

Introduction to Machine Learning for Bio-image Analysis

Robert Haase

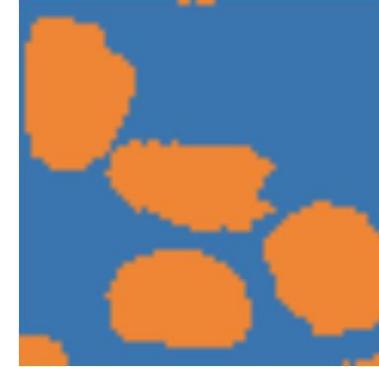
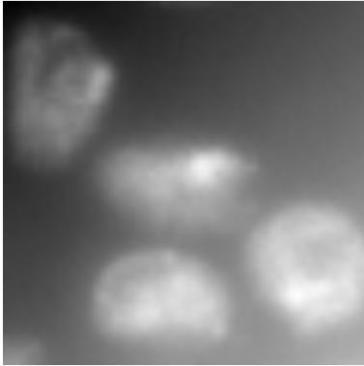
Reusing materials from Johannes Soltwedel, Till Korten, Johannes Soltwedel, Laura Žigutytė (TU Dresden), Ryan Savill (MPI-CBG Dresden), Matthias Täschner (ScaDS.AI/Uni Leipzig) and the Scikit-learn community.

Follow-up: Tiled versus full image processing

Results were very different. What can we do about it?

Pixel-Classification

Quiz: How is this task called?



Combinatorial
Segmentation



Semantic
Segmentation



Instance-
Segmentation



Connected-Component
Segmentation

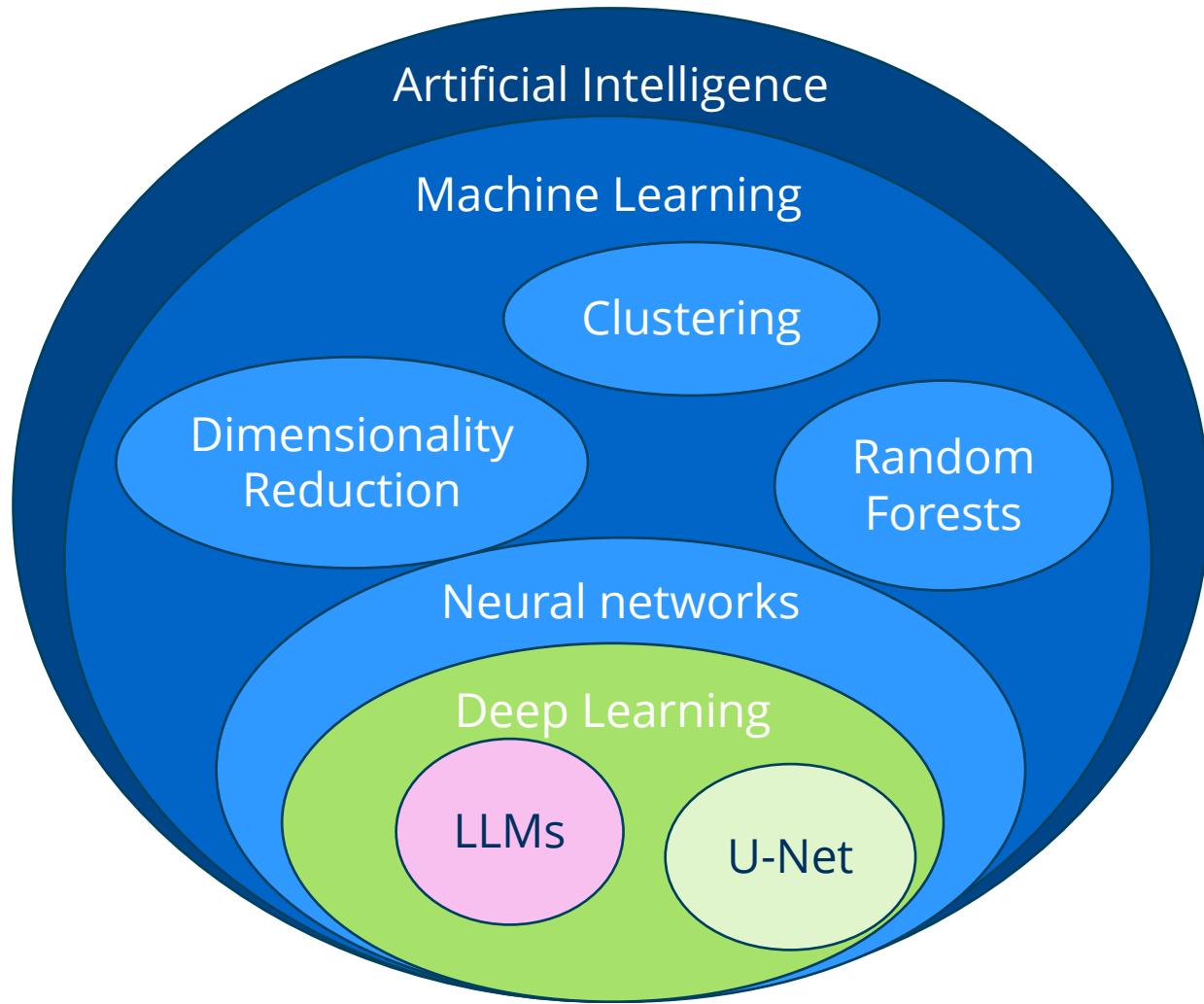


Introduction to Machine Learning for Bio-image Analysis

Robert Haase

Reusing materials from Johannes Soltwedel, Till Korten, Johannes Soltwedel, Laura Žigutytė (TU Dresden), Ryan Savill (MPI-CBG Dresden), Matthias Täschner (ScaDS.AI/Uni Leipzig) and the Scikit-learn community.

Artificial intelligence



Today
Next week
The week after

Artificial intelligence

Narrow AI

- Application specific
- Trained on labelled data
- Reflexive tasks
- Cannot extrapolate

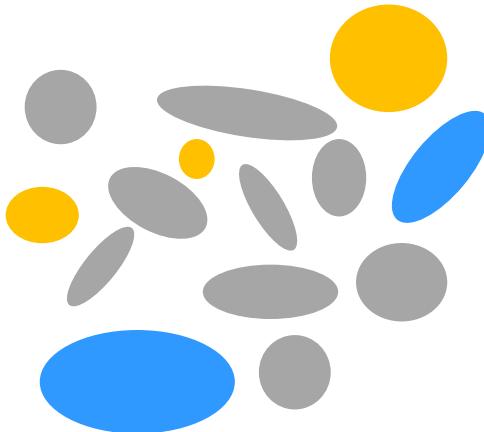
Great for data analysis tasks

General AI

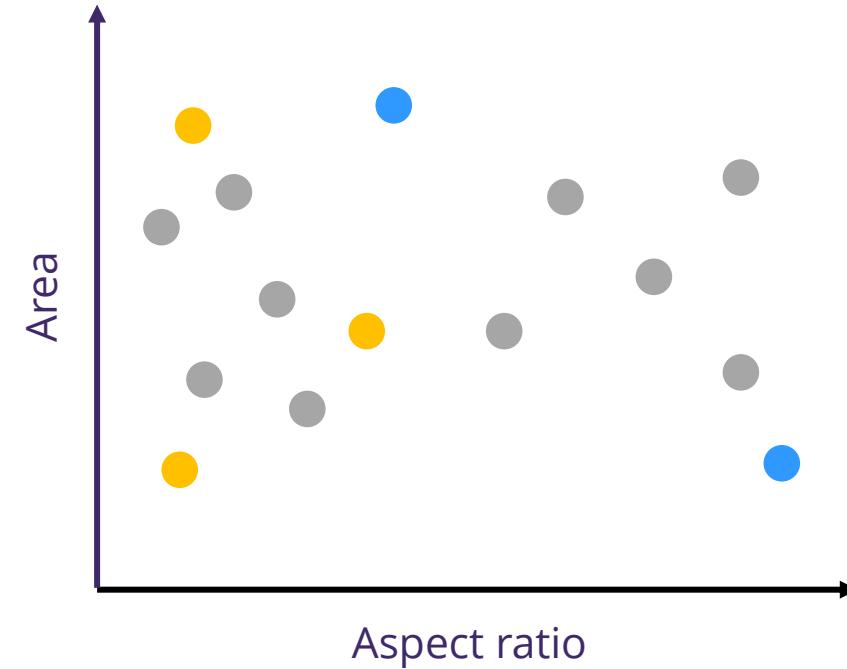
- Human capabilities
- Access to knowledge of humanity, beyond individuals
- Can create *new* solutions by working creatively

Labelled data

- E.g. for shape differentiation of objects
- Partially labelled data  Bias?



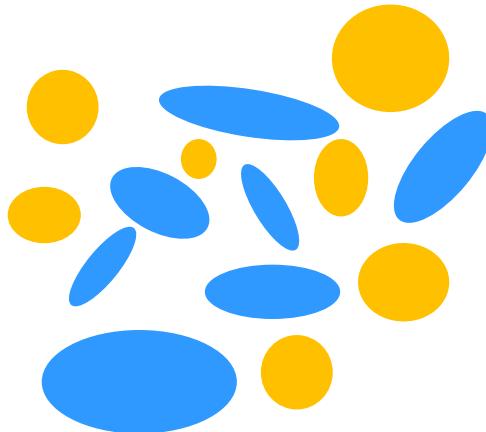
Elongated
Round
Unlabelled



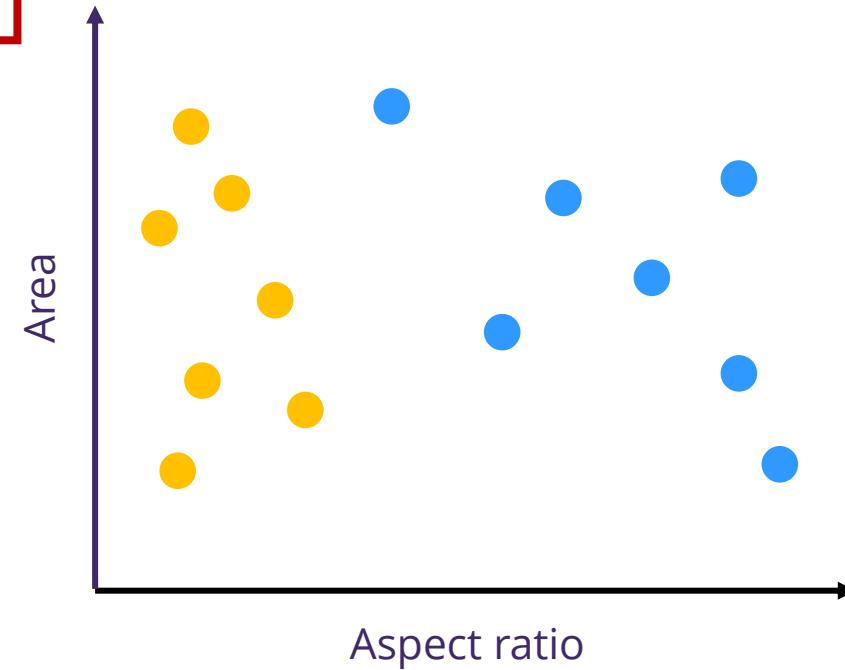
Labelled data

- E.g. for shape differentiation of objects
- Fully labelled data

Typically expensive



Elongated
Round
Unlabelled

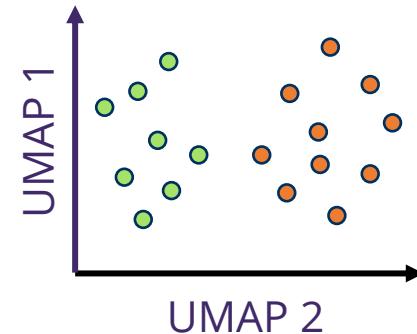


Artificial intelligence



Unsupervised ML

- Dimensionality reduction
- Clustering
- Detecting patterns in unlabeled data
- Hypothesis generation



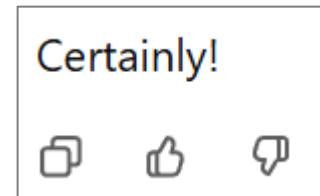
Supervised ML

- Learning tasks otherwise only humans could do
- Train a model based on labeled data, predict a classification



Generative AI

- Produces new data provided a context, often with human language prompts
- Hyped since 2022, with yet unclear limitations



Unsupervised Machine Learning for Bio-image Analysis

Robert Haase

Reusing materials from Johannes Soltwedel, Till Korten, Johannes Müller, Laura Žigutytė (TU Dresden), Ryan Savill (MPI-CBG), Matthias Täschner (ScaDS.AI/Uni Leipzig) and the Scikit-learn community.

Hypothesis-driven quantitative biology

Hypothesis: Cell shape can be influenced by modifying X.

Null-Hypothesis: Circularity of modified cells is similar to cells in the control group.

Should we use a
different
segmentation
algorithm?

Sample preparation

Imaging

Shall we use a
different
microscope?

Cell segmentation

Circularity measurement

Is circularity the
right parameter to
measure?

Statistics

Hypothesis generating quantitative biology

Hypothesis: Cell shape can be influenced by modifying X.

Question: Which image-derived parameter is influenced when modifying X?

Sample preparation

Imaging

Cell segmentation algorithm A, algorithm B, algorithm C

Measurement of circularity, solidity, elongation, extend, texture, intensity, topology ...

Statistics

Which segmentation algorithms allow measurements that show a relationship with X?

Why?

Which parameter shows any relationship with X?

Feature selection

- Which measurement / parameter / feature is related to the effect I'm investigating?
- Example goals:

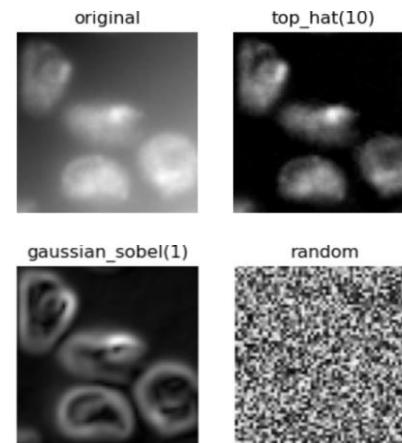


- Amplitude
- Energy
- Duration
- ...



- Noise
- Tourists jumping on a sensor
- Earthquake approaching

Signal classification



Pixel classification



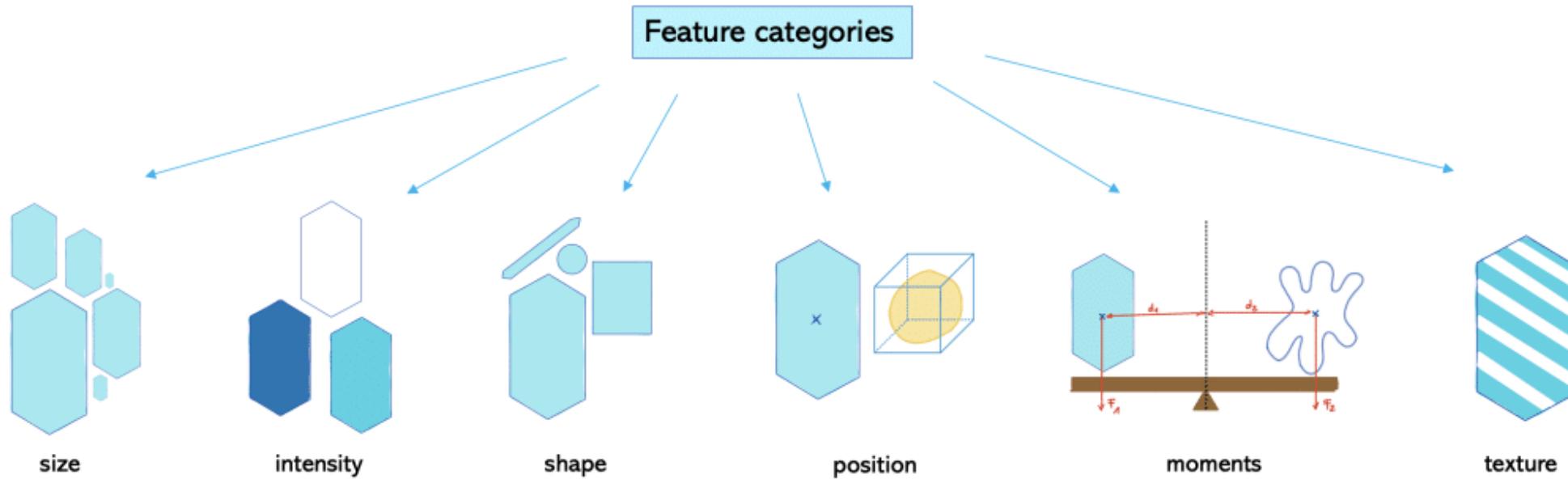
- Area
- Perimeter
- Aspect ratio
- ...



- Round
- Elongated

Object classification

Feature selection



Which of these features reflect the phenotype we are perceiving?

Feature selection: challenges

- Features are not independent
 - Area and diameter
 - Roundness, circularity, solidity, extent, aspect ratio, elongation, Feret's diameter, ...
- Best classification most likely involves multiple features
- Vast amount of features can hardly be visualized
- Need for dimensionality reduction
 - Principal component analysis (PCA)
 - t-Distributed Stochastic Neighbour Embedding (t-SNE)
 - Uniform Manifold Approximation and Projection (UMAP)
- Grouping objects (clustering)

Feature selection

Question: Which features shall I analyse?

Challenges:

- Physical properties versus measurable features
- Correlation versus causation
- Too many features

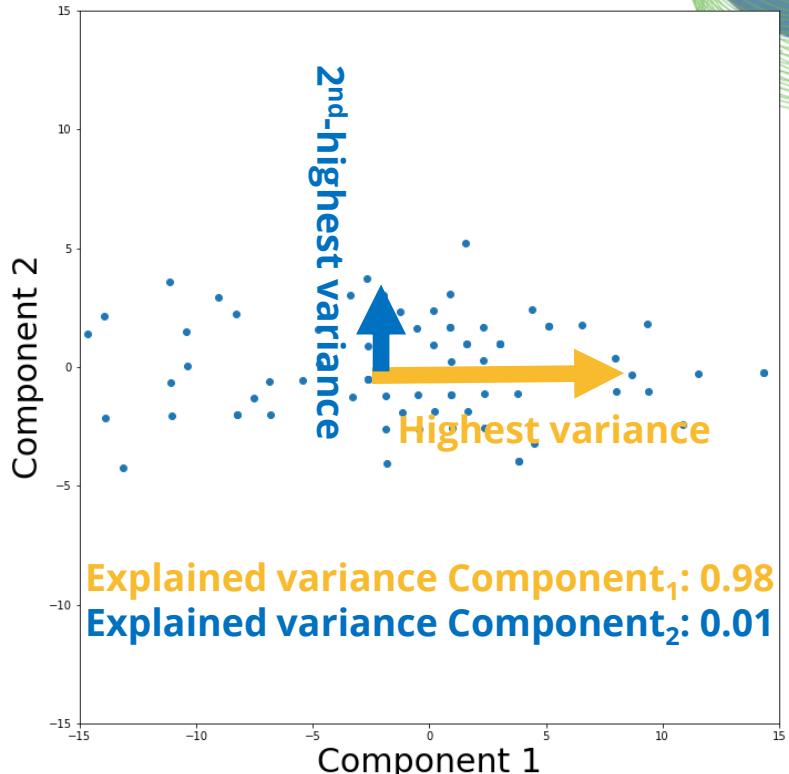
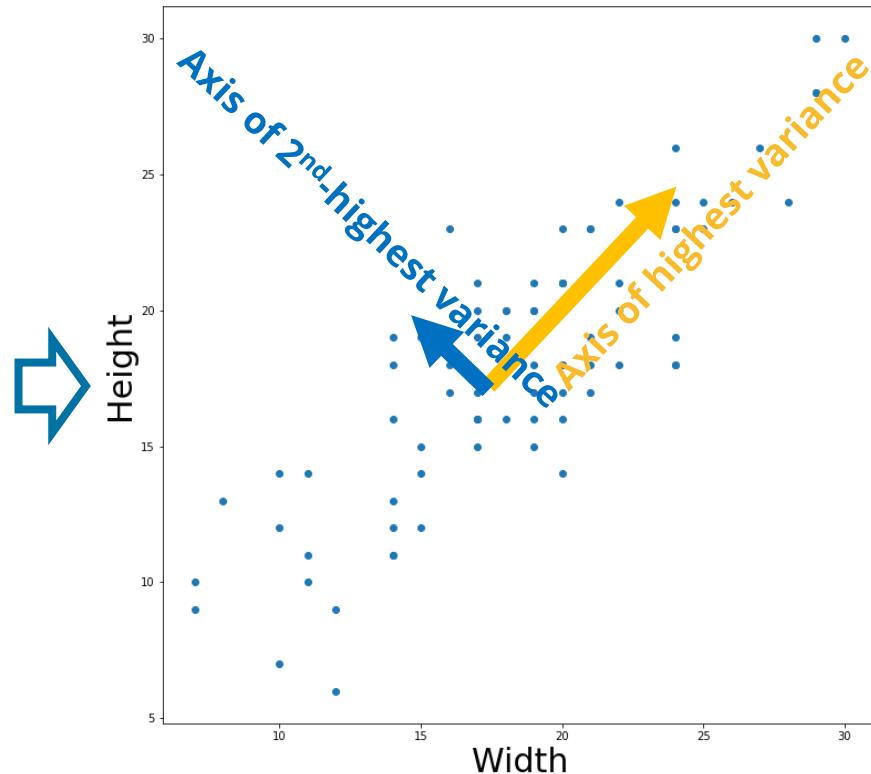
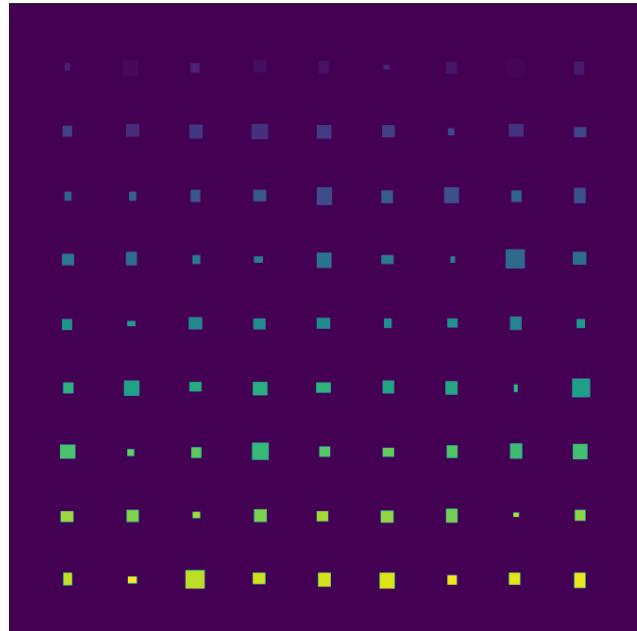
If you have no idea -> unsupervised machine learning

- Dimensionality reduction
- Clustering

PCA: Principal Component Analysis

Decomposes data into linear combinations of features that explain the highest variance

