00 1.25

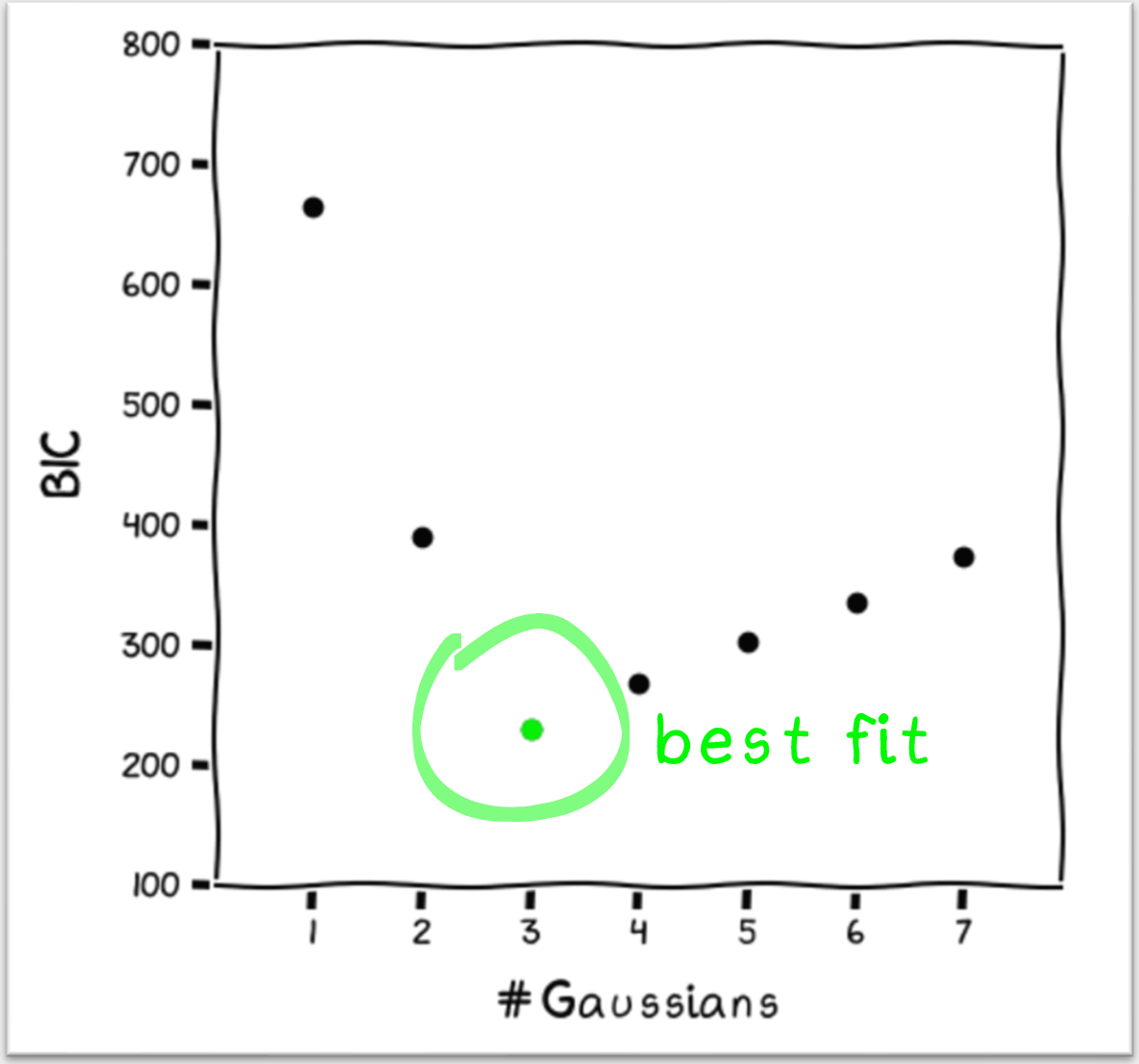




works amazingly if your data is Gaussian!



works amazingly if your data is Gaussian!



GMM

Bayesian Information Criterion

$$\mathrm{BIC} = k \ln n - 2 \ln L$$

k: # of params

n: # of data points

L: maximum likelihood

GMMs

pros

easy

really fast!

amazing

if your data is Gaussian

good for uneven clusters

provides statistics

provides probabilities

cons

need to decide on N

outliers

horrible

if your data isn't Gaussian

scales ok but not too well

means_ : *array-like of shape (n_components, n_features)*

The mean of each mixture component.

covariances_ : *array-like*

The covariance of each mixture component. The shape depends on `covariance_type`:

`aic`(X)	Akaike information criterion for the current model on the input X.
`bic`(X)	Bayesian information criterion for the current model on the input X.
`predict_proba`(X)	Evaluate the components' density for each sample.