Example: Squares of different size

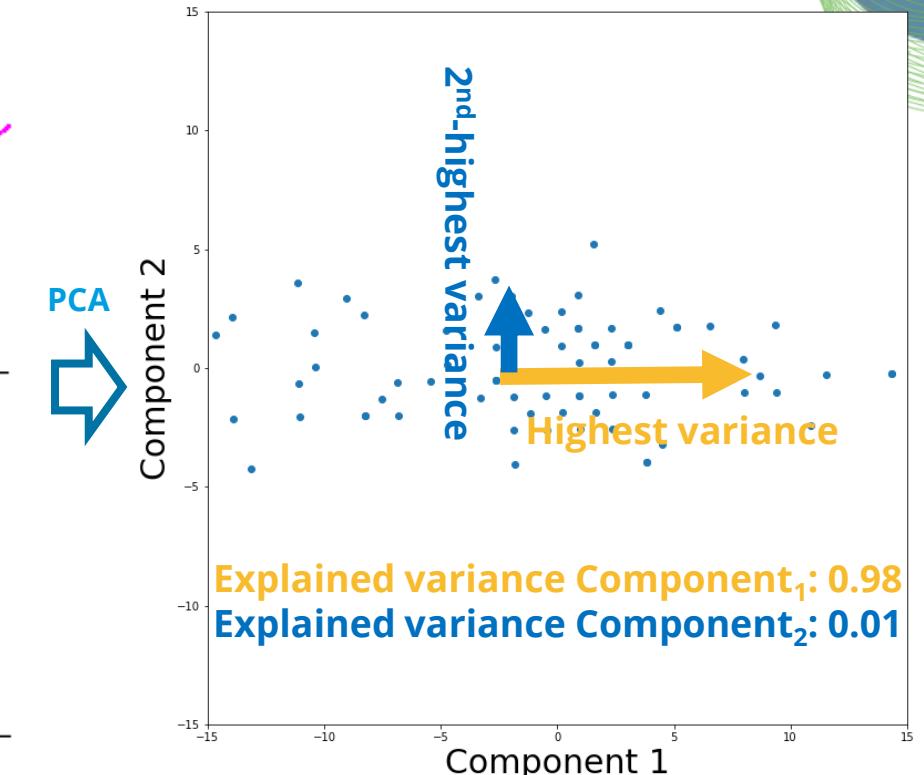
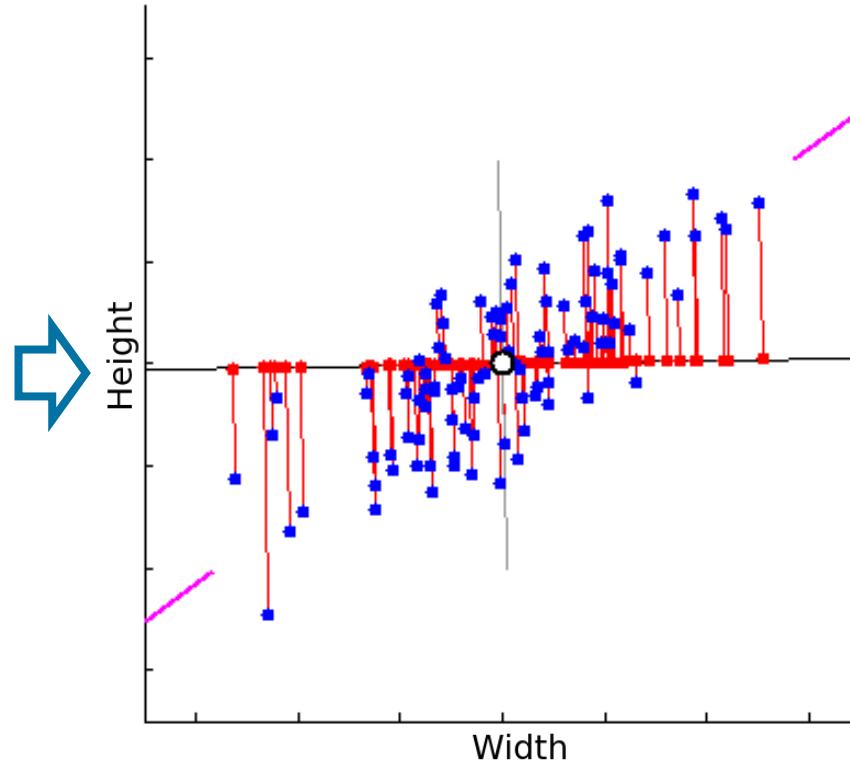
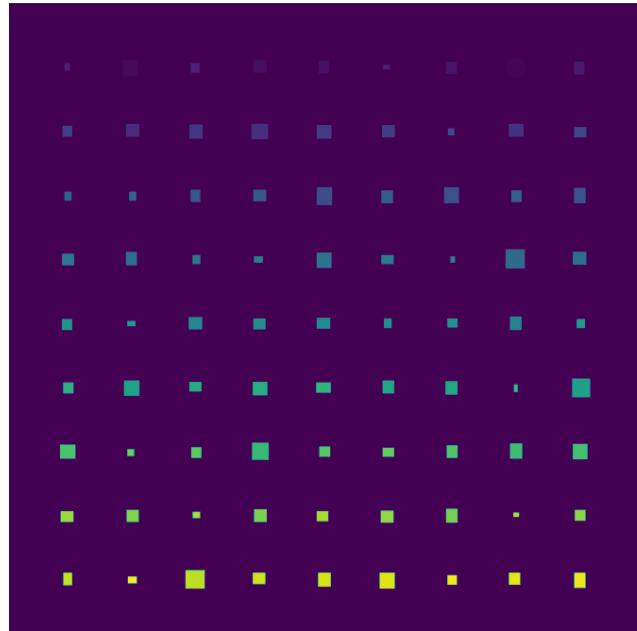


→ PCA transforms width/height measurements into a coordinate system that explains existing variance better

PCA: Principal Component Analysis

Decomposes data into linear combinations of features that explain the highest variance

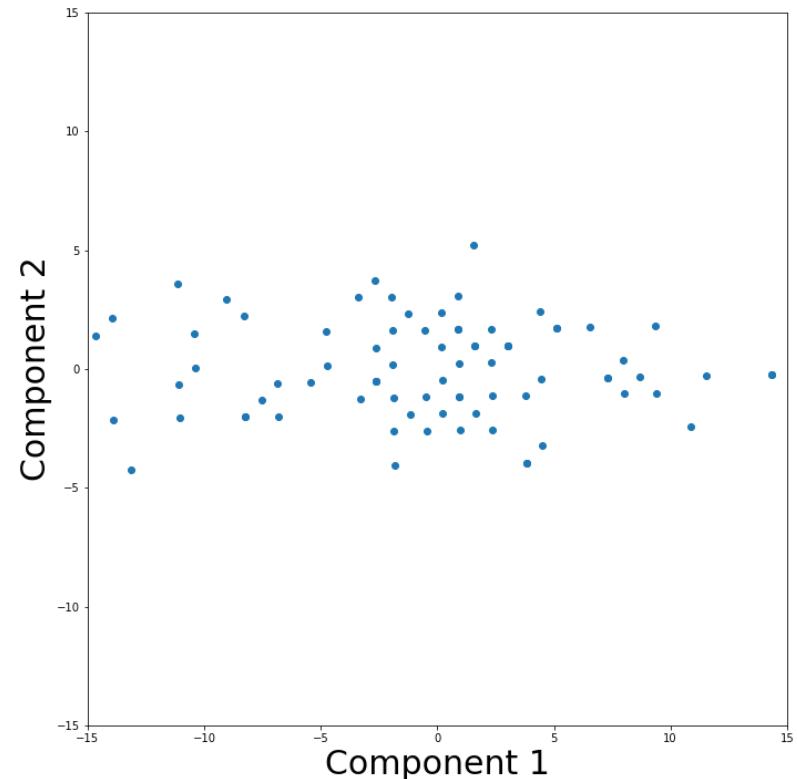
Example: Squares of different size



→ PCA transforms width/height measurements into a coordinate system that explains existing variance better

Embeddings

- N-dimensional latent space
- Axes typically have no meaningful/physical name (PCA1, UMAP1, ...) and no physical unit
- Allow representing complex measurements, things, relationships in numeric space.
- Example:
 - You measure amplitude, frequency, wavelength, etc.,
 - derive a 2D-embedding from it,
 - to visualize the data or
 - to better process data



PCA in Python: `sklearn.decomposition.PCA`

- Import package

```
from sklearn.decomposition import PCA
```

- Apply PCA

```
pca = PCA(n_components=2)  
pca.fit(standardized_data)
```

- Transform data into new coordinate system

```
transformed_data = pca.transform(data)
```

Important!

Always check the explained variance along the PCA component axes!

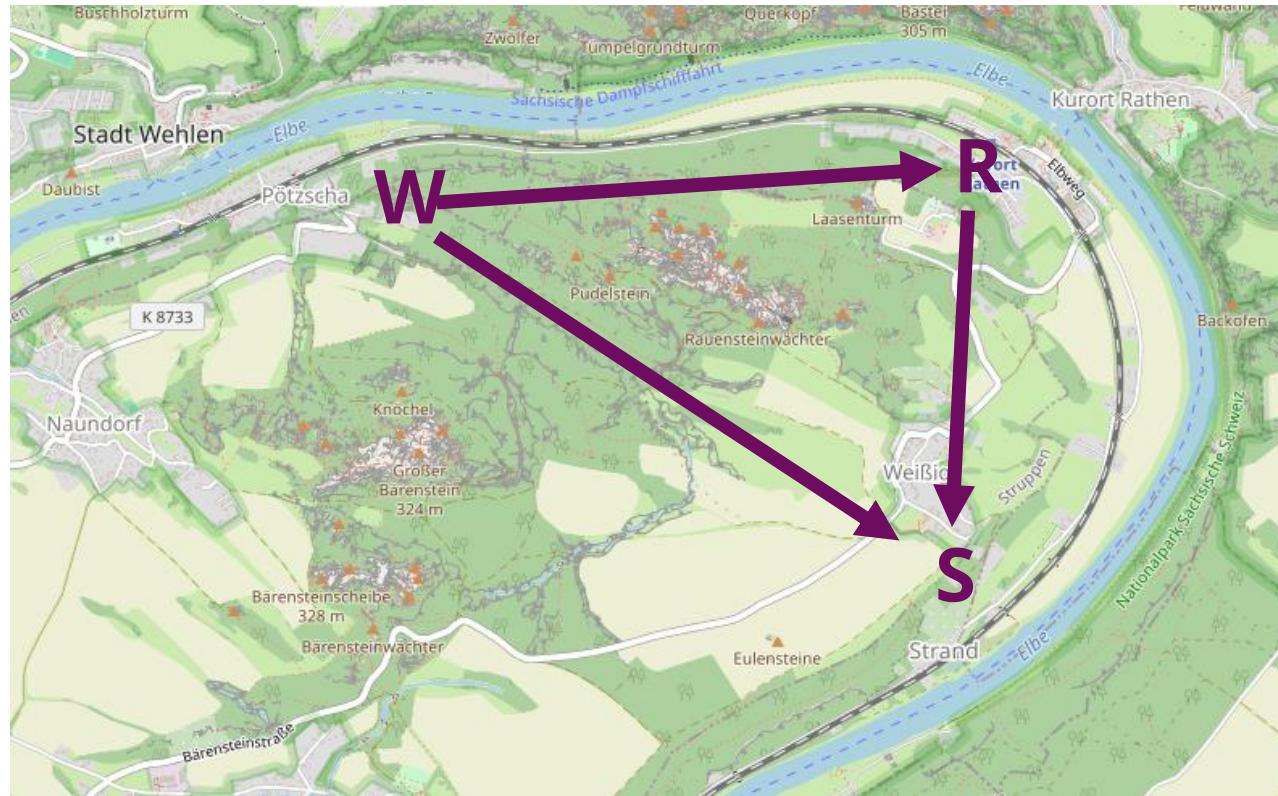
```
pca.explained_variance_ratio_
```

```
array([0.98773142, 0.01226858])
```

The screenshot shows the official scikit-learn website. At the top, there's a navigation bar with links for Install, User Guide, API, Examples, Community, and More. Below the header, the text "scikit-learn Machine Learning in Python" is displayed, along with buttons for Getting Started, Release Highlights for 1.1, and GitHub. To the right, there's a list of bullet points: "Simple and efficient tools for predictive data analysis", "Accessible to everybody, and reusable in various contexts", "Built on NumPy, SciPy, and matplotlib", and "Open source, commercially usable - BSD license". The main content area is divided into several sections: "Classification" (with a sub-section for "Applications: Spam detection, image recognition." and "Algorithms: SVM, nearest neighbors, random forest, and more..."), "Regression" (with a sub-section for "Applications: Drug response, Stock prices. Algorithms: SVR, nearest neighbors, random forest, and more..."), "Clustering" (with a sub-section for "Applications: Customer segmentation, Grouping experiment outcomes Algorithms: k-means, spectral clustering, mean-shift, and more..."), "Dimensionality reduction" (with a sub-section for "Reducing the number of random variables to consider." and "Applications: Visualization, Increased efficiency"), "Model selection" (with a sub-section for "Comparing, validating and choosing parameters and models." and "Applications: Improved accuracy via parameter tuning"), and "Preprocessing" (with a sub-section for "Feature extraction and normalization." and "Applications: Transforming input data such as text for use with machine learning algorithms"). Each section contains a small image illustrating its purpose.

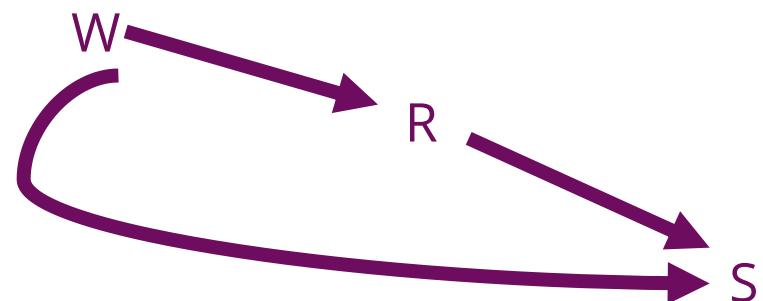
Non-Euclidian spaces

Not all features might be distances



Use travel time between W and S as metric for distance

→ Travelling from **Wehlen** to **Strand** by bike is probably faster if you make a detour through **Rathen**



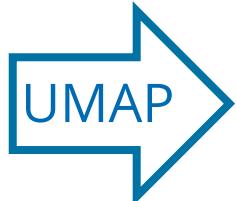
Uniform Manifold Approximation Projection (UMAP)

Structural, hierarchical, non-linear transformation

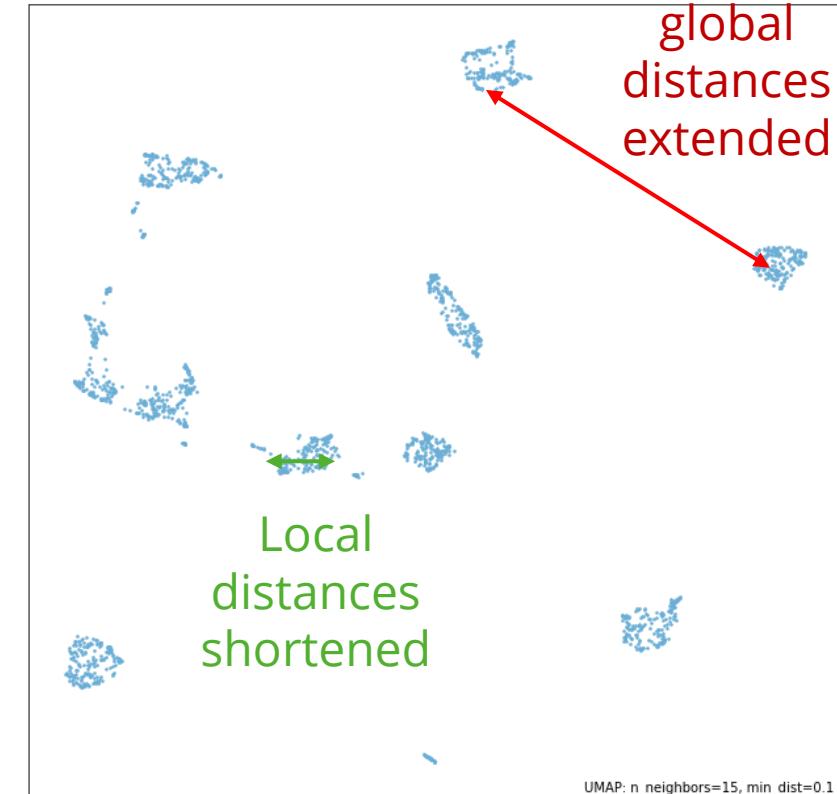
Modifies density of data points.

	count	mean	std
label	44.0	22.500000	12.845233
area	44.0	401.863636	202.852288
bbox_area	44.0	542.750000	295.106376
equivalent_diameter	44.0	21.781085	6.174086
convex_area	44.0	423.295455	216.613747
max_intensity	44.0	234.909091	17.517856
mean_intensity	44.0	190.116971	15.034153
min_intensity	44.0	128.000000	0.000000
extent	44.0	0.758804	0.063276
local_centroid-0	44.0	11.439824	4.126230
local_centroid-1	44.0	10.138666	3.491815
solidity	44.0	0.953153	0.024749
feret_diameter_max	44.0	26.382434	8.915046
major_axis_length	44.0	25.876797	9.591558
minor_axis_length	44.0	18.872898	5.158791
orientation	44.0	0.053057	0.691430
eccentricity	44.0	0.600434	0.165688
standard_deviation_intensity	44.0	29.556705	5.507399
aspect_ratio	44.0	1.374342	0.397611
roundness	44.0	0.762889	0.156695
circularity	44.0	0.918858	0.133208

Many dimensions



UMAP 2



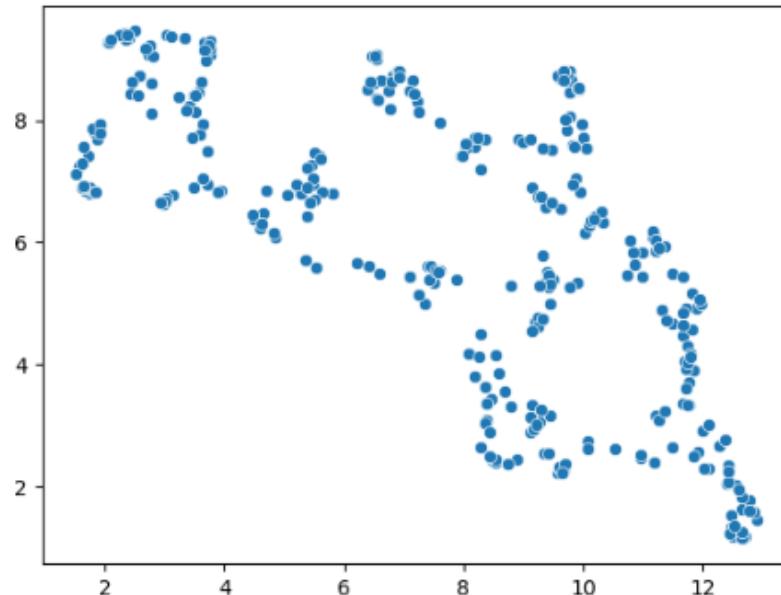
UMAP 1

Uniform Manifold Approximation Projection (UMAP)

Non-deterministic algorithm: You execute it twice, you get different results.

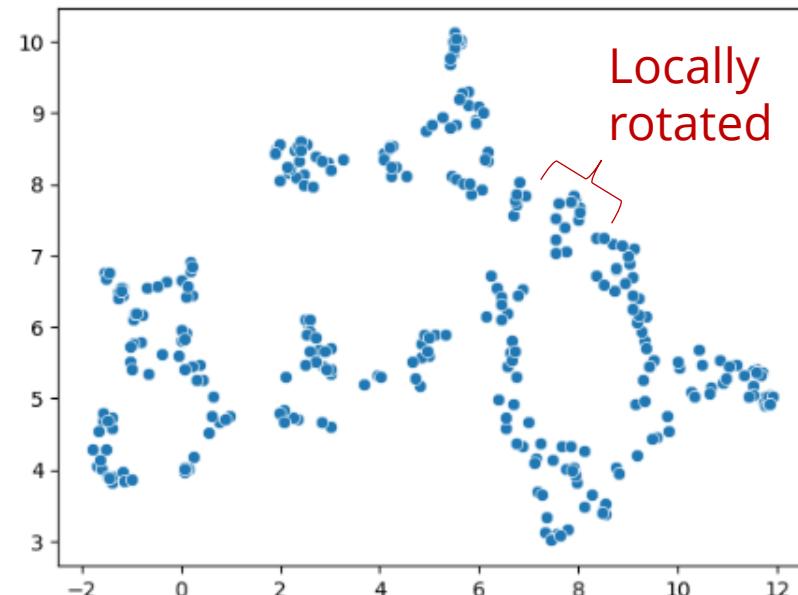
```
[11]: reducer = umap.UMAP()  
embedding2 = reducer.fit_transform(scaled_statistics)  
  
seaborn.scatterplot(x=embedding2[:, 0],  
                     y=embedding2[:, 1])
```

```
[11]: <AxesSubplot: >
```

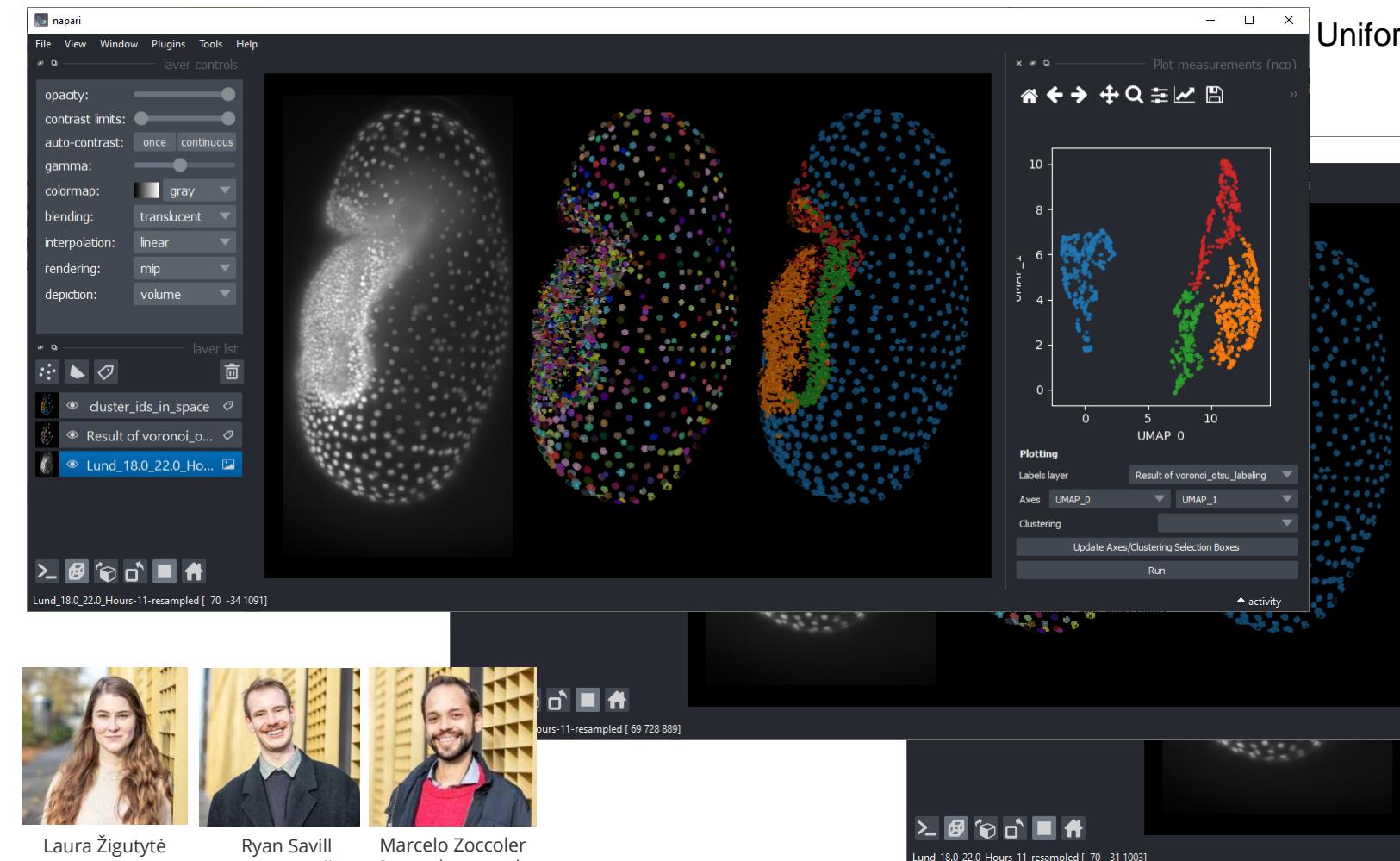


```
[12]: reducer = umap.UMAP()  
embedding2 = reducer.fit_transform(scaled_statistics)  
  
seaborn.scatterplot(x=embedding2[:, 0],  
                     y=embedding2[:, 1])
```

```
[12]: <AxesSubplot: >
```



Dimensionality reduction



Uniform manifold approximation and projection (UMAP)

t-distributed stochastic neighbor embedding (t-SNE)

Principal component analysis (PCA)



UMAP in Python

Selecting columns from a pandas DataFrame

```
[8]: measurements.describe().T
```

	count	mean	std
label	44.0	22.500000	12.845233
area	44.0	401.863636	202.852288
bbox_area	44.0	542.750000	295.106376
equivalent_diameter	44.0	21.781085	6.174086
convex_area	44.0	423.295455	216.613747
max_intensity	44.0	234.909091	17.517856
mean_intensity	44.0	190.116971	15.034153
min_intensity	44.0	128.000000	0.000000
extent	44.0	0.758804	0.063276
local_centroid-0	44.0	11.439824	4.126230
local_centroid-1	44.0	10.138666	3.491815
solidity	44.0	0.953153	0.024749
feret_diameter_max	44.0	26.382434	8.915046
major_axis_length	44.0	25.876797	9.591558
minor_axis_length	44.0	18.872898	5.158791
orientation	44.0	0.053057	0.691430
eccentricity	44.0	0.600434	0.165688
standard_deviation_intensity	44.0	29.556705	5.507399
aspect_ratio	44.0	1.374342	0.397611
roundness	44.0	0.762889	0.156695
circularity	44.0	0.918858	0.133288

```
[9]: selected_measurements = measurements[[
    'area',
    'equivalent_diameter',
    'convex_area',
    'max_intensity',
    'mean_intensity',
    'min_intensity',
    'extent',
    'solidity',
    'feret_diameter_max',
    'major_axis_length',
    'minor_axis_length',
    'eccentricity',
    'standard_deviation_intensity',
    'aspect_ratio',
    'roundness',
    'circularity']]
```

```
[9]: selected_measurements.describe().T
```

Select reasonable features

	count	mean	std
area	44.0	401.863636	202.852288
equivalent_diameter	44.0	21.781085	6.174086
convex_area	44.0	423.295455	216.613747
max_intensity	44.0	234.909091	17.517856
mean_intensity	44.0	190.116971	15.034153
min_intensity	44.0	128.000000	0.000000
extent	44.0	0.758804	0.063276
solidity	44.0	0.953153	0.024749
feret_diameter_max	44.0	26.382434	8.915046
major_axis_length	44.0	25.876797	9.591558
minor_axis_length	44.0	18.872898	5.158791
eccentricity	44.0	0.600434	0.165688
standard_deviation_intensity	44.0	29.556705	5.507399
aspect_ratio	44.0	1.374342	0.397611
roundness	44.0	0.762889	0.156695
circularity	44.0	0.918858	0.133288

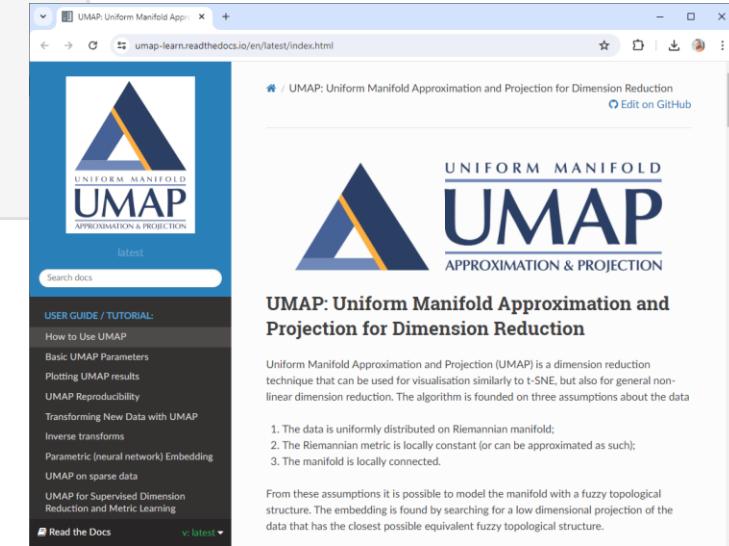
UMAP in Python

```
[10]: # configure UMAP algorithm
umap = UMAP(n_neighbors=5, n_components=2)

# apply algorithm
transformed_data = umap.fit_transform(selected_measurements.values.tolist())

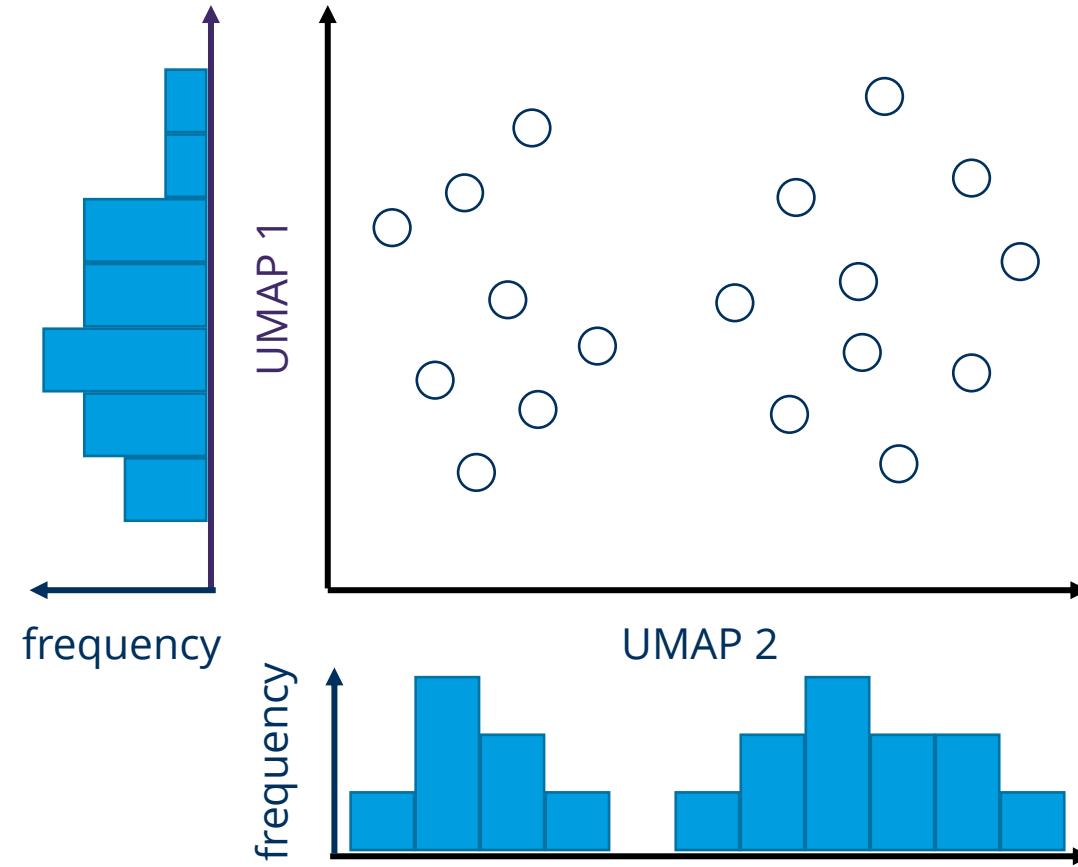
# store results back in table
measurements['UMAP0'] = transformed_data[:,0]
measurements['UMAP1'] = transformed_data[:,1]
```

Data conversion



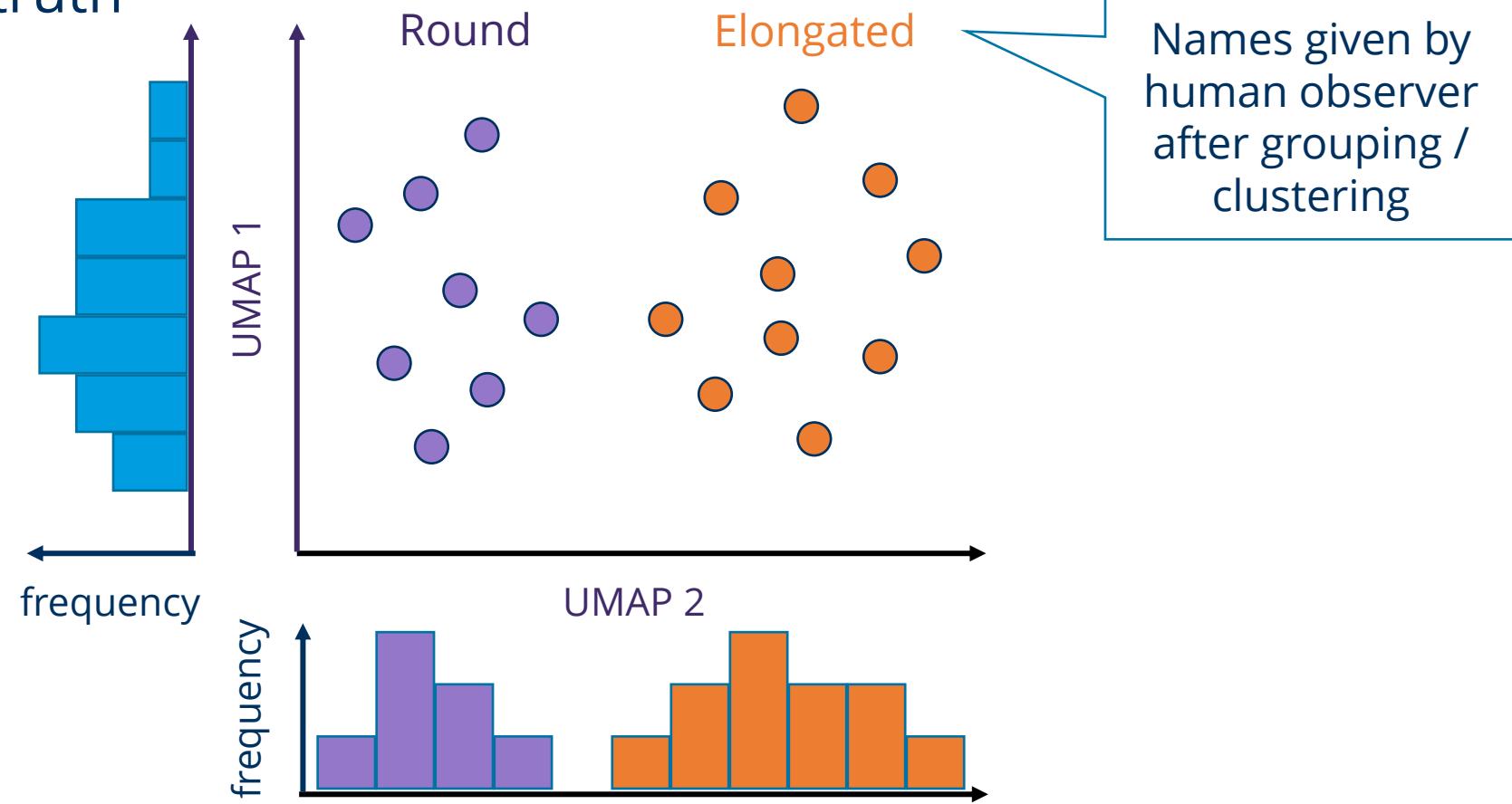
Clustering

Unsupervised machine learning may include grouping objects without given ground truth



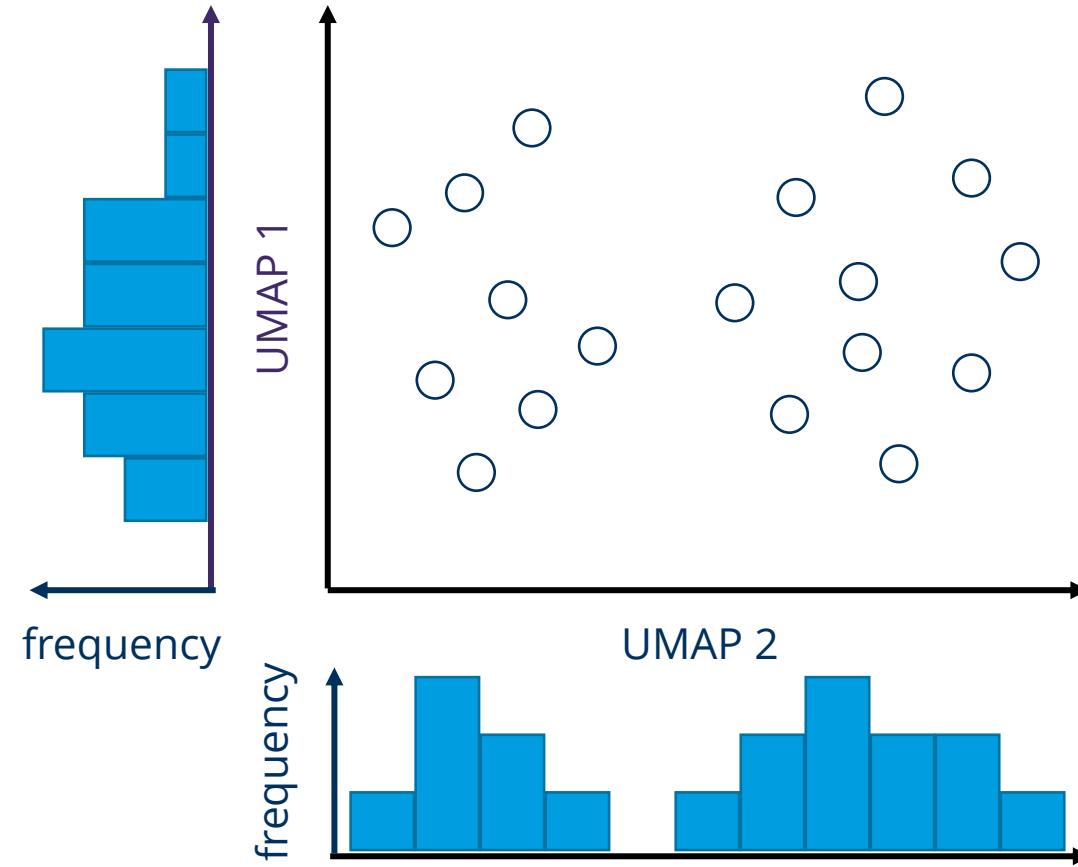
Clustering

Unsupervised machine learning may include grouping objects without given ground truth



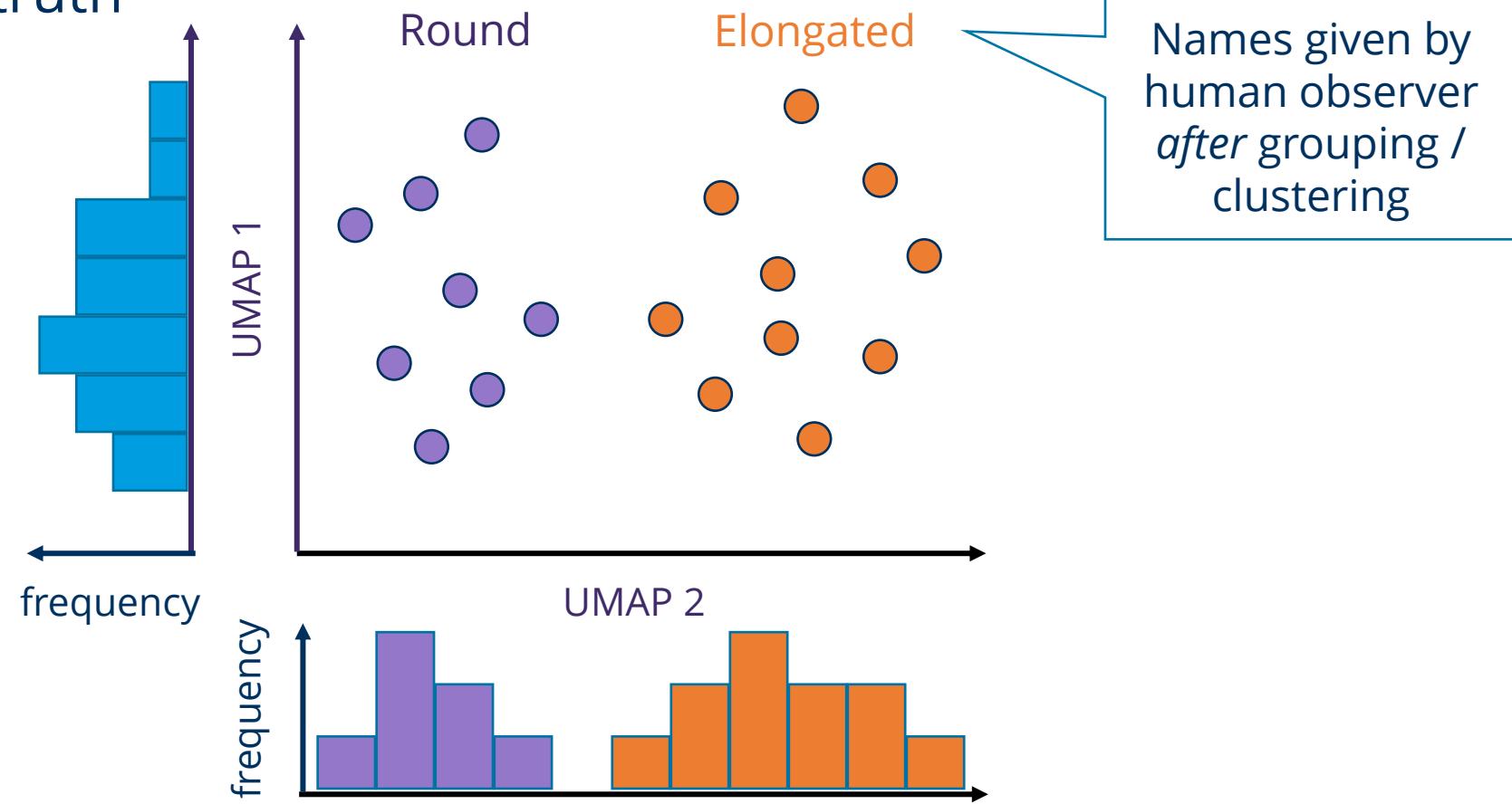
Clustering

Unsupervised machine learning may include grouping objects without given ground truth



Clustering

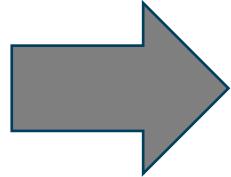
Unsupervised machine learning may include grouping objects without given ground truth



Clustering: Manual

[9]:

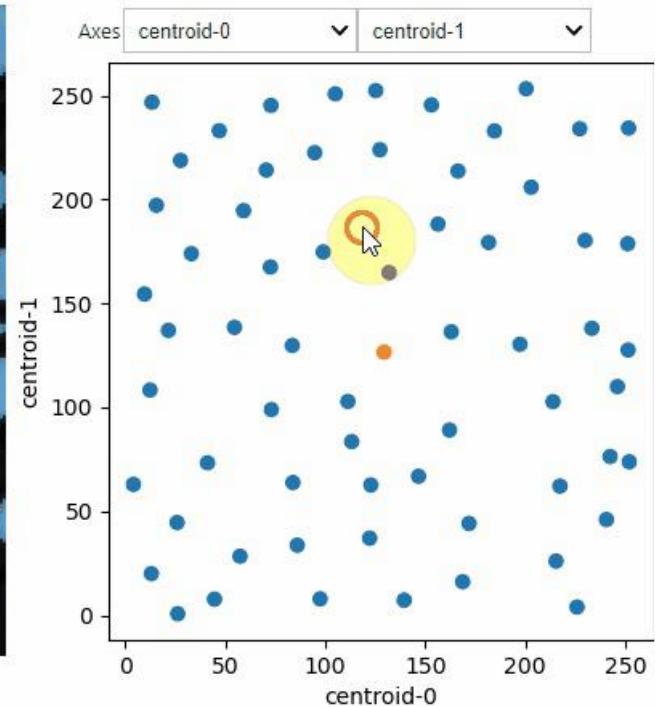
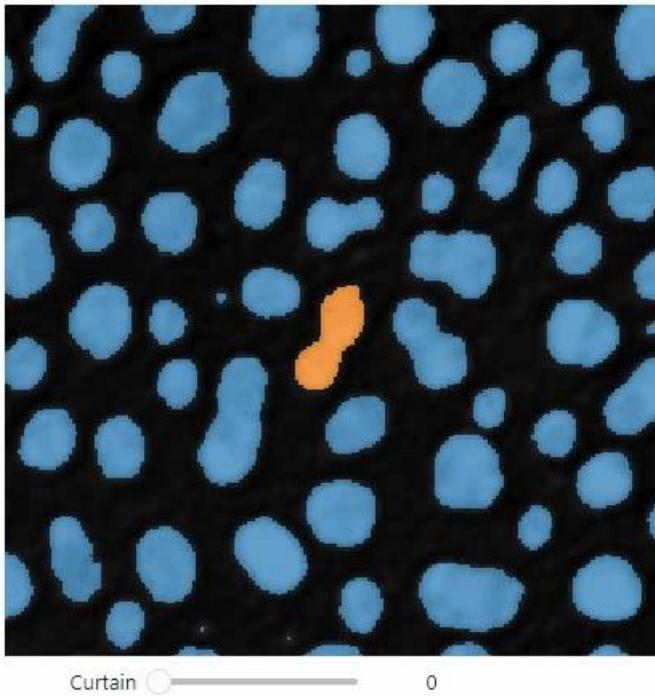
	count	mean	std
area	44.0	401.863636	202.852288
equivalent_diameter	44.0	21.781085	6.174086
convex_area	44.0	423.295455	216.613747
max_intensity	44.0	234.909091	17.517856
mean_intensity	44.0	190.116971	15.034153
min_intensity	44.0	128.000000	0.000000
extent	44.0	0.758804	0.063276
solidity	44.0	0.953153	0.024749
feret_diameter_max	44.0	26.382434	8.915046
major_axis_length	44.0	25.876797	9.591558
minor_axis_length	44.0	18.872898	5.158791
eccentricity	44.0	0.600434	0.165688
standard_deviation_intensity	44.0	29.556705	5.507399
aspect_ratio	44.0	1.374342	0.397611
roundness	44.0	0.762889	0.156695
circularity	44.0	0.918858	0.133288



[6]:

```
stackview.clusterplot(image=image,
                      labels=labeled_image,
                      df=df,
                      column_x="area",
                      column_y="aspect_ratio",
                      zoom_factor=1.6,
                      alpha=0.7)
```

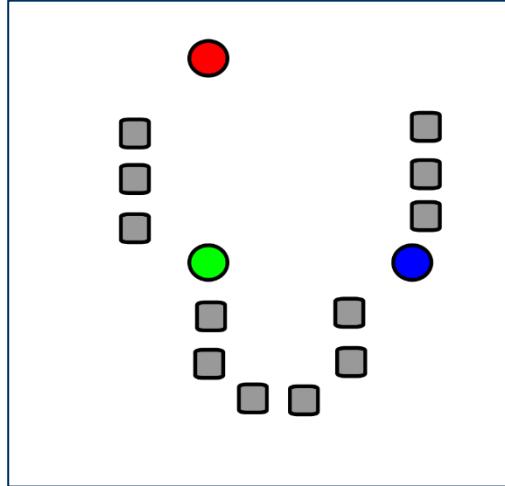
[6]:



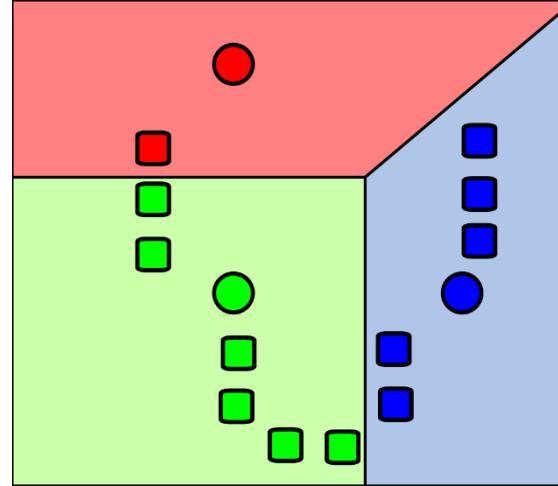
K-Means Clustering

Clustering algorithm, where you *only* need to specify the number of clusters.