Spectral Clustering

`sklearn.cluster`.SpectralClustering

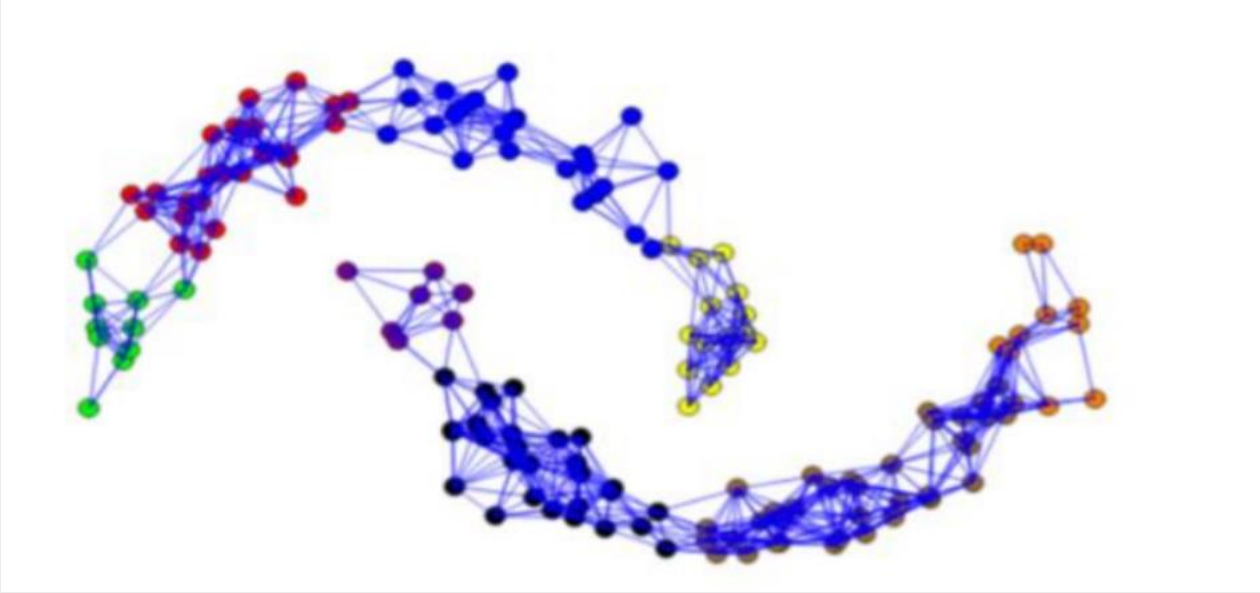
Apply clustering to a projection of the normalized Laplacian.

In practice Spectral Clustering is very useful when the structure of the individual clusters is highly non-convex, or more generally when a measure of the center and spread of the cluster is not a suitable description of the complete cluster, such as when clusters are nested circles on the 2D plane.

Spectral Clustering

`sklearn.cluster`.SpectralClustering

1. Calculate the distances between points
2. Construct a similarity graph



Affinity Matrix

How *close* are the data points i and j with a hard cut-off
Typically doesn't uses a Gaussian distance metric

$$A_{ij} = \exp\left[-\epsilon|x_i - x_j|^2\right]$$

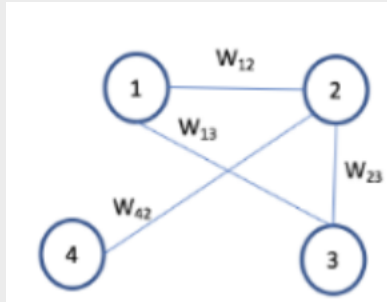
Degree Matrix

How many neighbors each point has

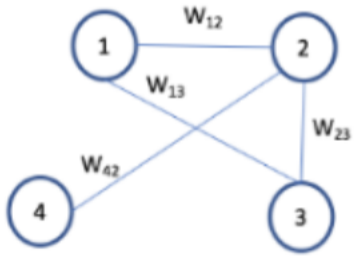
$$D_{ij} = \text{diag}\,[N]$$

Laplacian Matrix

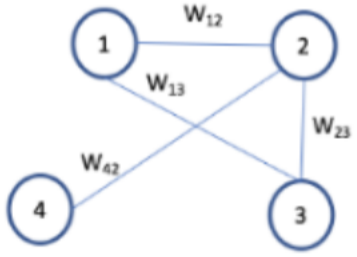
$$L = D - A$$



	1	2	3	4
1	0	w_{12}	w_{13}	0
2	w_{12}	0	w_{23}	w_{24}
3	w_{13}	w_{23}	0	0
4	0	w_{24}	0	0