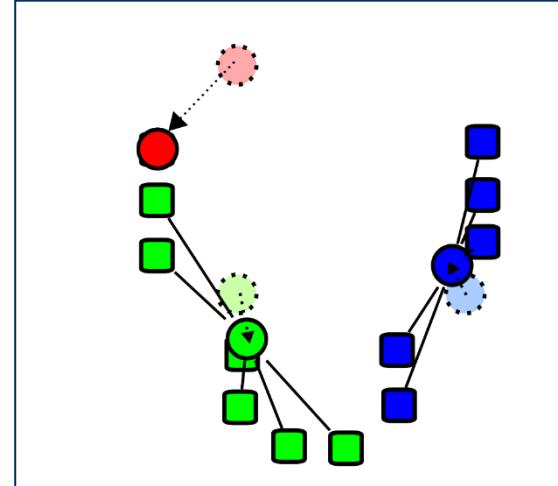
Step1: Random initialization of cluster centers



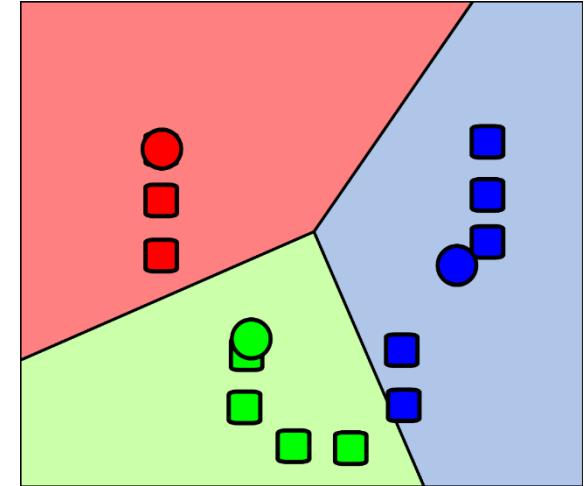
Step2: Tessellation of space into cluster regions



Step3: Replace cluster center with centroids

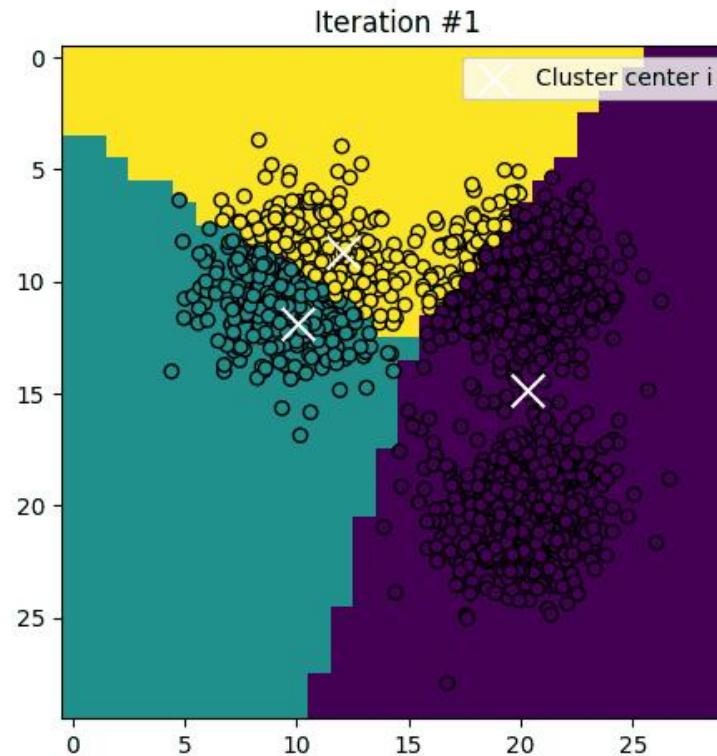


Step4: Repeat 2&3 until convergence



K-Means Clustering

Clustering algorithm, where you *only* need to specify the number of clusters.



K-Means Clustering

Goal: group data points into k groups so that variance within group is minimal.

In Python:

```
from sklearn import cluster
```

Create

```
clusterer = cluster.KMeans(n_clusters=3)
clusterer.fit(X)
```

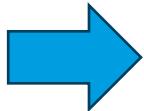
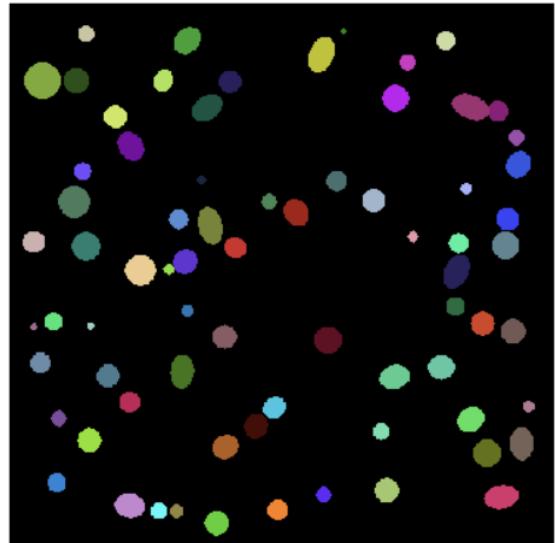
Predict

```
predicted_class = clusterer.predict(X)
```

Walk-through: Data Exploration

Goal: Understand shape measurements

Data: Shape measurements from *randomly shaped blobs*.



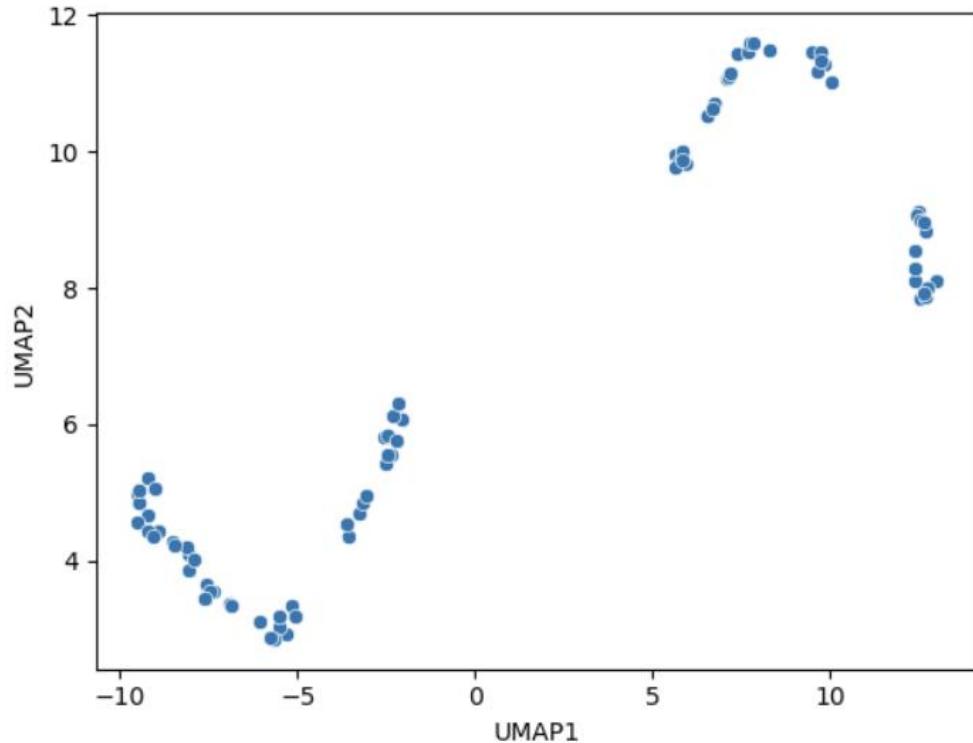
	label	area	perimeter	minor_axis_length	major_axis_length	circularity	solidity	aspect_ratio	elongation
0	1	97.0	32.970563	11.092860	11.092860	1.121318	0.788288	1.000000	0.000000
1	2	285.0	60.284271	19.052651	19.052651	0.985477	0.785116	1.000000	0.000000
2	3	473.0	79.597980	21.823280	27.594586	0.938138	0.785448	1.264456	0.209146
3	4	321.0	63.112698	19.033334	21.456036	1.012701	0.786033	1.127287	0.112915
4	5	407.0	72.769553	22.155138	23.384406	0.965839	0.785586	1.055485	0.052568
...									

Walk-through: Data Exploration

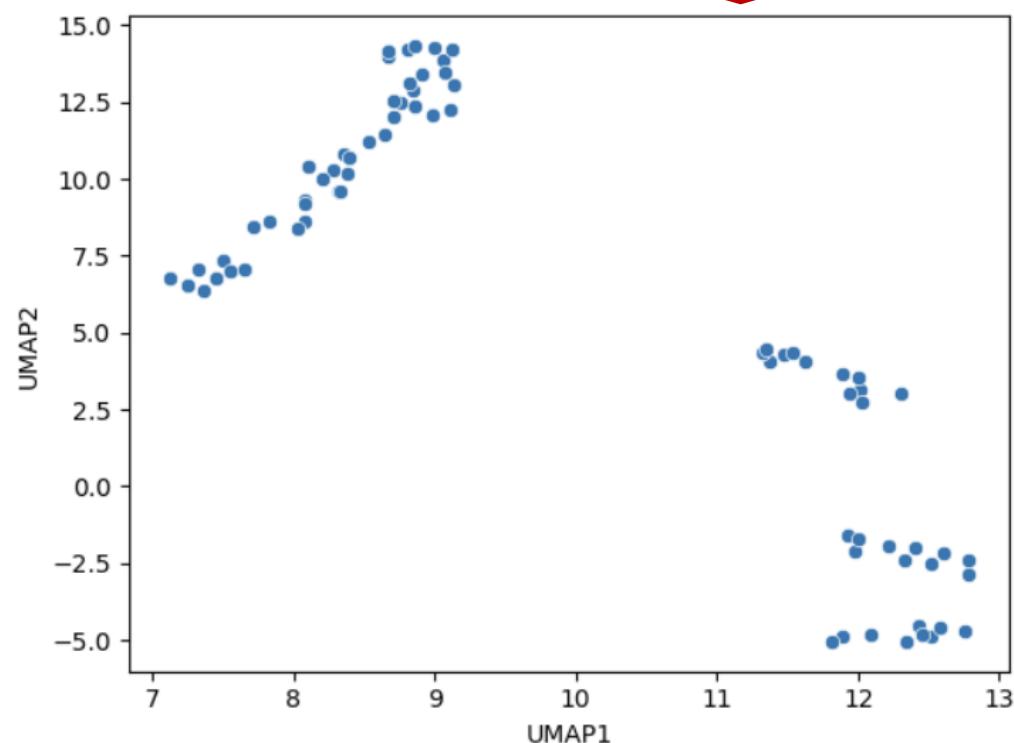
Step 1: Dimensionality reduction (UMAP)

Observation: There appear to be 2 *distinct groups*

Run 1:



Run 2:



Pinning the random seed is no solution to this general problem.

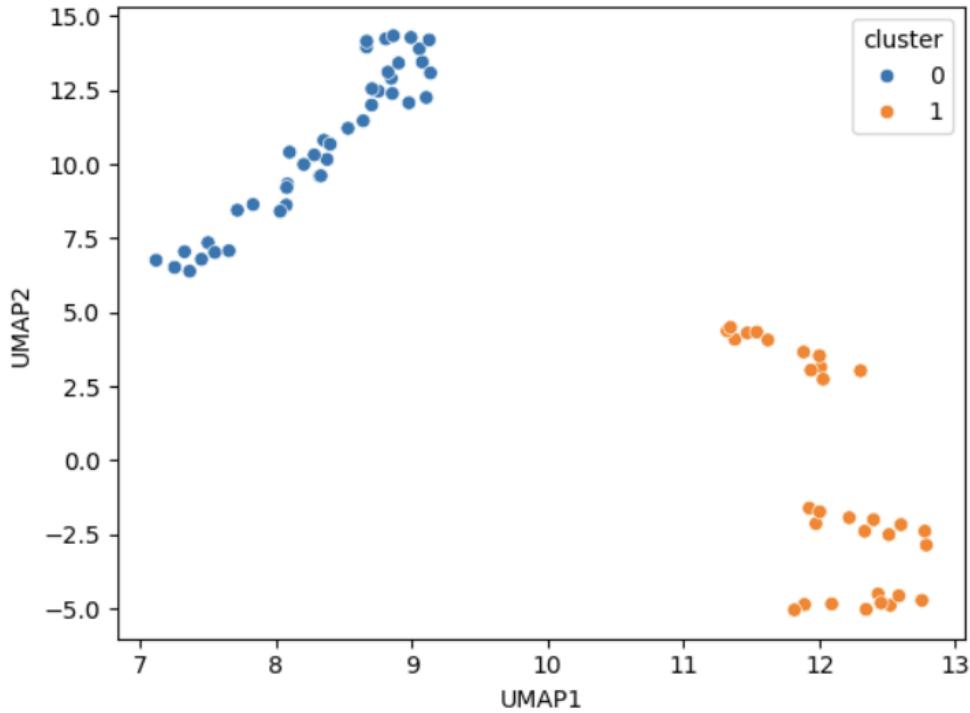
Beware: UMAPs are non-deterministic. Different runs lead to different results.

Walk-through: Data Exploration

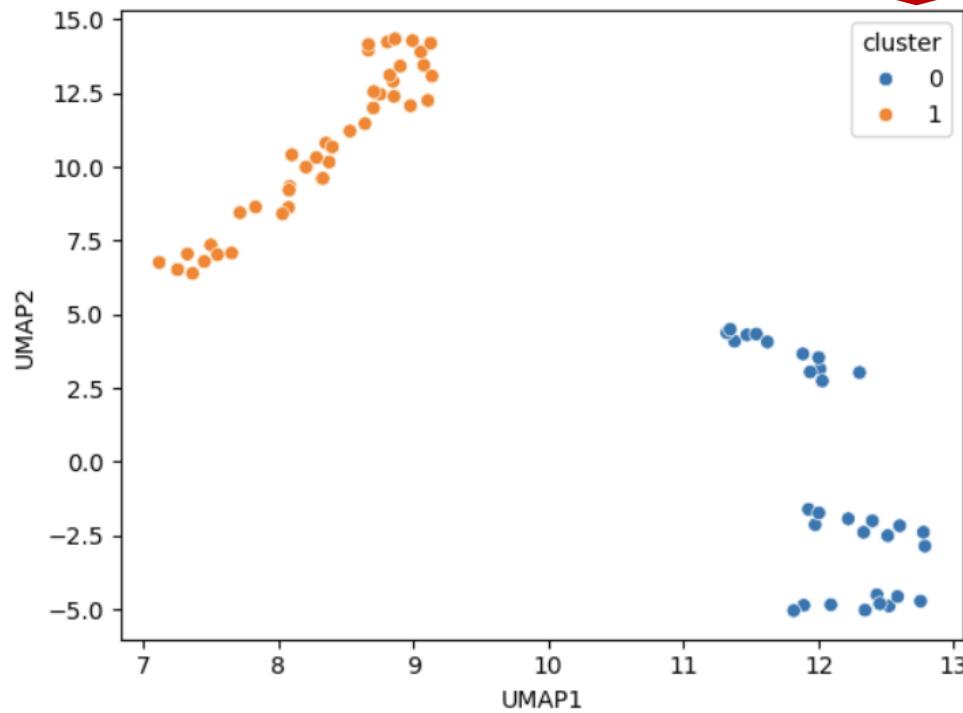
Step 2: Clustering data into 2 clusters

Using K-Means clustering

Run 1:

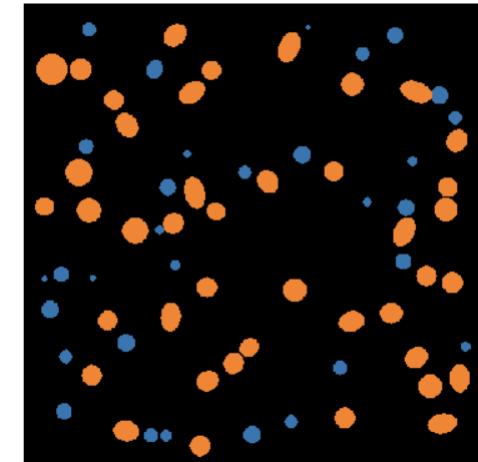


Run 2:



Pinning the random seed is no solution to this general problem.

Beware: Clustering-algorithms are non-deterministic. Different runs lead to different results.



Walk-through: Data Exploration

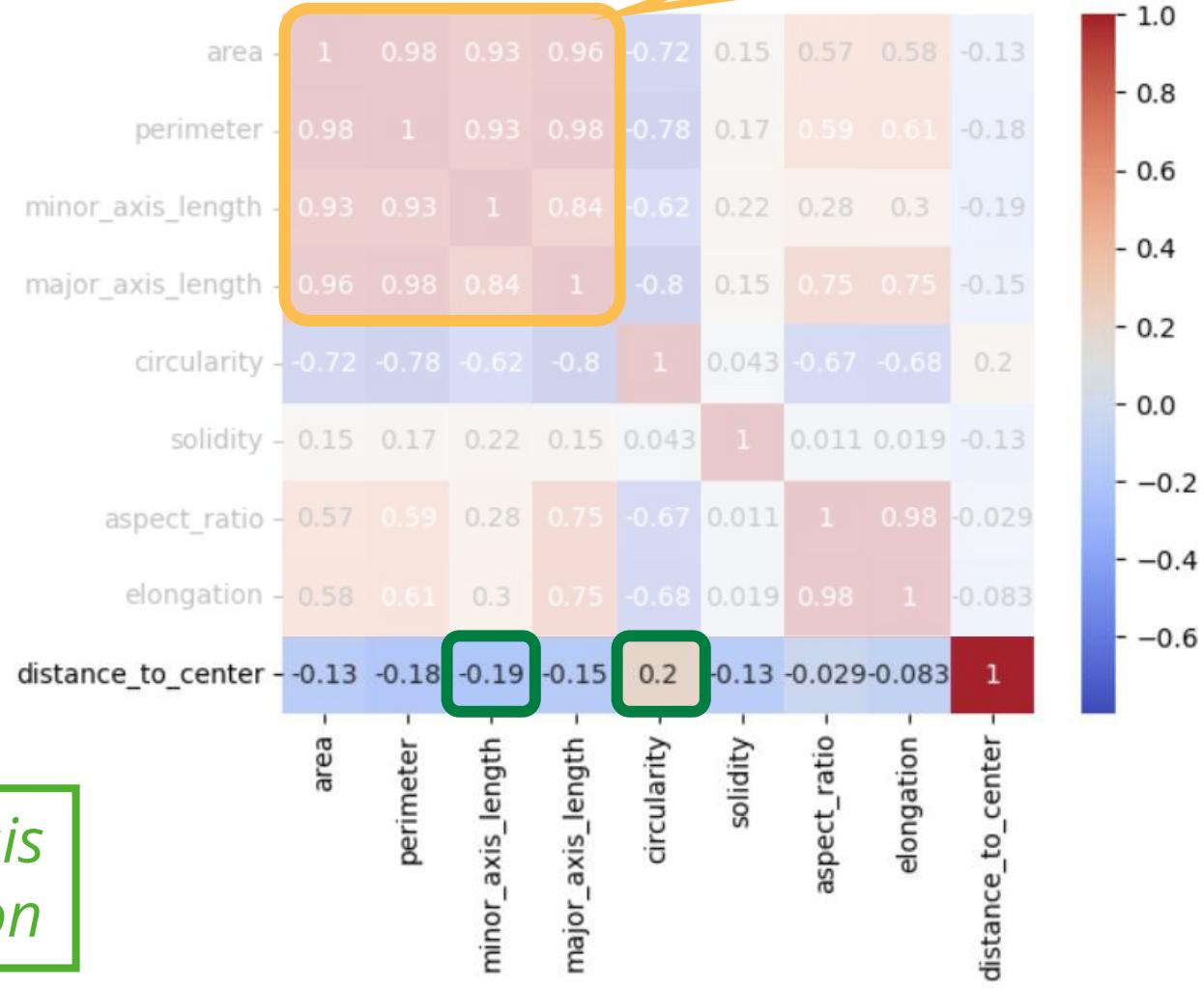
Step 3: Feature selection

Based on correlation
with distance to cluster-centers

Side note: beware of
feature correlation.

Hypothesis:
"Circularity and minor_axis_length
allow to predict round vs.
elongated classification."

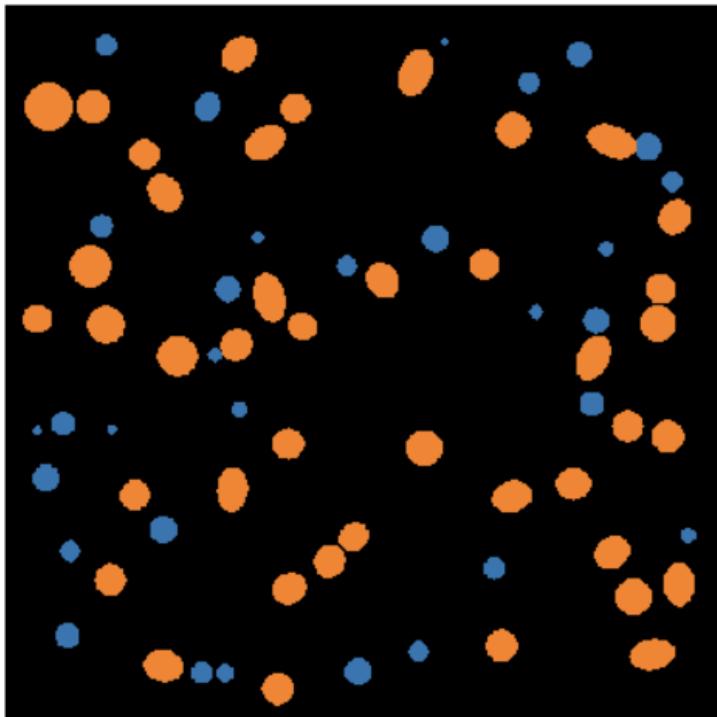
Hypothesis
generation



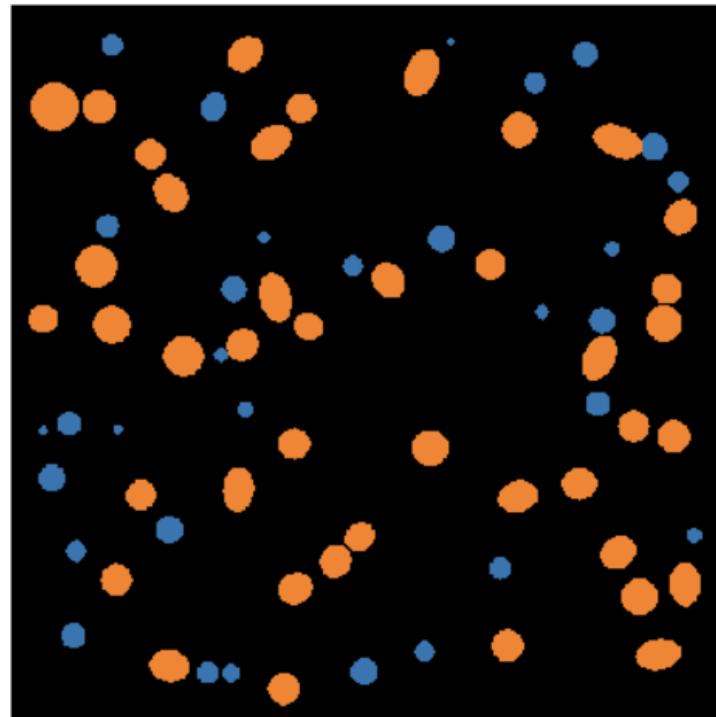
Walk-through: Data Exploration

Step 4: Train a classifier (supervised ML)

Goal: Eliminate non-determinism



Clustering result (non-deterministic)

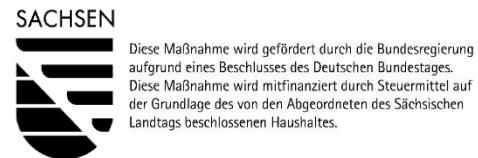


Classification result (deterministic, repeatable)

Supervised and Unsupervised Machine Learning for Bio-image Analysis

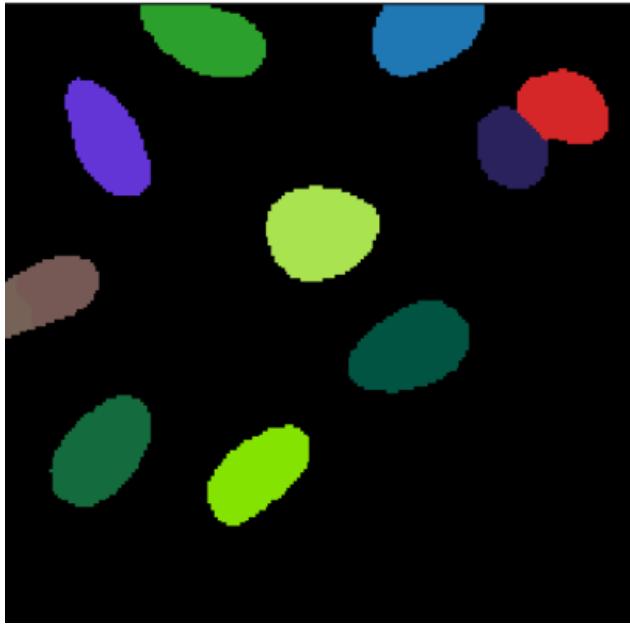
Robert Haase

Reusing materials from Johannes Soltwedel, Till Korten, Johannes Müller, Laura Žigutytė (TU Dresden), Ryan Savill (MPI-CBG), Matthias Täschner (ScaDS.AI/Uni Leipzig) and the Scikit-learn community.



Recap: Terminology

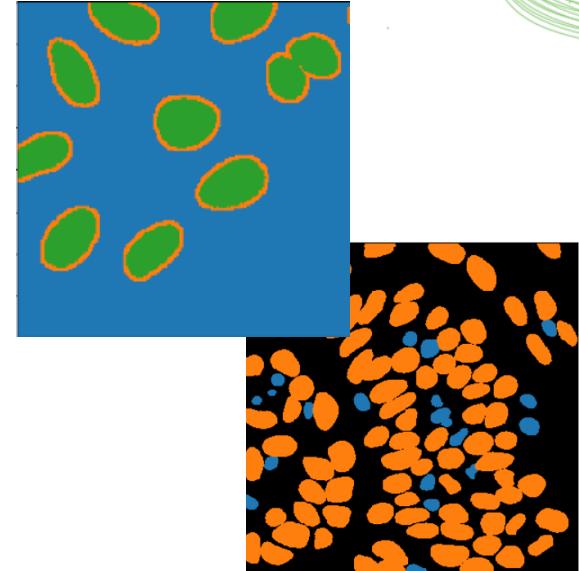
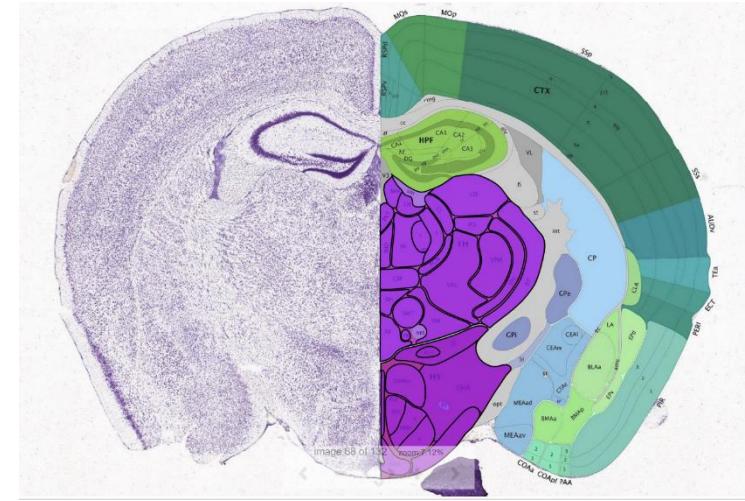
Instance segmentation



Instances:

- Cells, nuclei, cats, dogs, cars, trees

Semantic segmentation

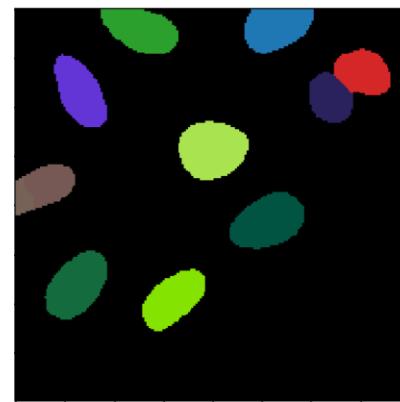


Regions:

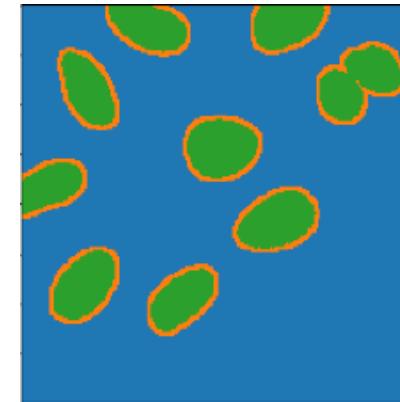
- Anatomical, geographical
- All pixels belonging to the same type of object have the same value

Recap: Terminology

Instance segmentation

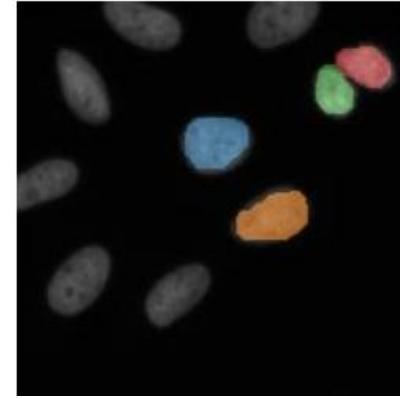


Semantic segmentation



Annotations are typically drawn by humans (e.g. to train machine learning models)

Sparse instance annotation



Sparse semantic annotation

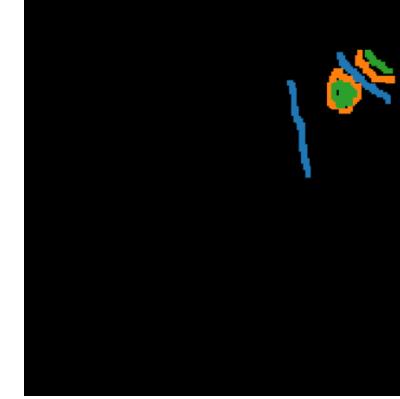


Image segmentation using thresholding

Recap: Finding the right workflow towards a good segmentation takes time

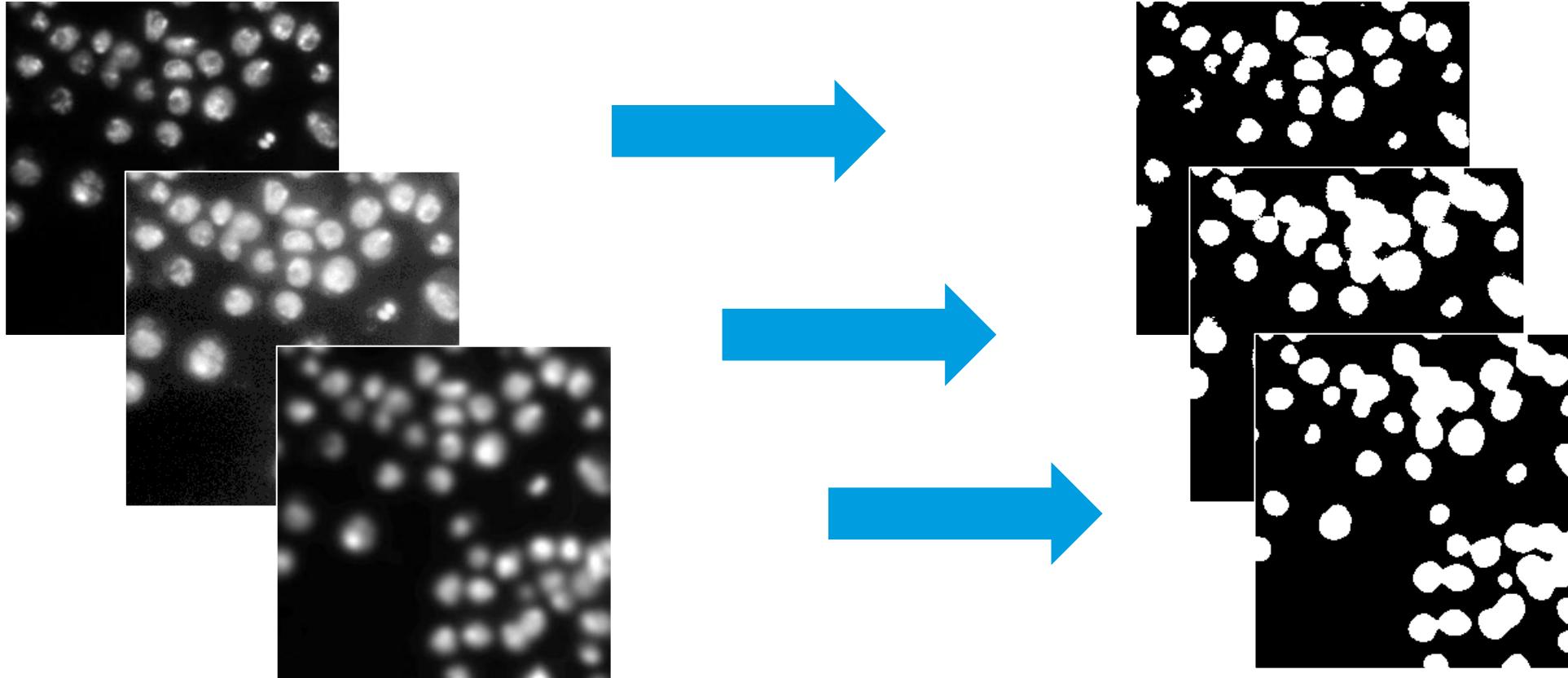


Image segmentation using thresholding

Recap: Combining images, e.g. using Difference of Gaussian (DoG)

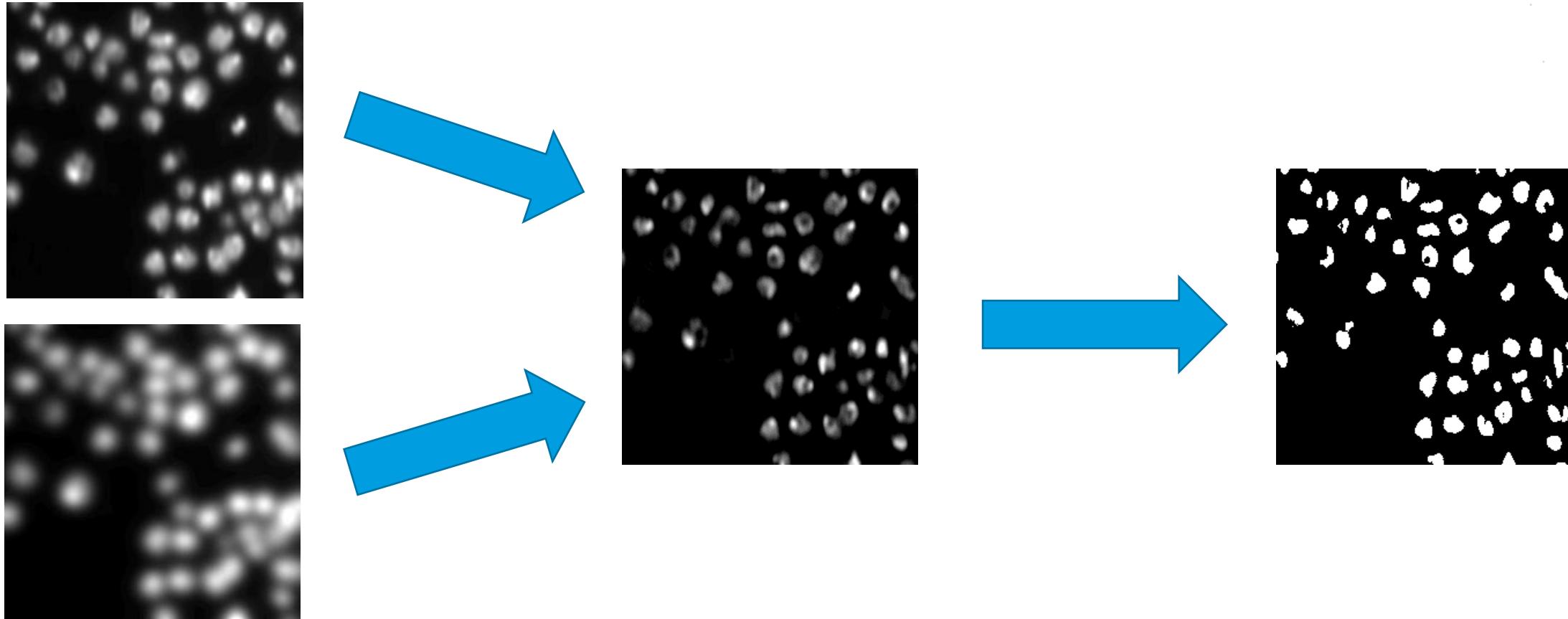
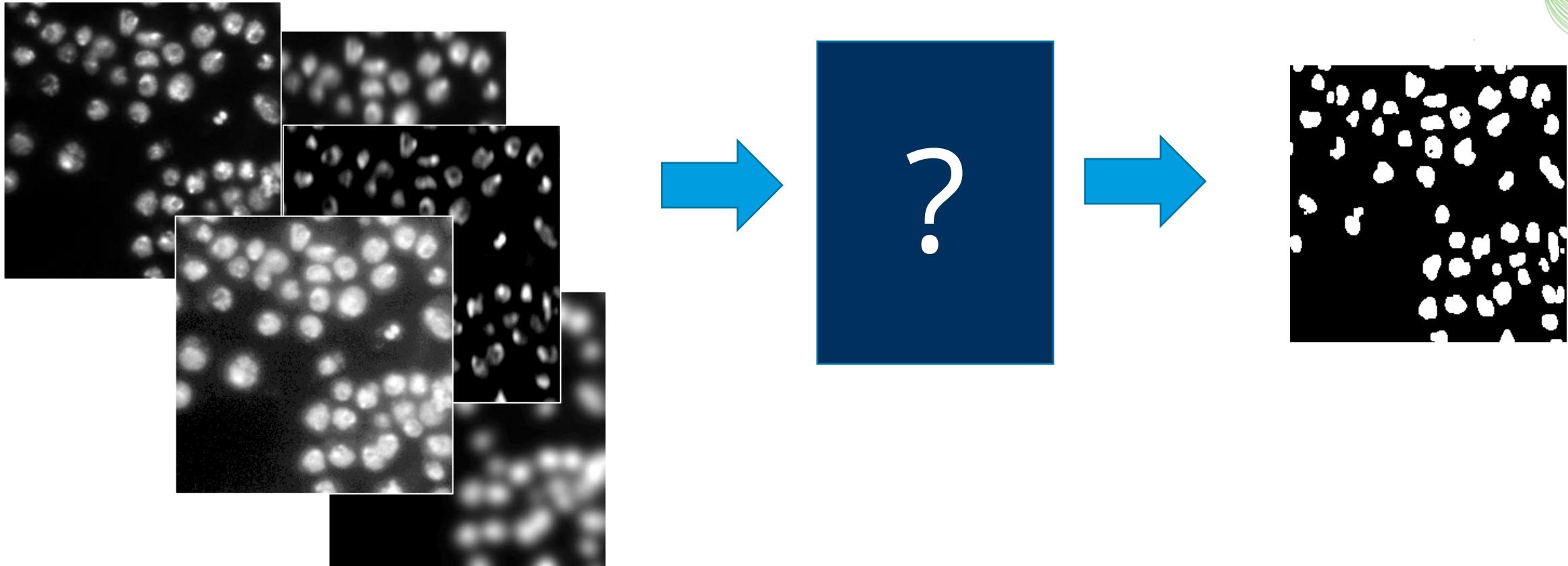


Image segmentation using thresholding

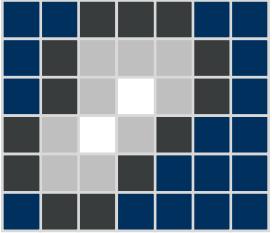
Might there be a technology for optimization which combination of images can be used to get the best segmentation result?



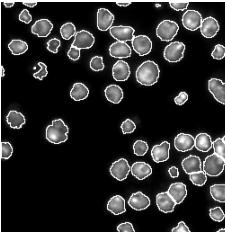
Supervised Machine learning

Automatic construction of predictive models from given data

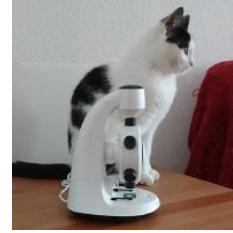
Pixels,



Objects,

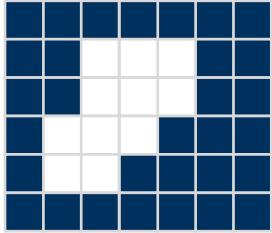


Images



Audio, Text, Measurements, ...

Semantic
Segmentation



Object
classification

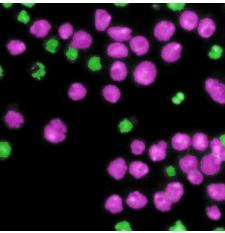
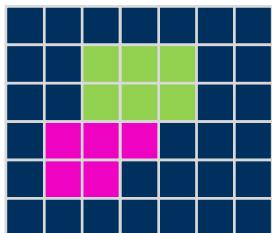


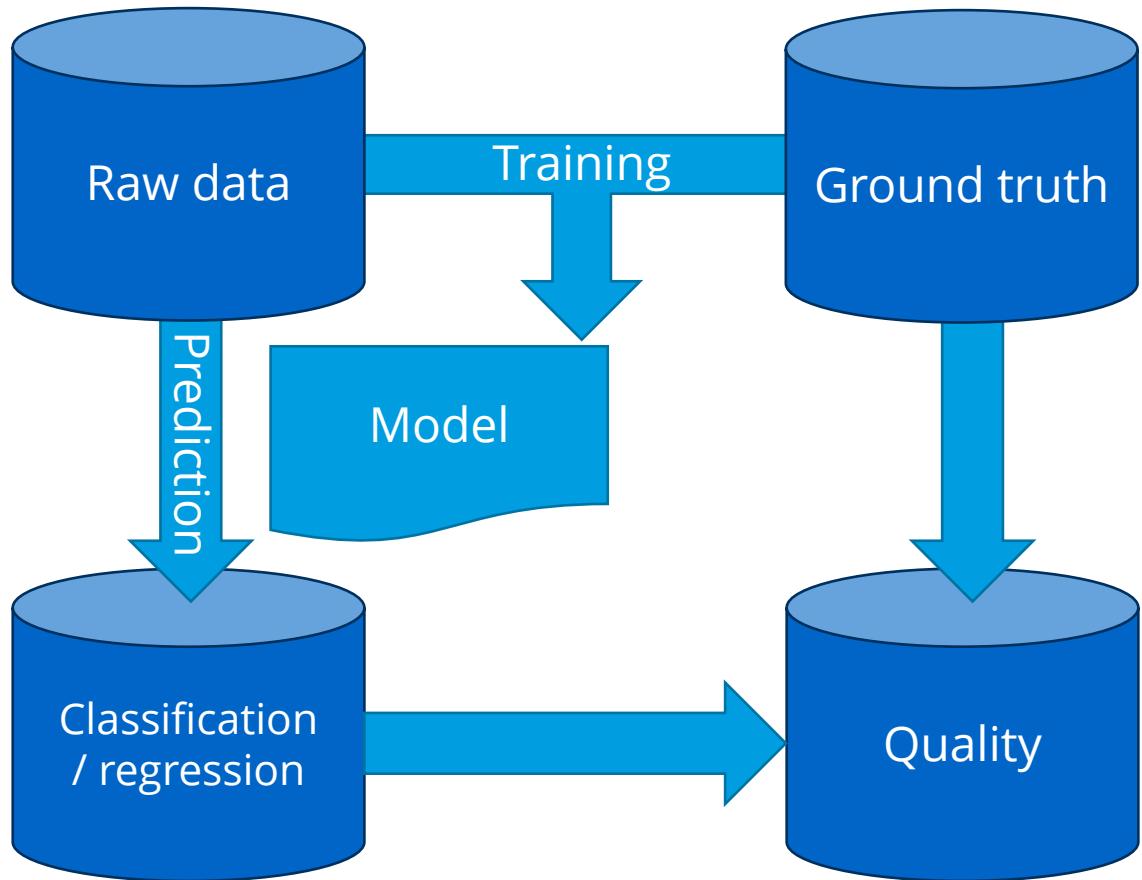
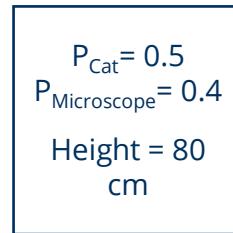
Image
classification