	1	2	3	4
1	2			
2		2		
3			2	
4				1



	1	2	3	4
1	d_1	$-w_{12}$	$-w_{13}$	0
2	$-w_{12}$	d_2	$-w_{23}$	$-w_{24}$
3	$-w_{13}$	$-w_{23}$	d_3	0
4	0	$-w_{24}$	0	d_4

Affinity Matrix

How *close* are the data points i and j with a hard cut-off
Typically doesn't uses a Gaussian distance metric

$$A_{ij} = \exp\left[-\epsilon|x_i - x_j|^2\right]$$

Degree Matrix

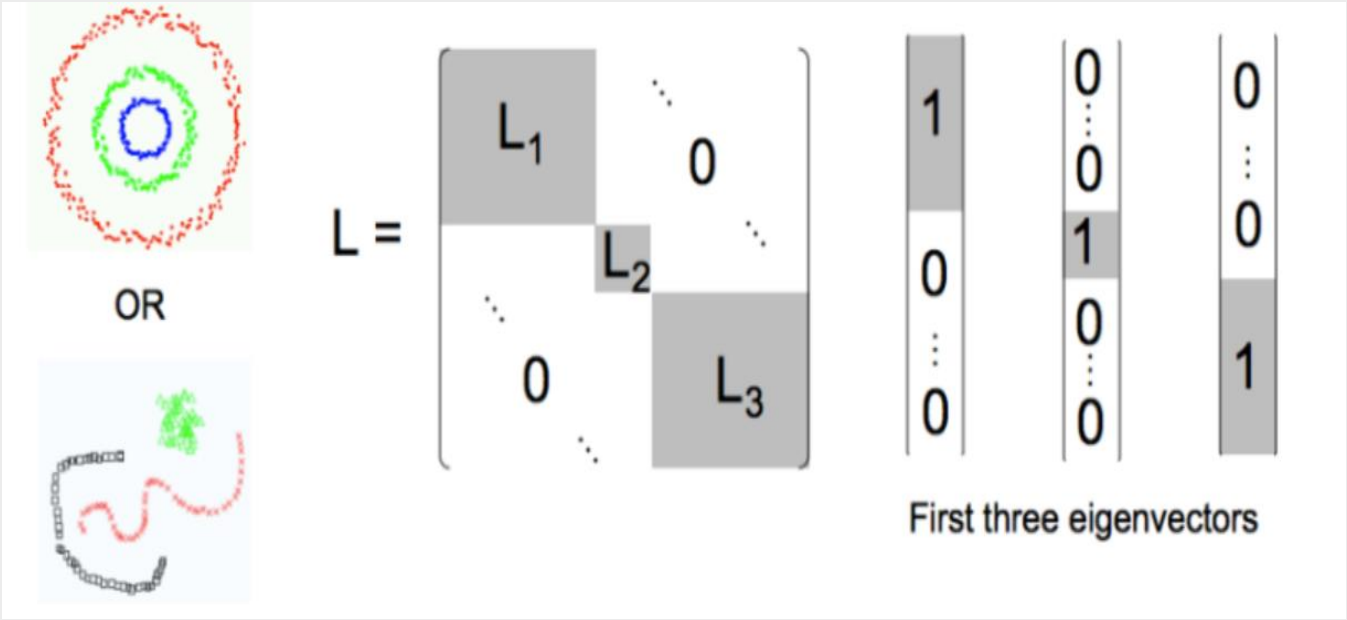
How many neighbors each point has

$$D_{ij} = \text{diag}\,[N]$$

Laplacian Matrix

$$L = D - A$$

Simplify L



3. Find N smallest eigenvalues (diagonalize it)
4. Project everything into N-dim space
5. Run k-means!

Affinity Matrix

How *close* are the data points i and j with a hard cut-off
Typically doesn't uses a Gaussian distance metric