Instance segmentation



Cont. quantity



Annotated
raw data,
usually
generated by
humans

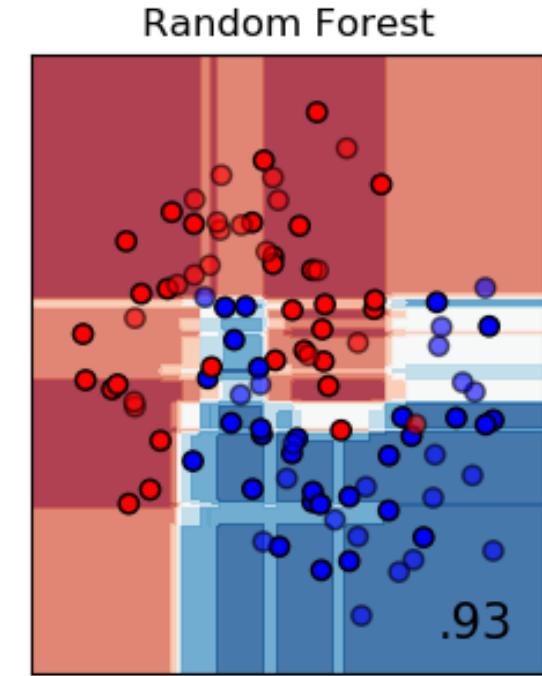
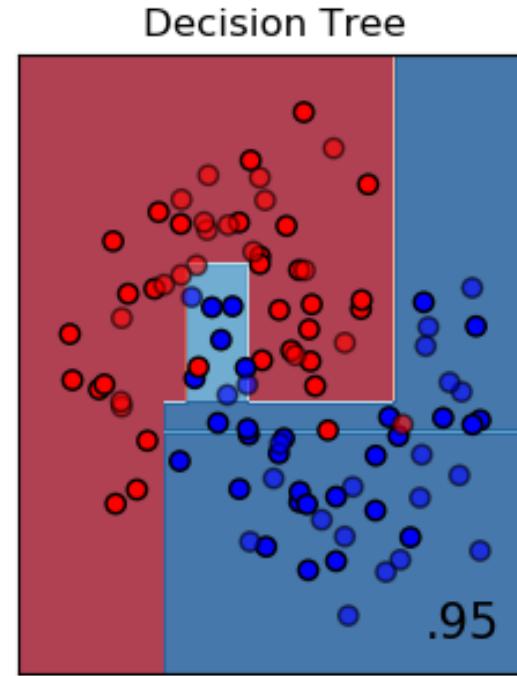
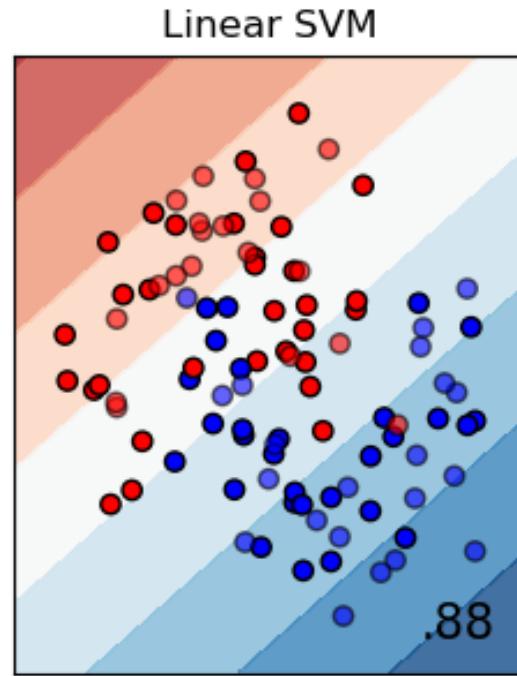
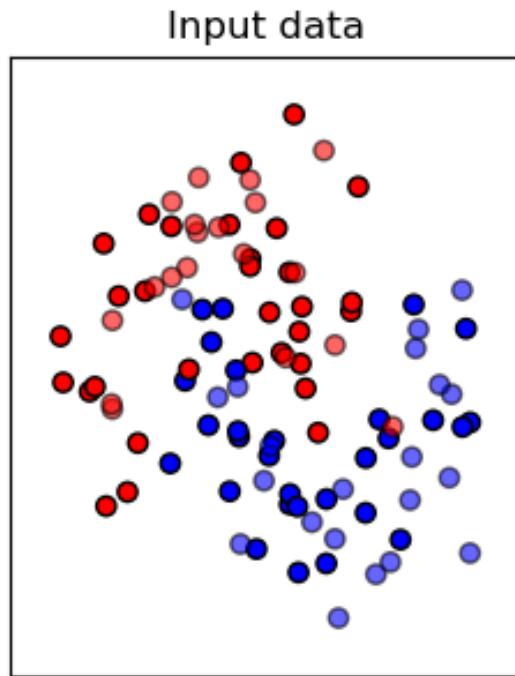
Precision,
Recall

Quiz

What is the difference between classification and regression?

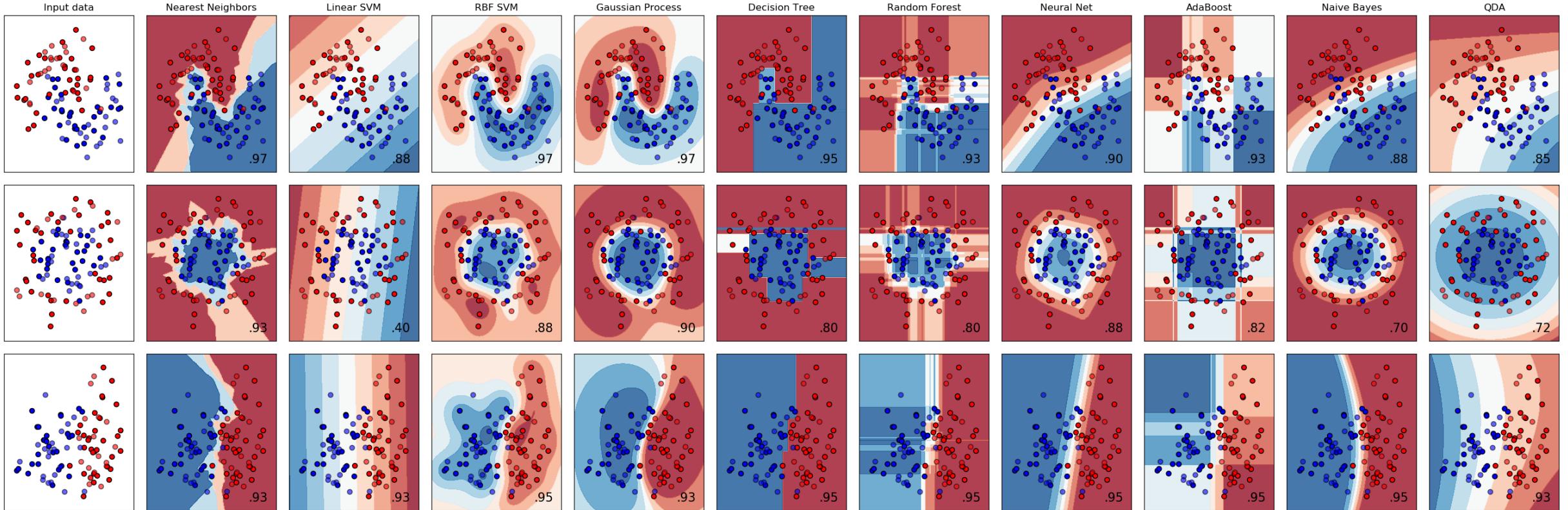
Goal

Guess classification (color) from position of a sample in parameter space.



Approaches

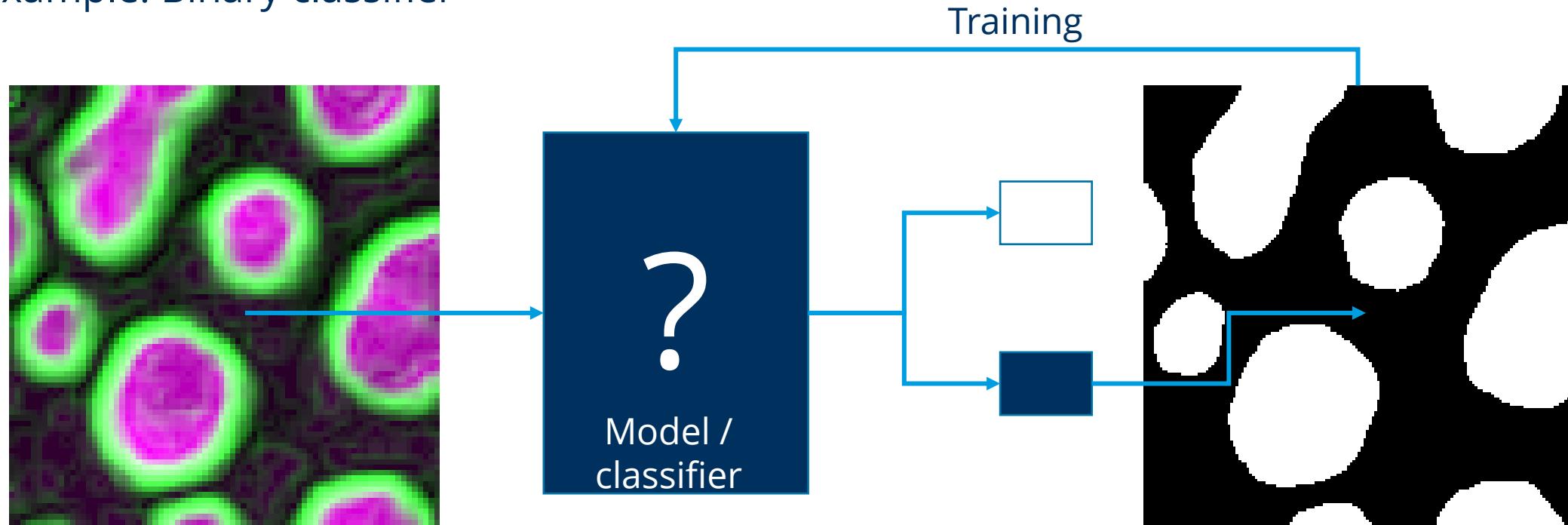
The right approach depends on data, computational resources and desired quality



Machine learning for image segmentation

Supervised machine learning: We give the computer some ground truth to learn from
The computer derives a *model* or a *classifier* which can judge if a pixel should be foreground (white) or background (black)

Example: Binary classifier



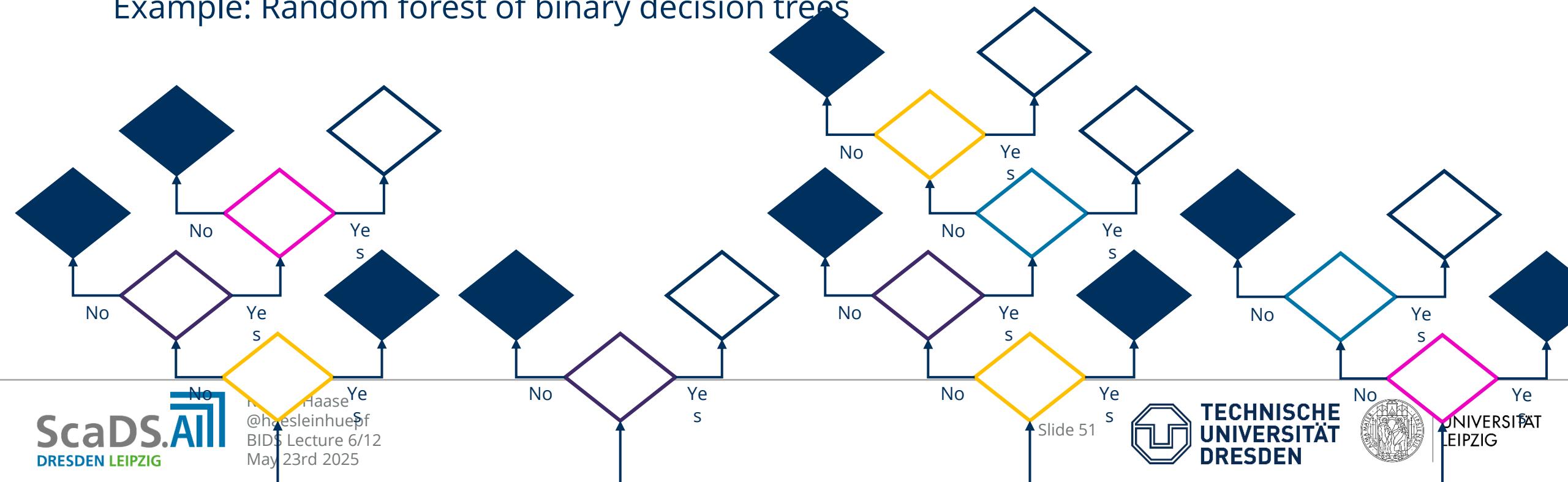
Random forest based image segmentation

Decision trees are classifiers, they decide if a pixel should be white or black

Random decision trees are randomly initialized, afterwards evaluated and selected

Random forests consist of many random decision trees

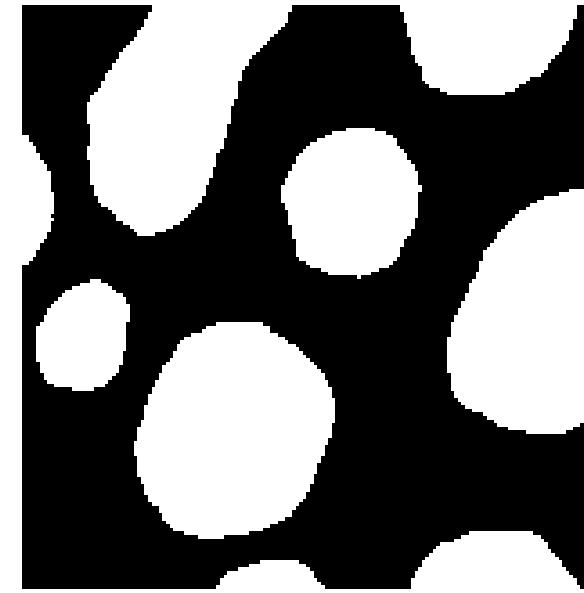
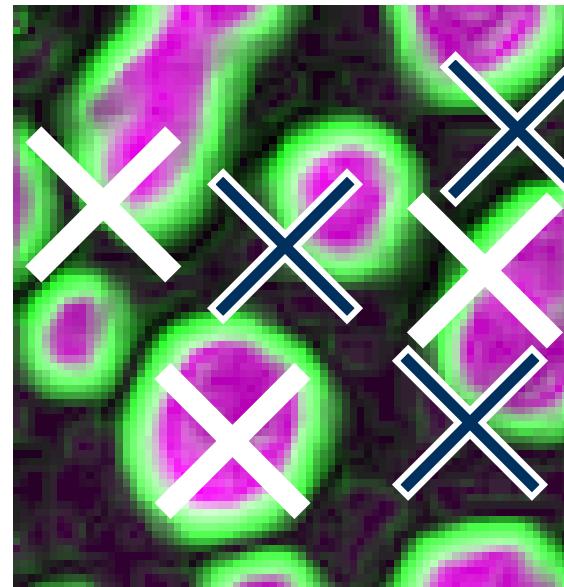
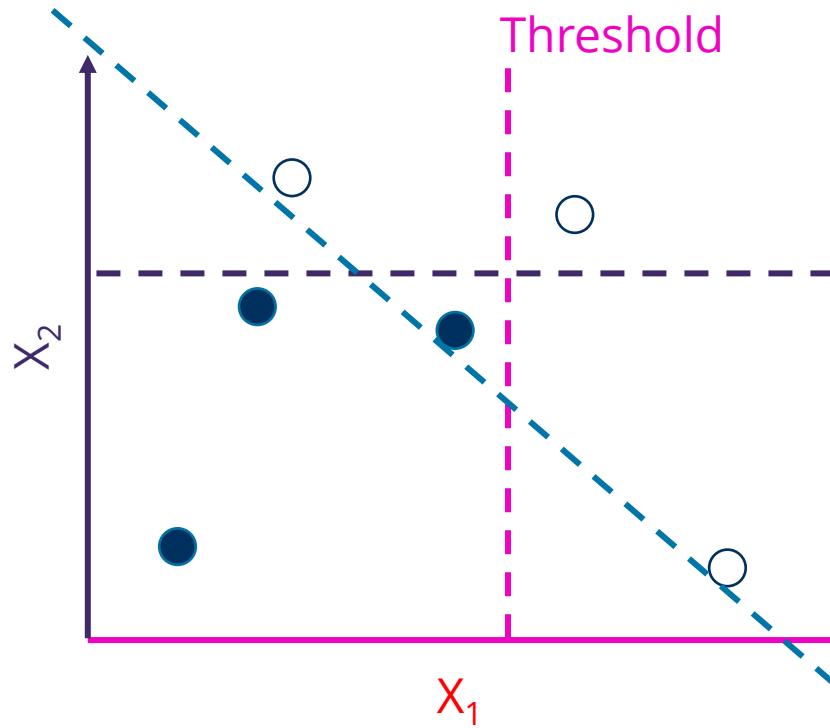
Example: Random forest of binary decision trees



Deriving random decision trees

For efficient processing, we randomly *sample* our data set

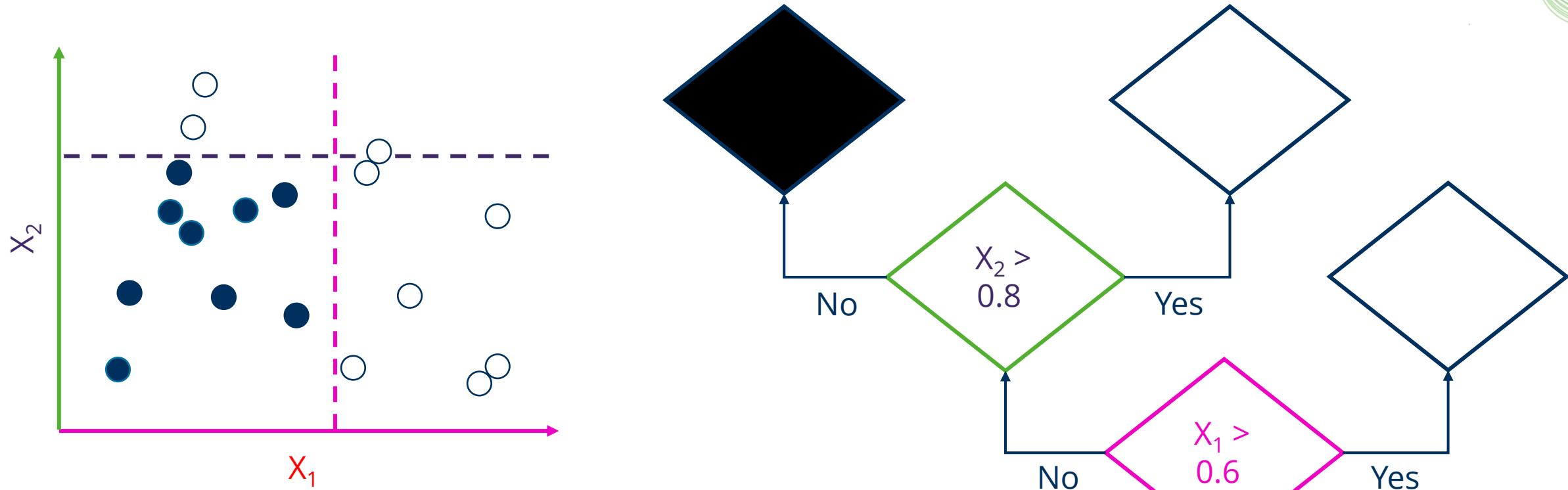
- Individual pixels, their intensity and their classification



Note: You cannot use a single threshold to make the decision

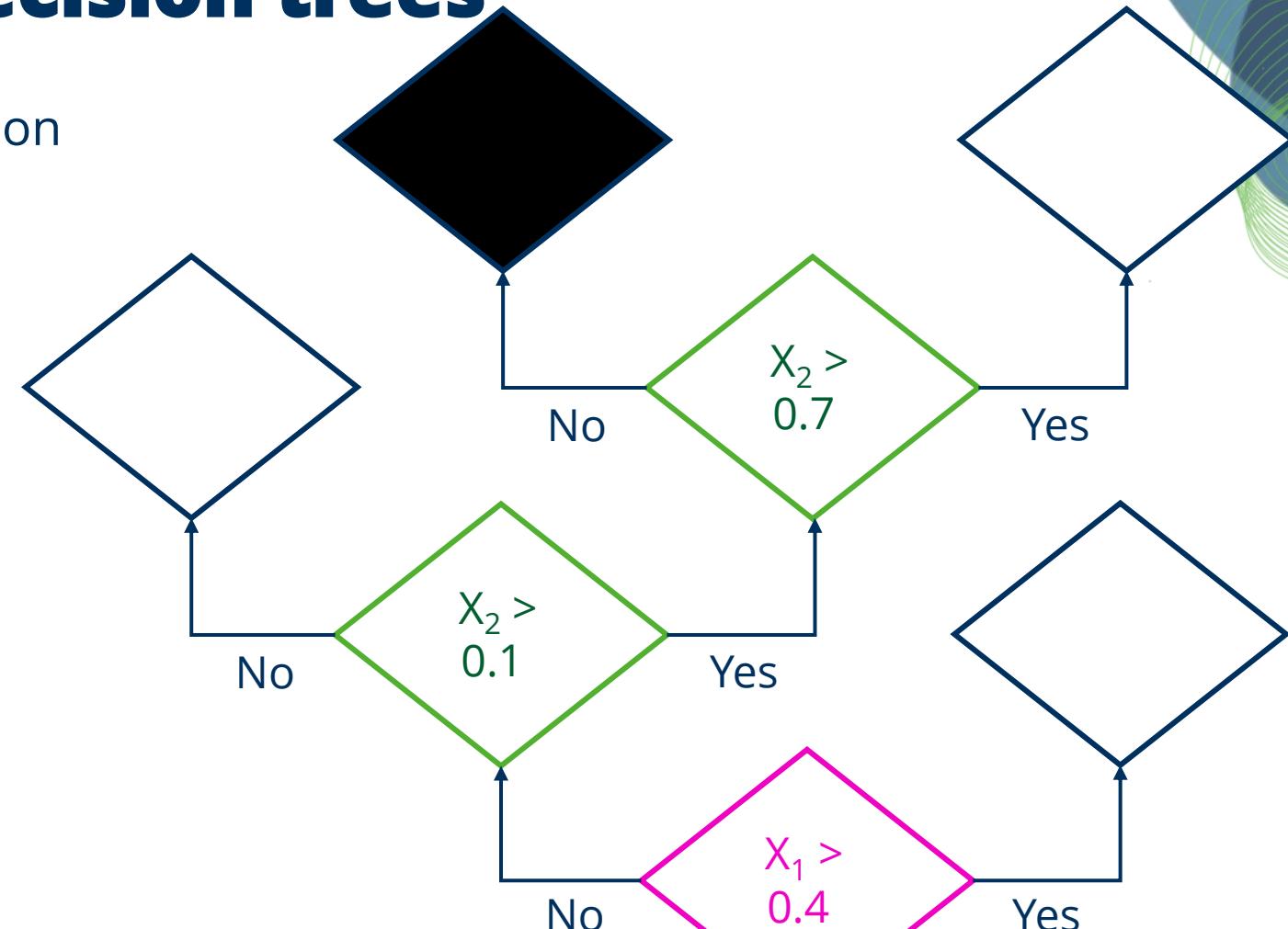
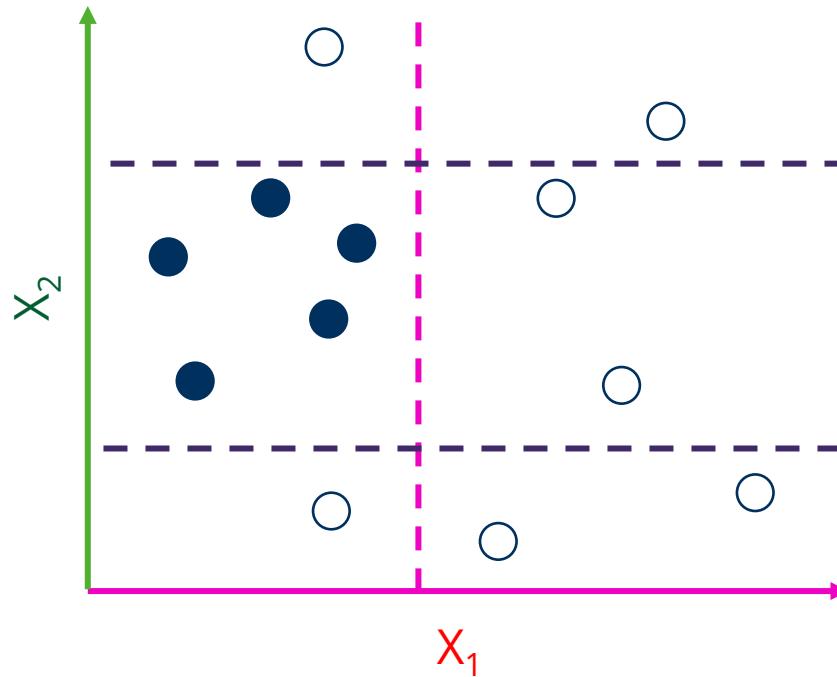
Deriving random decision trees

Decision trees combine several thresholds on several parameters



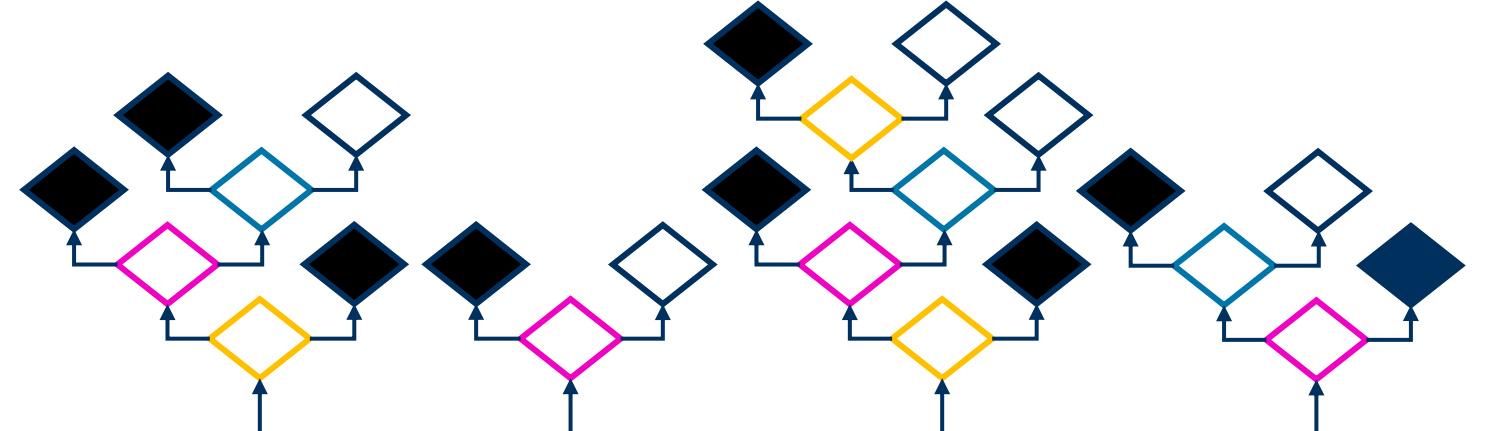
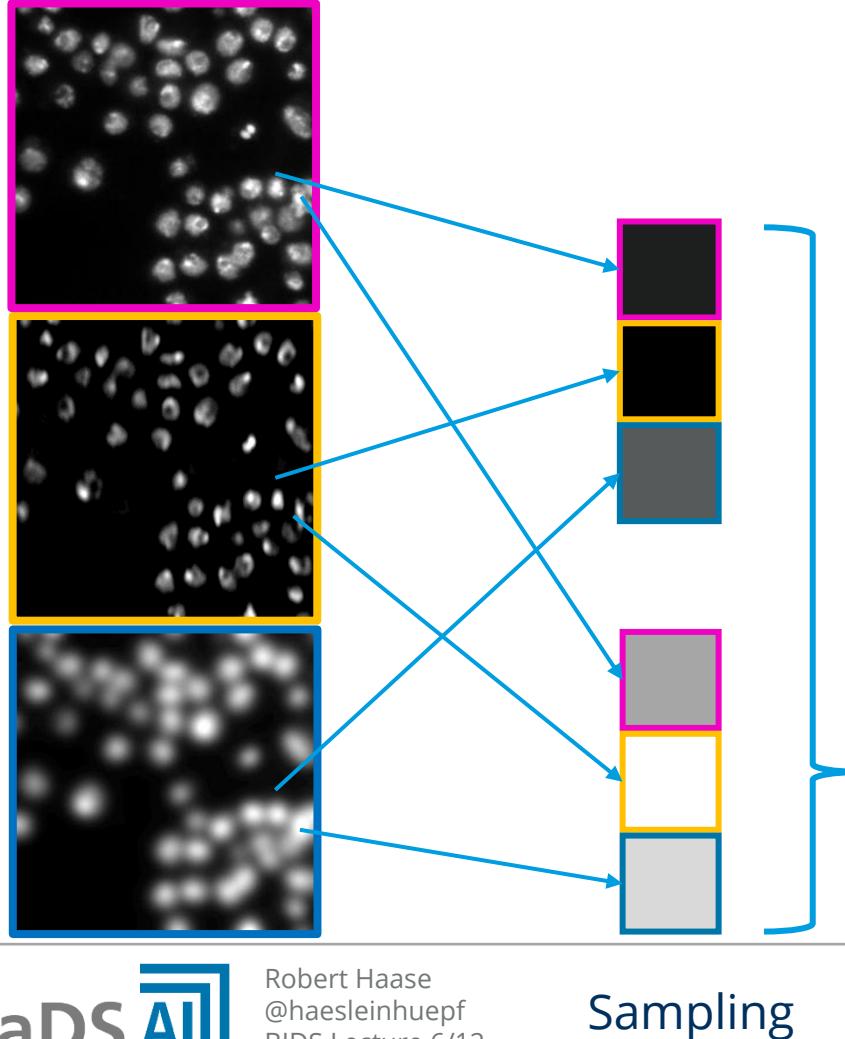
Deriving random decision trees

Depending on sampling, the decision trees are different



Random Forest Pixel Classifiers

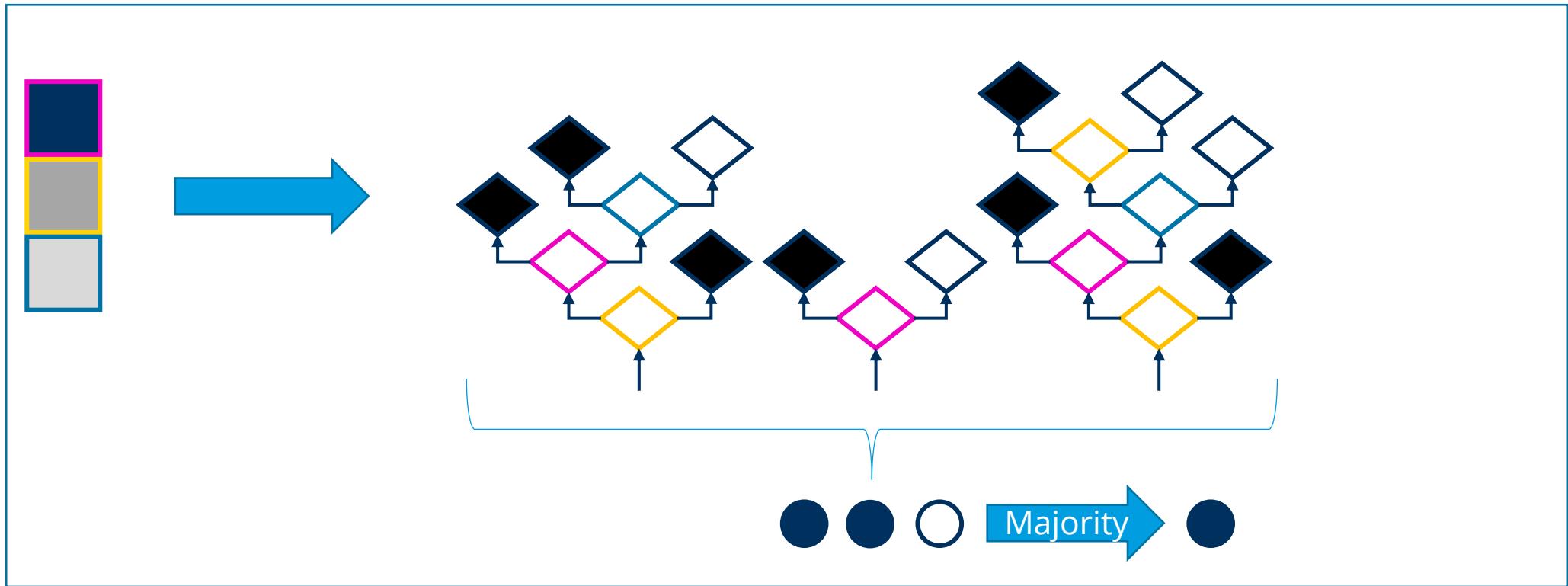
By training many decision trees, errors are equilibrated



Random Forest Pixel Classifiers

Combination of individual tree decisions by voting or max / mean

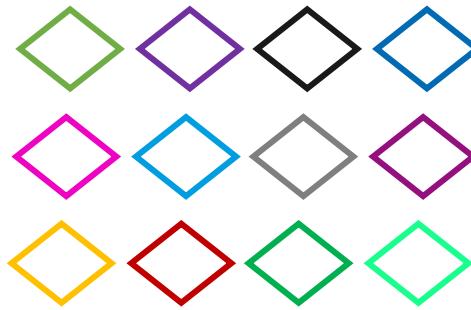
Prediction



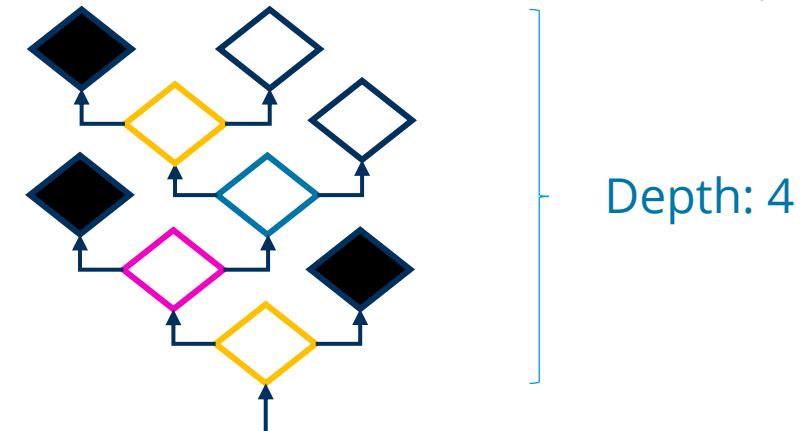
Random Forest Pixel Classifiers

Typical numbers for pixel classifiers in microscopy

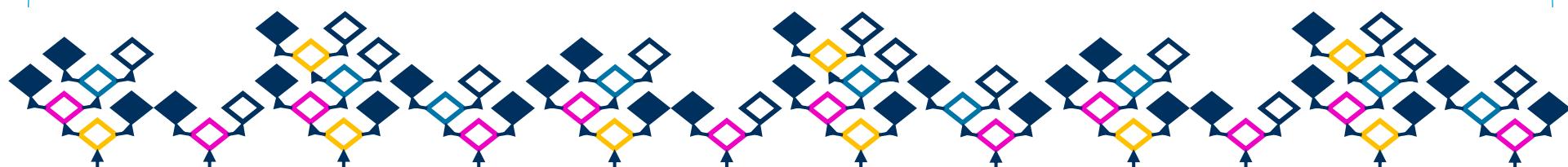
Available features:



- Gaussian blur image
- DoG image
- LoG image
- Hessian
-



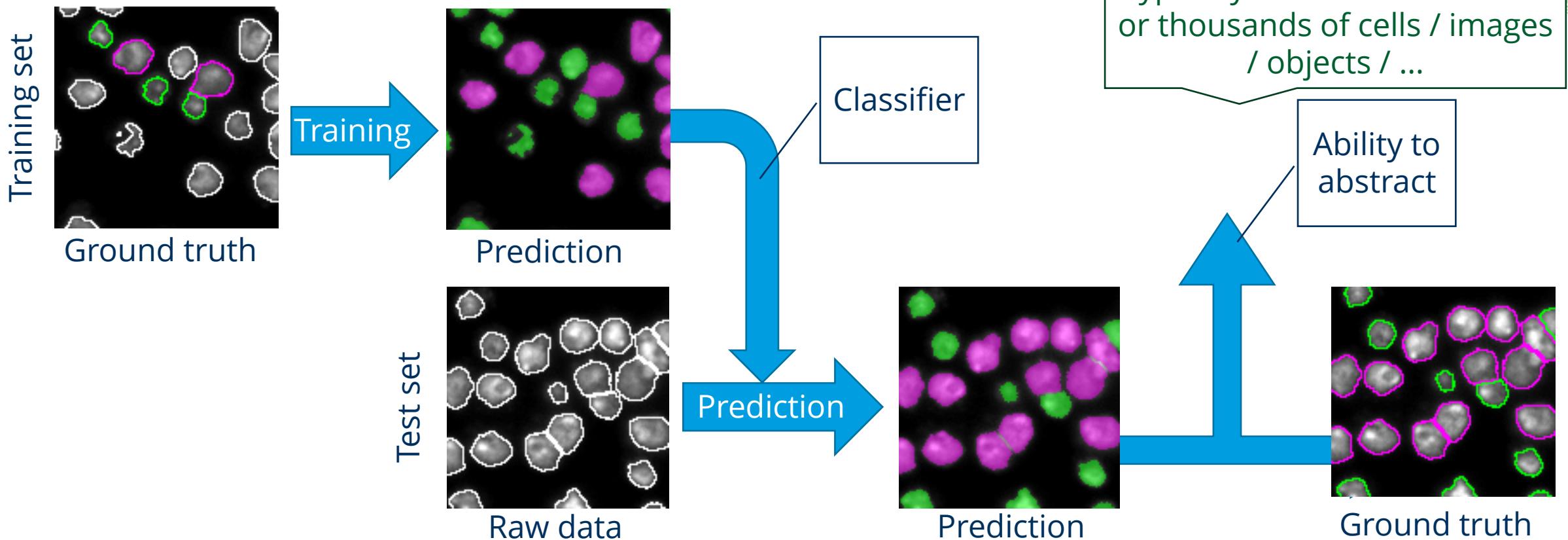
Number of trees: > 100



Model validation

In order to assess model quality, we split the ground truth into two sets

- Training set (50%-90% of the available data)
- Test set (10%-50% of the available data)

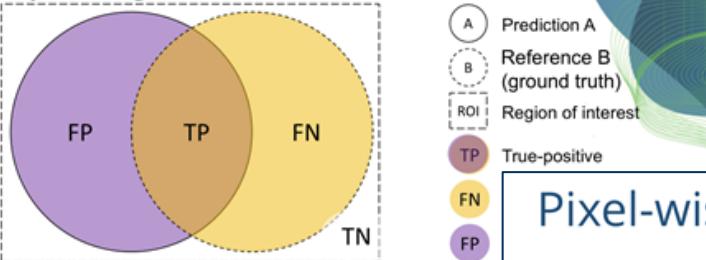


Model validation

Recap

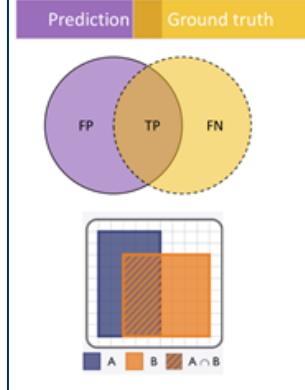
Segmentation quality estimation

- In general
 - Define what's positive and what's negative.
 - Compare with a reference to figure out what was true and false
- Welcome to the [Theory of Sets](#)



Pixel-wise versus Object-wise evaluation

- Pixel wise: Segmentation quality



True-positive: 4

False-negative: 5

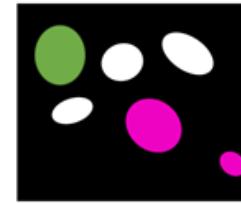
False-positive: 2

$$\text{IoU}(A,B) = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} = \frac{|A \cap B|}{|A \cup B|}$$

IoU = 4 / 11

Pixel-wise versus Object-wise evaluation

- Object wise: Detection quality



True-positive: 3

False-negative: 1

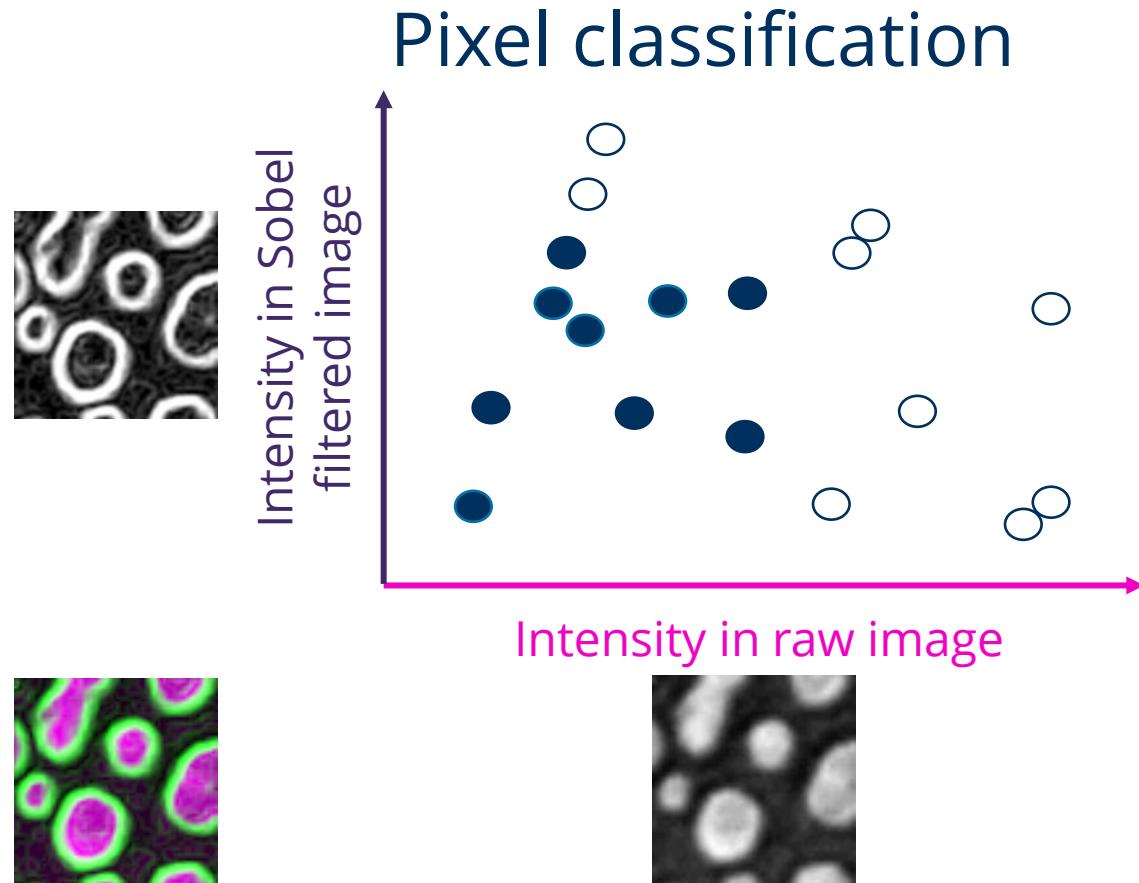
False-positive: 2

$$\text{IoU}(A,B) = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} = \frac{|A \cap B|}{|A \cup B|}$$

IoU = 1 / 2

Object classification

What if we exchange pixel features with object features?



Supervised and Unsupervised Machine Learning for Bio-image Analysis

Robert Haase

Reusing materials from Johannes Soltwedel, Till Korten, Johannes Müller, Laura Žigutytė (TU Dresden), Ryan Savill (MPI-CBG), Matthias Täschner (ScaDS.AI/Uni Leipzig) and the Scikit-learn community.

Using
Python

Funded by



Bundesministerium
für Bildung
und Forschung



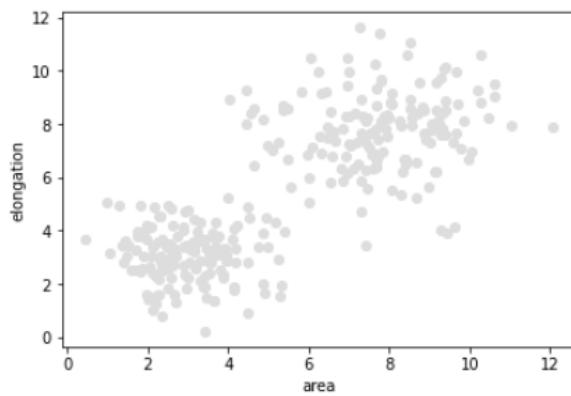
Diese Maßnahme wird gefördert durch die Bundesregierung aufgrund eines Beschlusses des Deutschen Bundestages.
Diese Maßnahme wird mitfinanziert durch Steuermittel auf der Grundlage des von den Abgeordneten des Sächsischen Landtags beschlossenen Haushaltes.