$$A_{ij} = \exp\left[-\epsilon|x_i - x_j|^2\right]$$

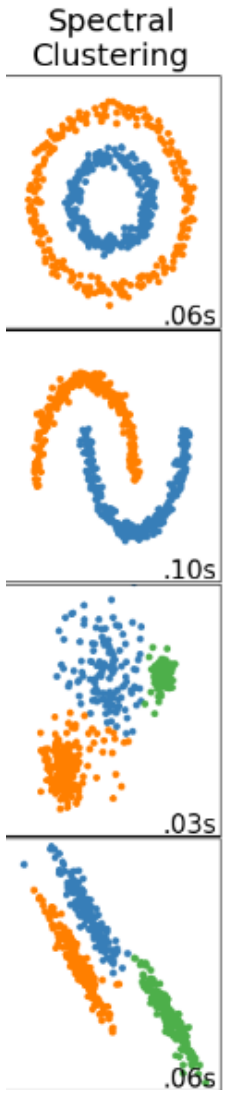
Degree Matrix

How many neighbors each point has

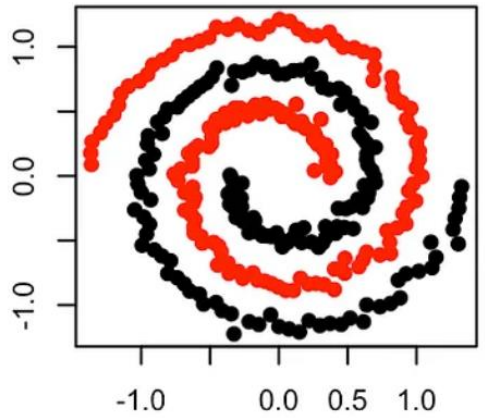
$$D_{ij} = \text{diag}\,[N]$$

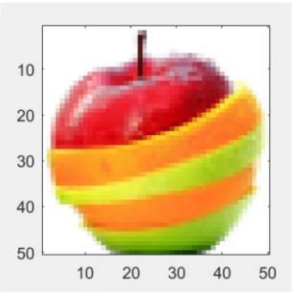
Laplacian Matrix

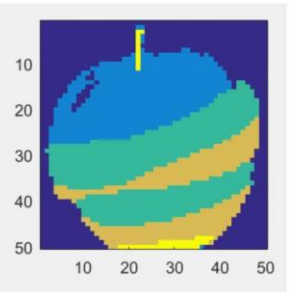
$$L = D - A$$


Spectral
Clustering
.06s
.10s
.03s
.06s


3

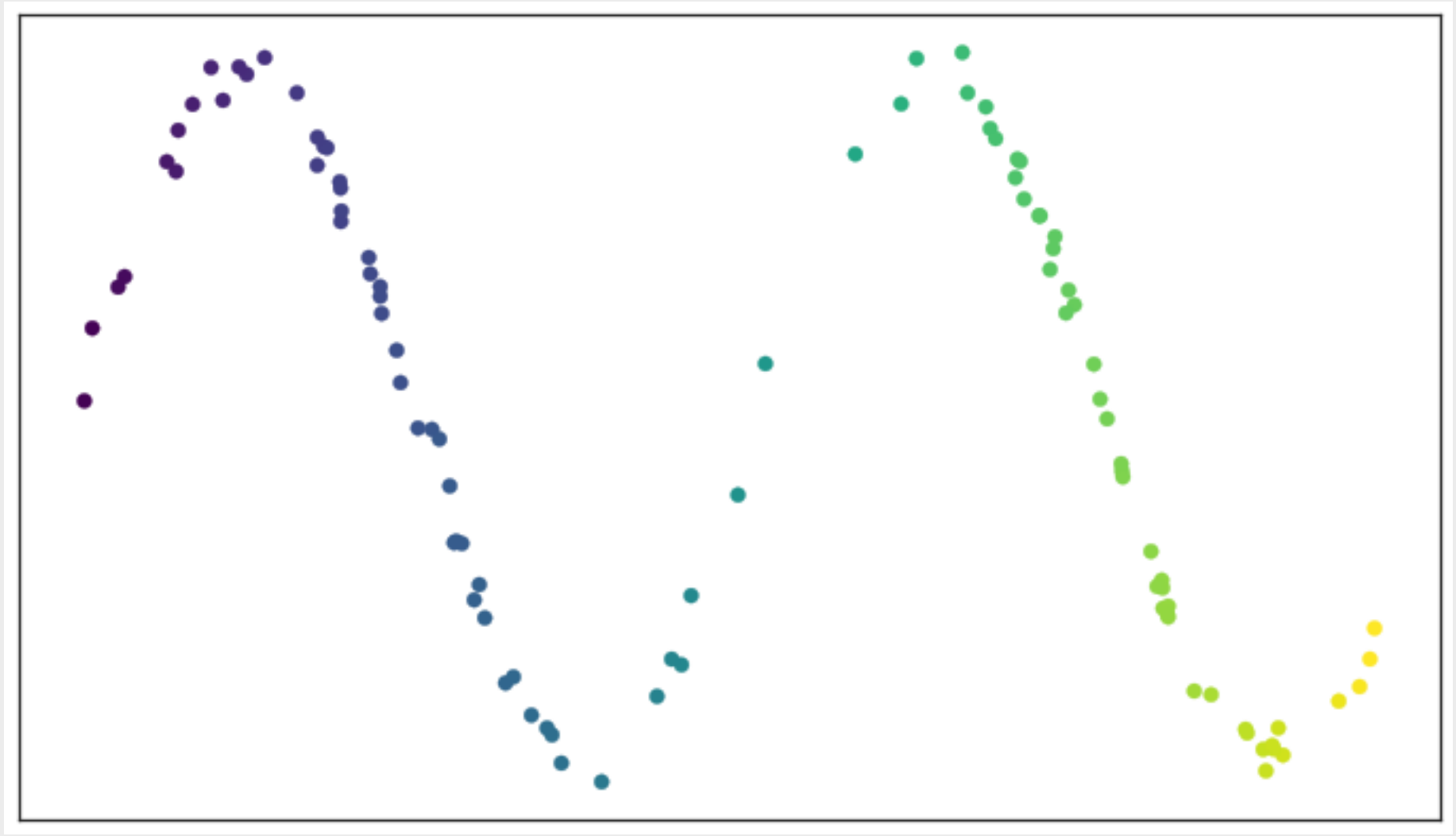

1.0
0.0
-1.0
-1.0
0.0
0.5
1.0

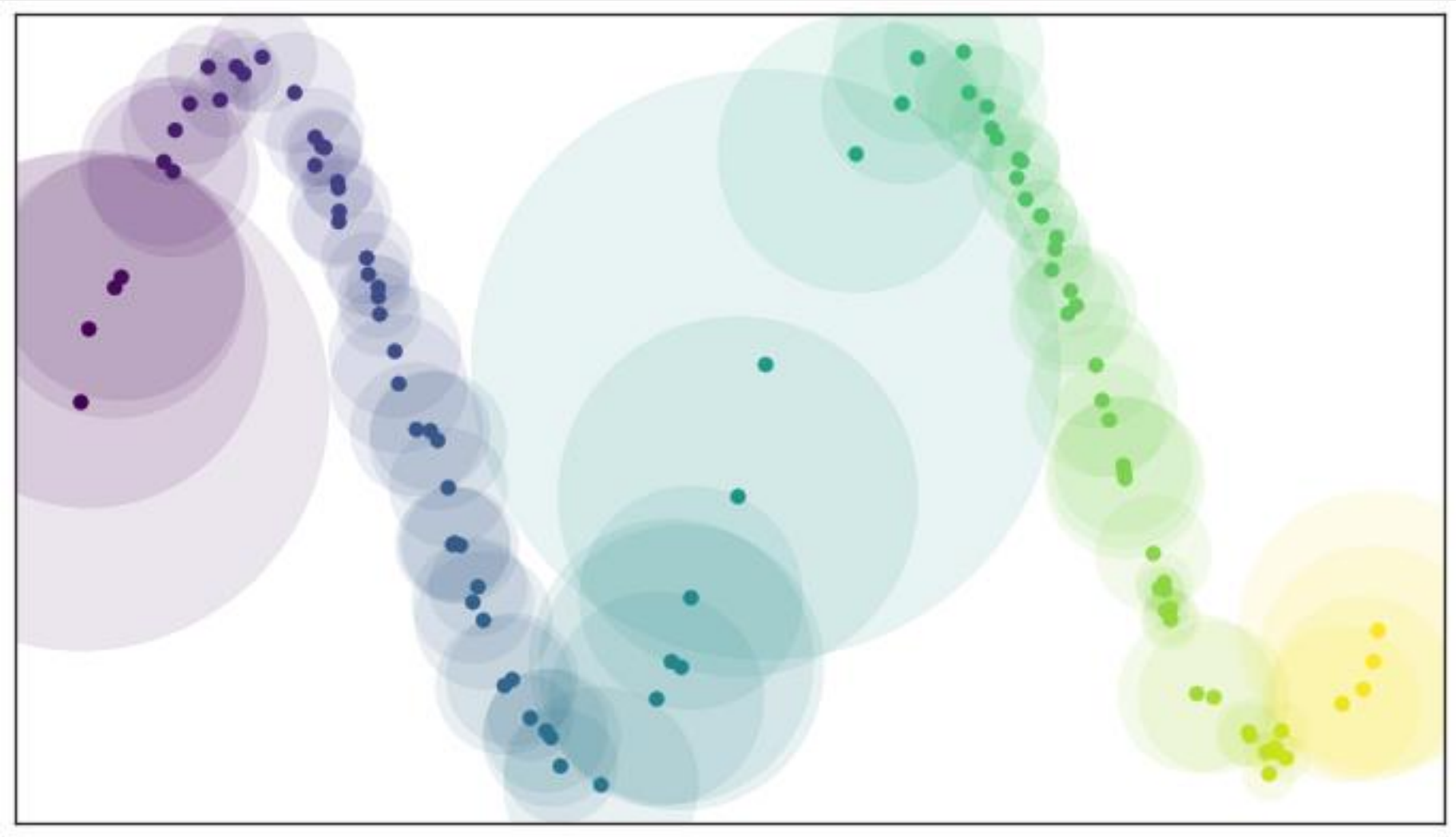

10
20
30
40
50
10
20
30
40
50

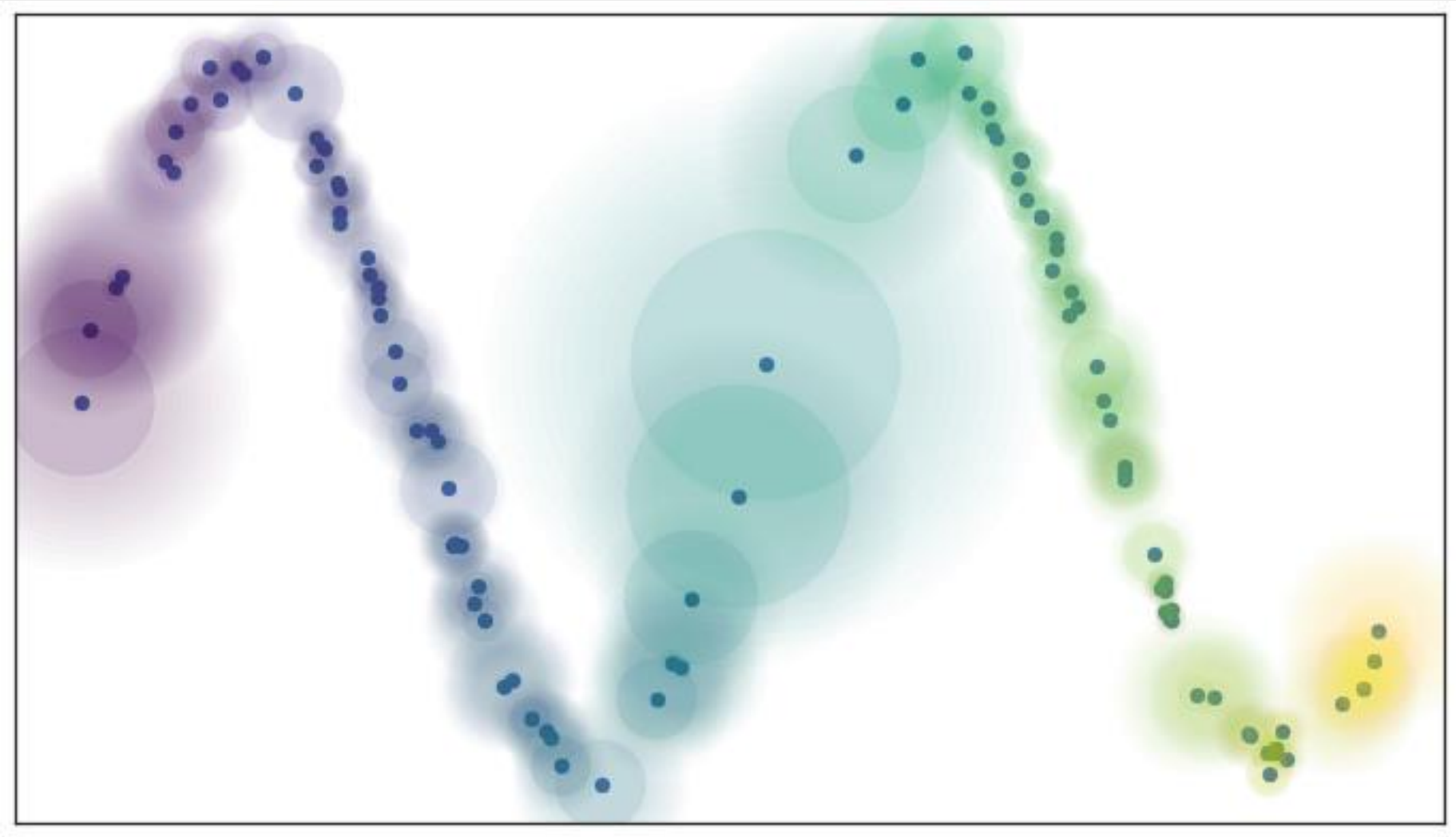

10
20
30
40
50
10
20
30
40
50

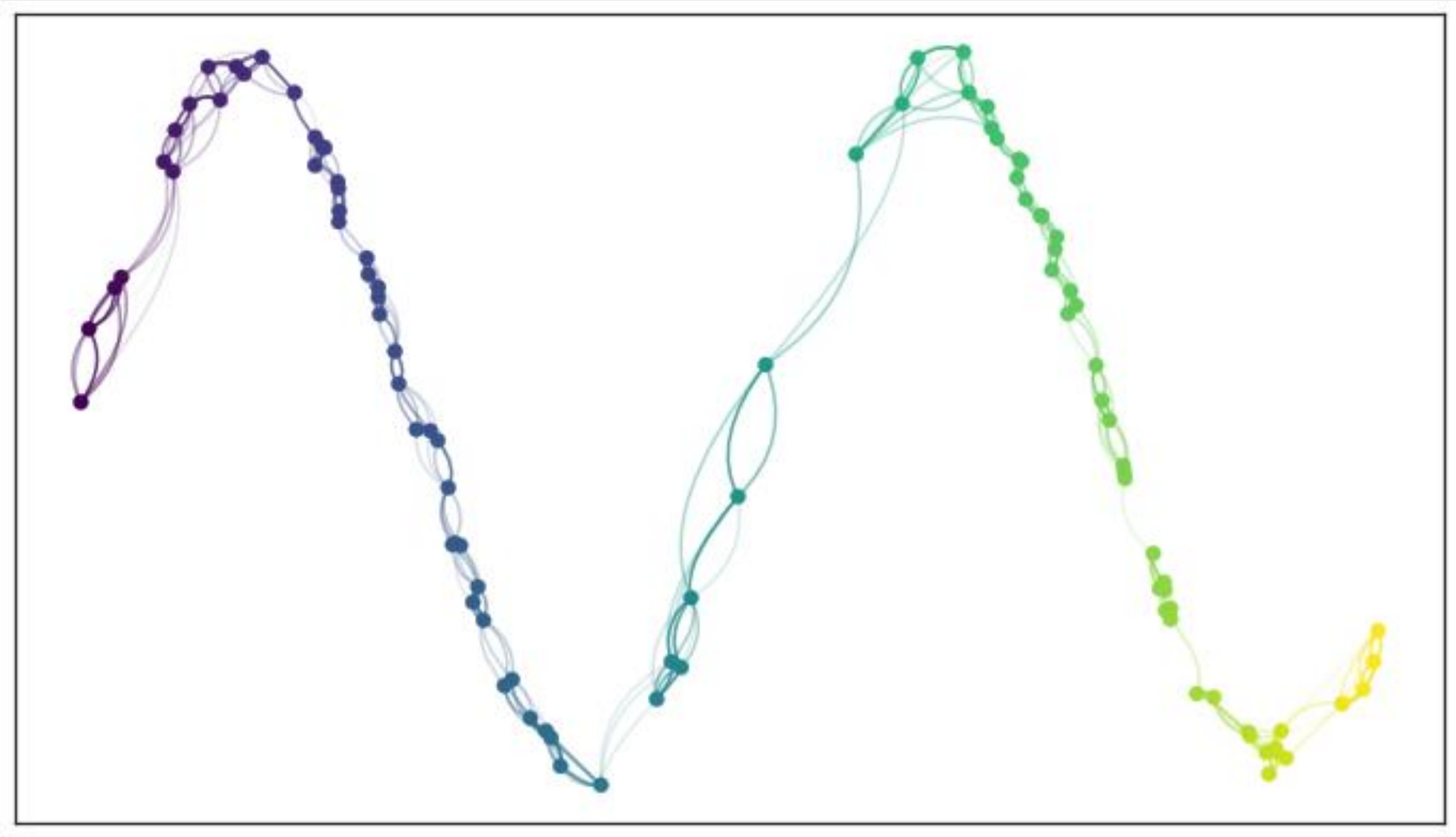
UMAPs

https://umap-learn.readthedocs.io/en/latest/

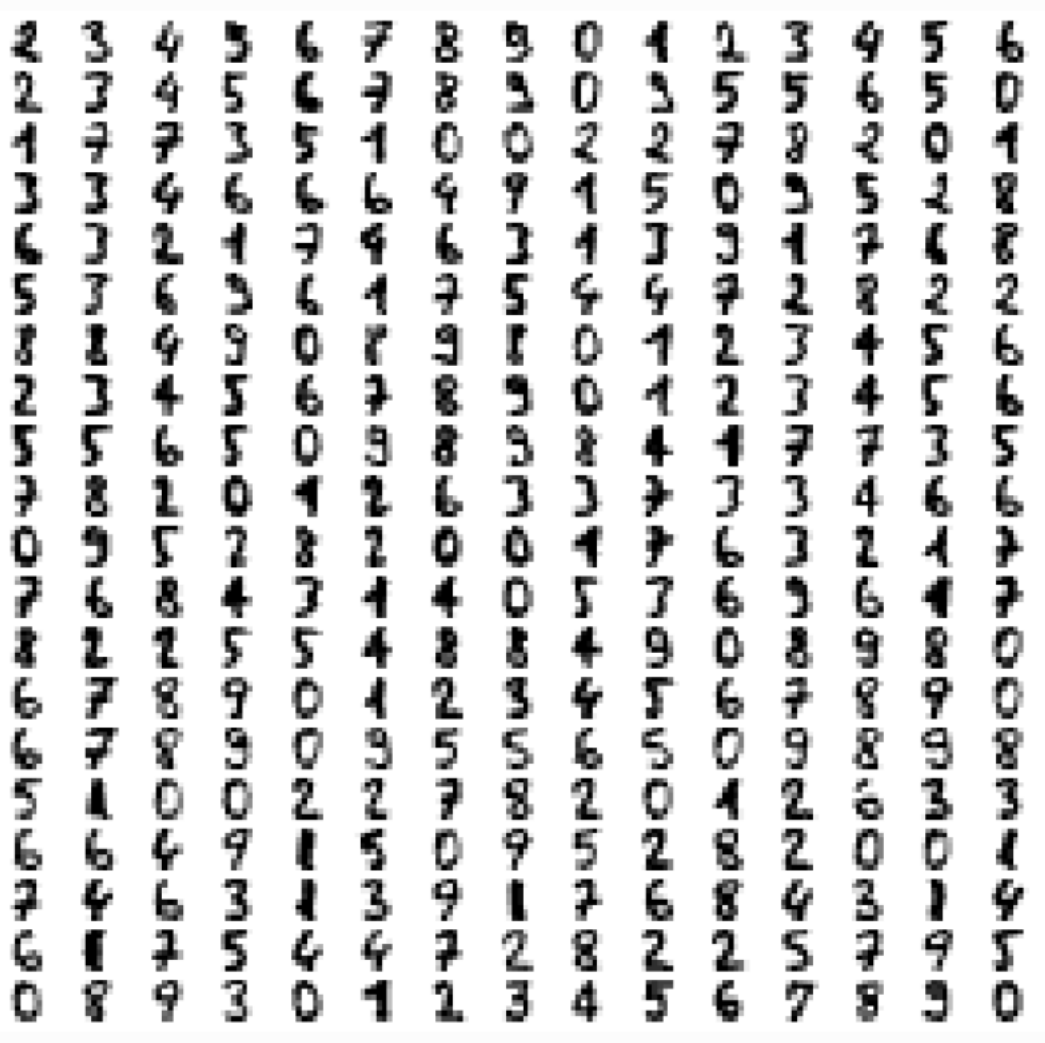








8x8 digits: 64-dimensional data



2-dimensional data w/ same graph

Final tip!

sklearn.model_selection.KFold ¶

class sklearn.model_selection.**KFold**(*n_splits=5, *, shuffle=False, random_state=None*) [source]

K-Fold cross-validator.

Provides train/test indices to split data in train/test sets. Split dataset into k consecutive folds (without shuffling by default).

Each fold is then used once as a validation while the k - 1 remaining folds form the training set.

Read more in the User Guide.

For visualisation of cross-validation behaviour and comparison between common scikit-learn split methods refer to Visualizing cross-validation behavior in scikit-learn