Tabular object classification

Classify objects starting from feature vectors (table columns)

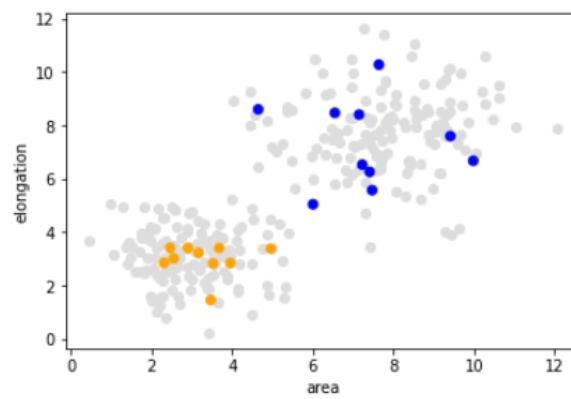
Raw data

	area	elongation
0	3.950088	2.848643
1	4.955912	3.390093
2	7.469852	5.575289
3	2.544467	3.017479
4	3.465662	1.463756
5	3.156507	3.232181
6	9.978705	6.676372
7	6.001683	5.047063
8	2.457139	3.416050
9	3.672295	3.407462
10	9.413702	7.598608



“Ground truth” annotation

annotation = [1, 1, 2, 1, 1, 1,

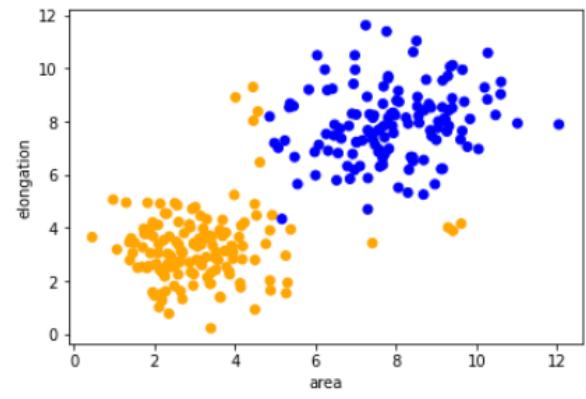


Classifier training

```
classifier = RandomForestClassifier()  
classifier.fit(train_data, train_annotation)
```

Classifier prediction

```
result = classifier.predict(validation_data)
```



Interactive pixel classification

Prepare an empty layer for annotations and keep a reference

```
labels = viewer.add_labels(  
    np.zeros(image.shape).astype(int))
```

Read annotations

```
manual_annotations = labels.data
```

```
from skimage.io import imshow  
  
imshow(manual_annotations,  
       vmin=0, vmax=2)
```

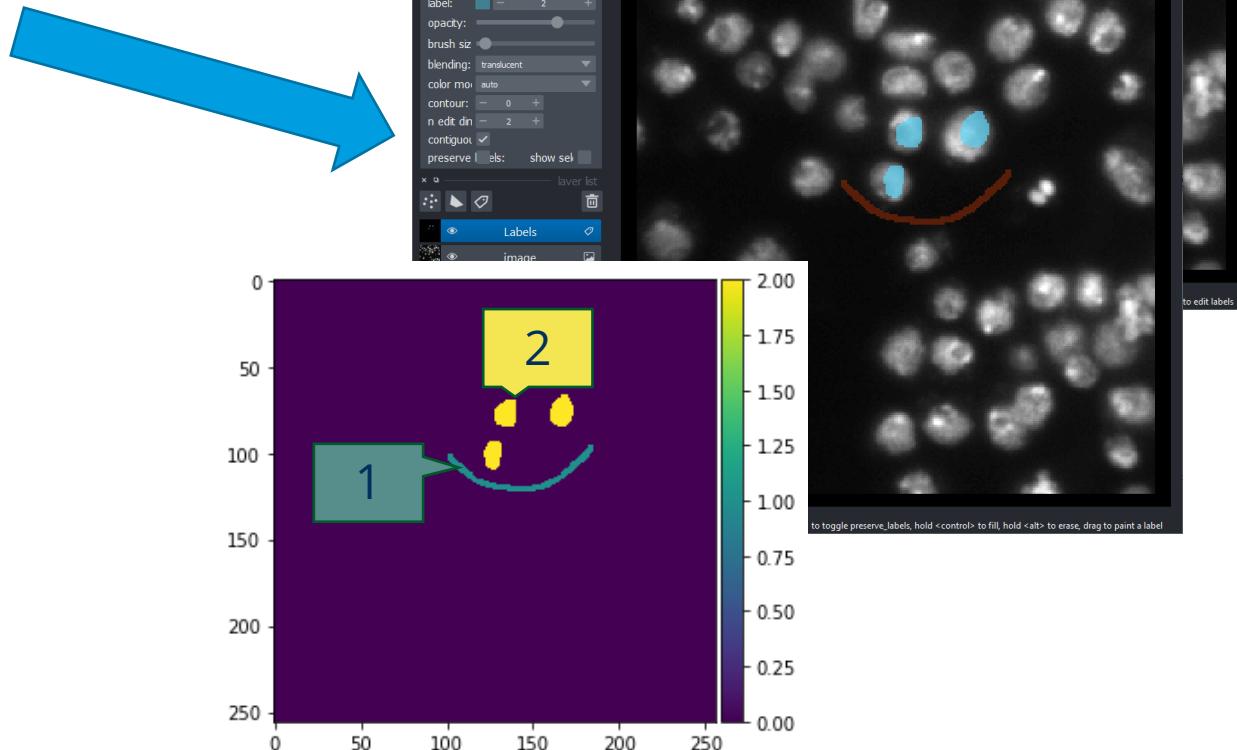
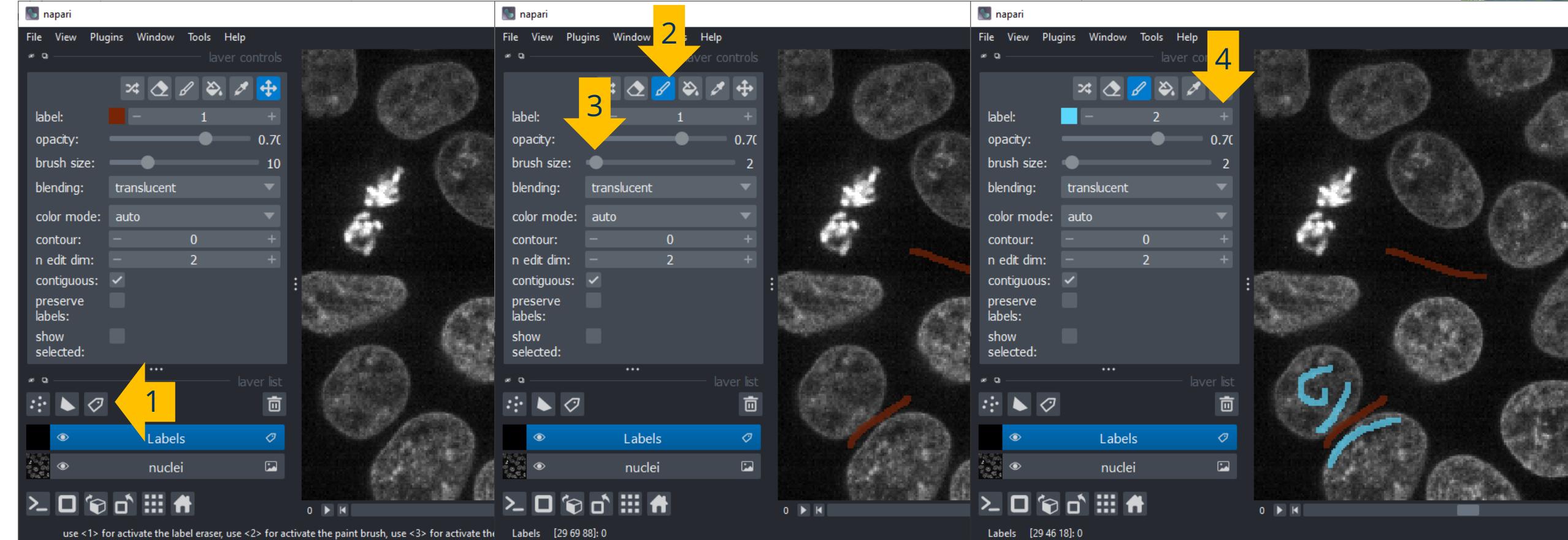


Image data source: BBB038v1, available from the Broad BioImage Benchmark Collection (Caicedo et al., Nature Methods, 2019).

Napari – common workflows

Pixel / object annotation drawing

- [1: Create empty labels layer]
- 2: Select paint brush tool
- 3: Decrease brush size
- 4: Increase label

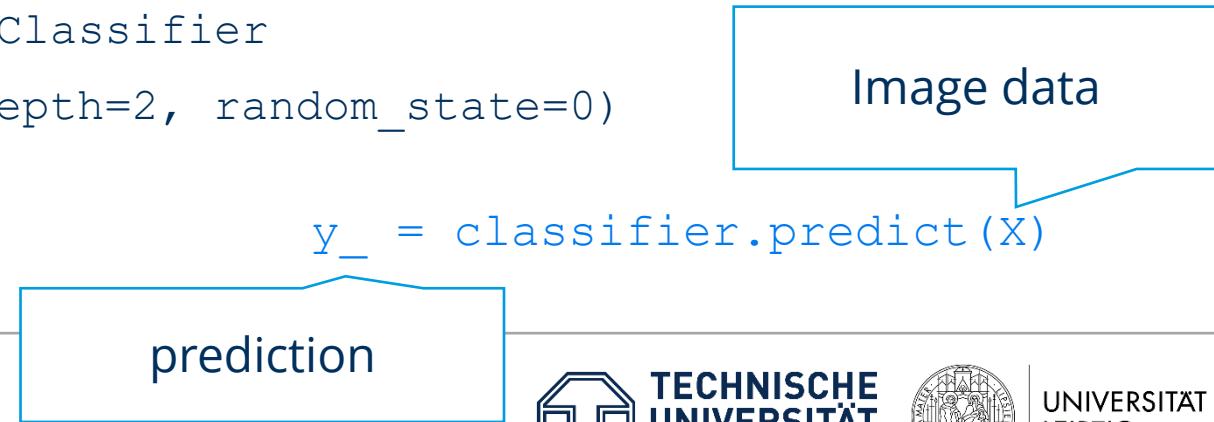


Interactive pixel classification

Pixel classification using scikit-learn

- Expects one-dimensional arrays for features and ground truth

```
# train classifier  
  
from sklearn.ensemble import RandomForestClassifier  
  
classifier = RandomForestClassifier(max_depth=2, random_state=0)  
  
classifier.fit(X, y)
```



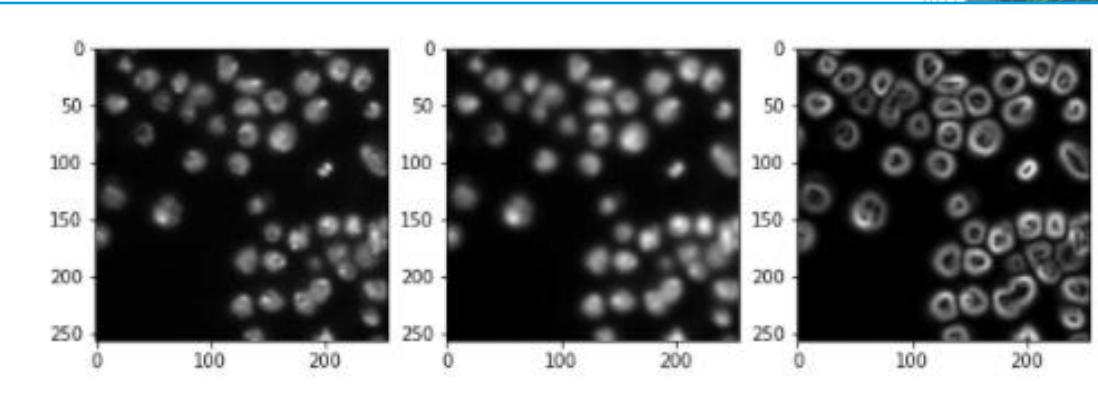
Interactive pixel classification

Pixel classification using scikit-learn

- Expects one-dimensional arrays for features and ground truth

```
# for training, we need to generate features
feature_stack = generate_feature_stack(image)
X, y = format_data(feature_stack, manual_annotations)

# train classifier
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(max_depth=2, random_state=0)
classifier.fit(X, y)
```



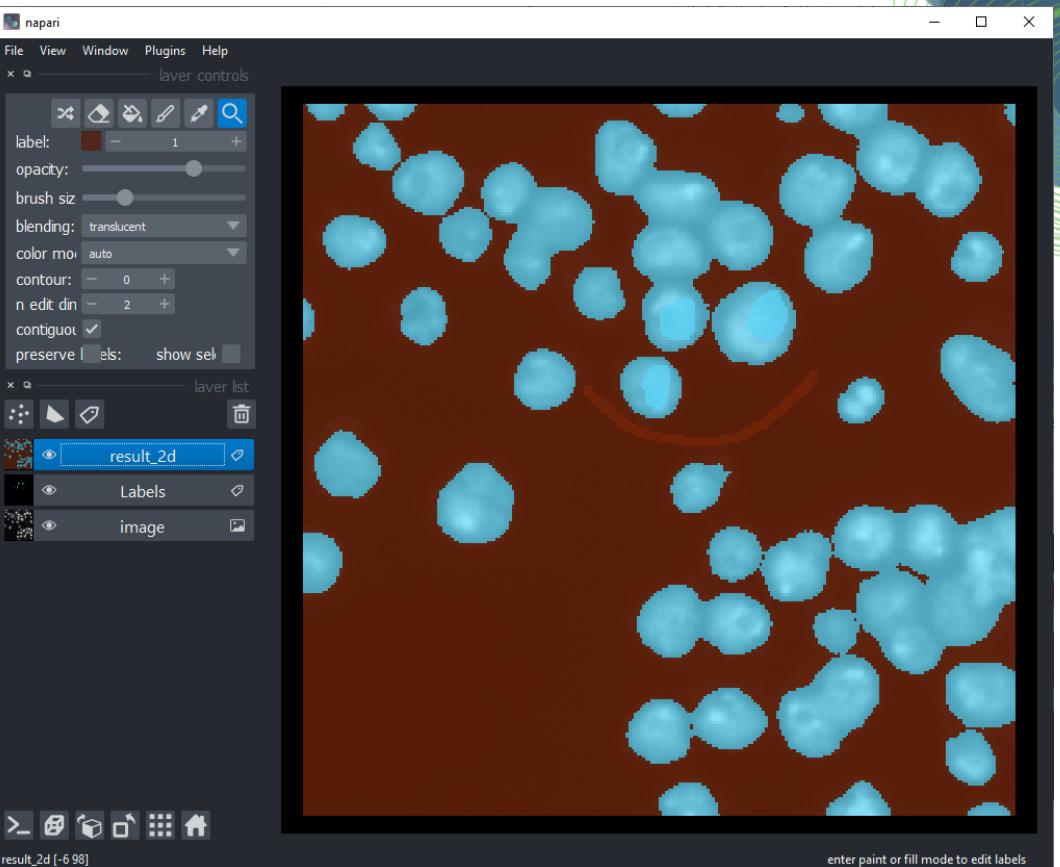
Interactive pixel classification

Pixel classification using scikit-learn

- Expects one-dimensional arrays for features and ground truth

```
# process the whole image and show result  
  
result_1d = classifier.predict(feature_stack.T)  
  
result_2d = result_1d.reshape(image.shape)  
  
viewer.add_labels(result_2d)
```

Convert 1D
result back to 2D



Supervised and unsupervised ML in scikit-learn

Common patterns

```
from sklearn.xyz import Model
```

```
model = Model(a=4, b=5)
```

```
model.fit(x, y) # supervised  
model.fit(x) # unsupervised
```

```
y_pred = model.predict(x)
```

$$y = f(x)$$

Interactive pixel classification

Jupyter notebooks and napari side-by-side

scikit_learn_random_forest_pixel_ x +

localhost:8889/notebooks/machine_learning/scikit_learn_random_forest_pixel_clas... Log

jupyter scikit_learn_random_forest_pixel_classifier (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python

In [30]:

```
import napari

# start napari
viewer = napari.Viewer()

# add image
viewer.add_image(image)

# add an empty Labels layer and keep it in a variable
labels = viewer.add_labels(np.zeros(image.shape).astype(int))
```

Interactive segmentation

We can also use napari to annotate some regions as negative (label = 1) and positive (label = 2).

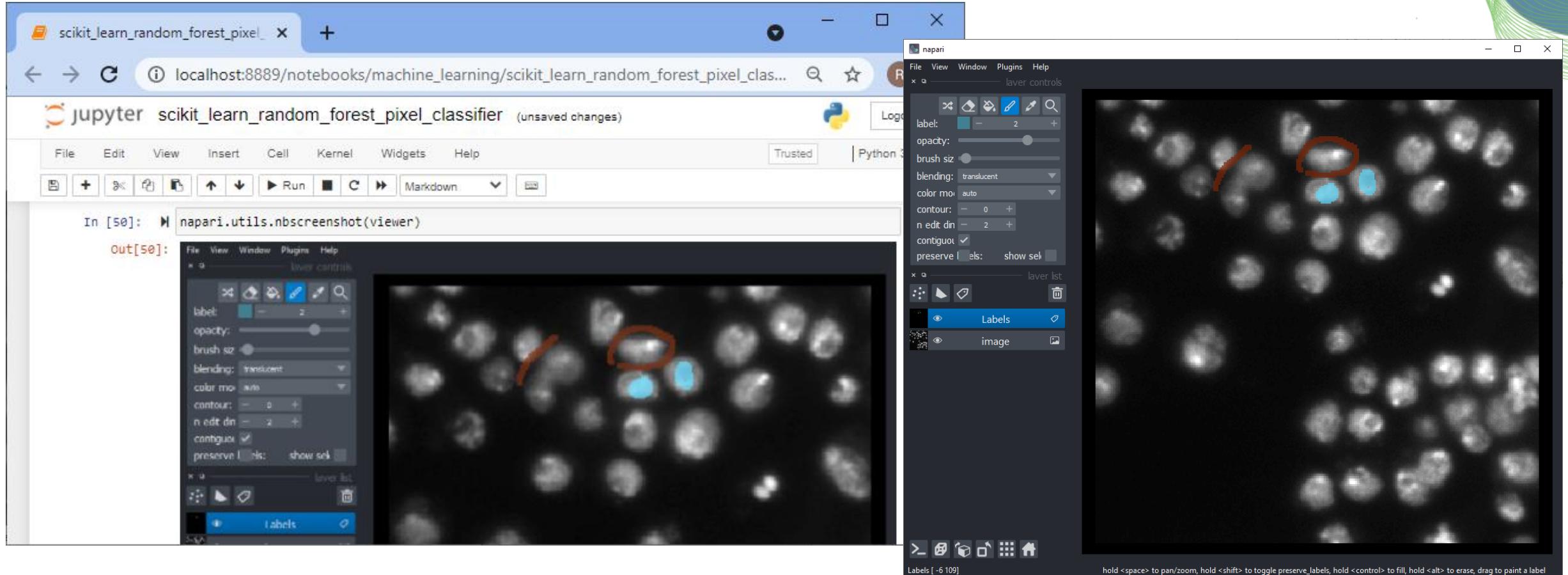
Go ahead after annotating at least two regions with labels 1 and 2.

Take a screenshot of the annotation:

Labels [-6 -7] enter paint or fill mode to edit labels

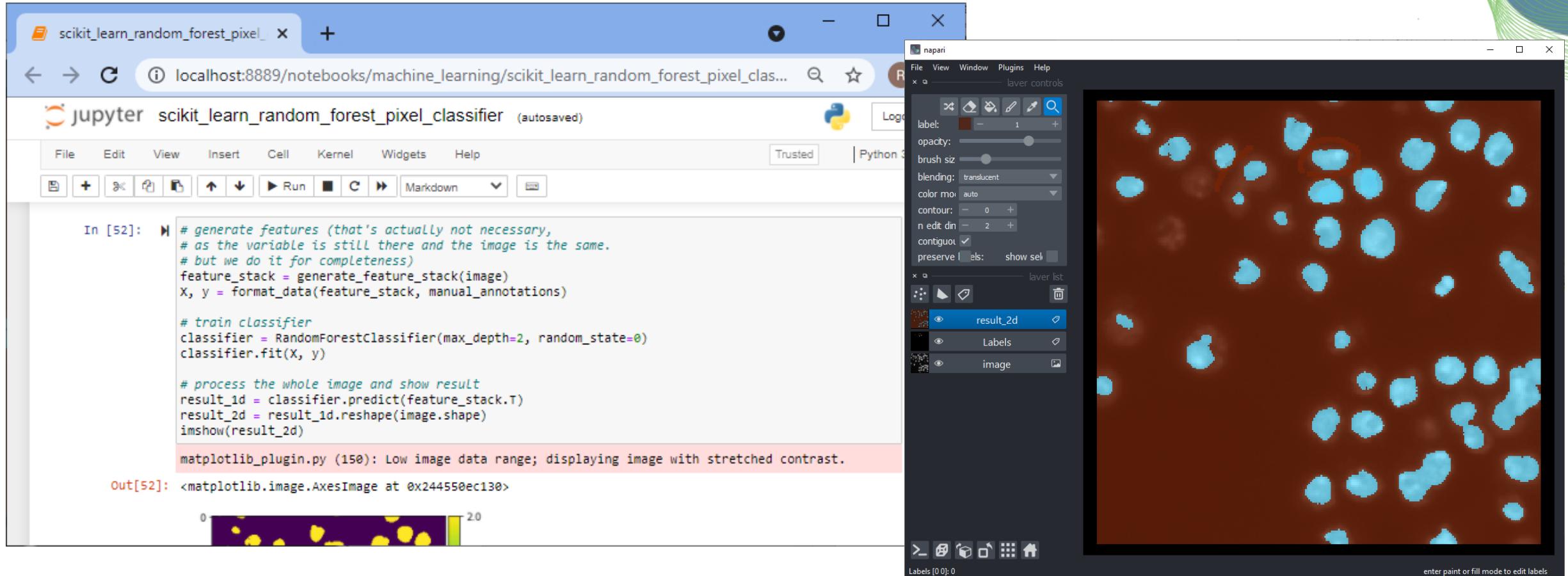
Interactive pixel classification

Jupyter notebooks and napari side-by-side



Interactive pixel classification

Jupyter notebooks and napari side-by-side



Supervised and Unsupervised Machine Learning for Bio-image Analysis

Robert Haase

Reusing materials from Johannes Soltwedel, Till Korten, Johannes Müller, Laura Žigutytė (TU Dresden), Ryan Savill (MPI-CBG), Matthias Täschner (ScaDS.AI/Uni Leipzig) and the Scikit-learn community.

GPU-accelerated

Using
Python

Funded by



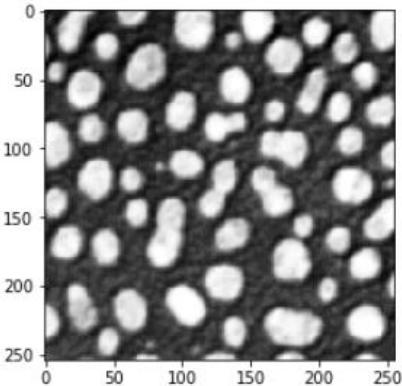
Bundesministerium
für Bildung
und Forschung



Diese Maßnahme wird gefördert durch die Bundesregierung aufgrund eines Beschlusses des Deutschen Bundestages.
Diese Maßnahme wird mitfinanziert durch Steuermittel auf der Grundlage des von den Abgeordneten des Sächsischen Landtags beschlossenen Haushaltes.

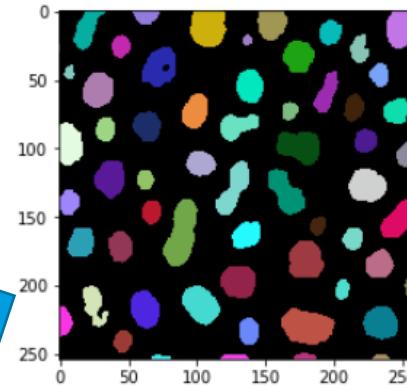
Accelerated pixel and object classification

APOC is a python library that makes use of OpenCL-compatible Graphics Cards to accelerate pixel and object classification



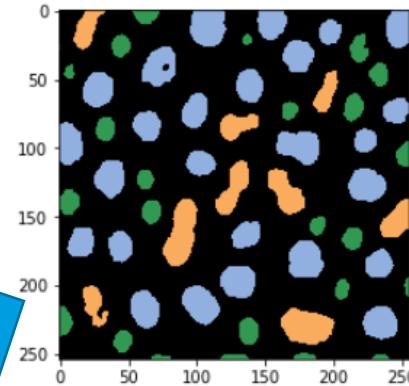
Raw image

Object segmentation

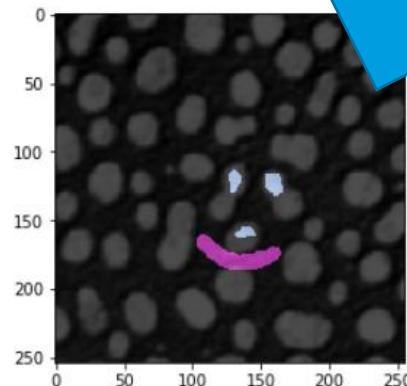


Object label image

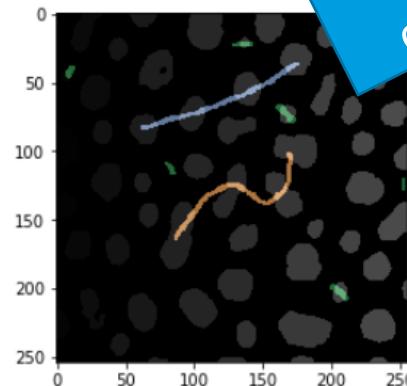
Object classification



Class label image



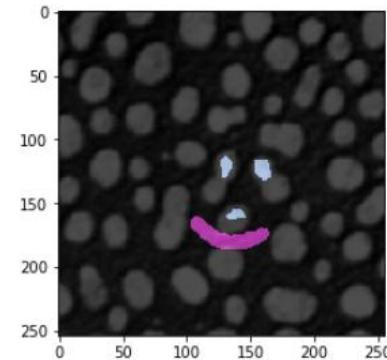
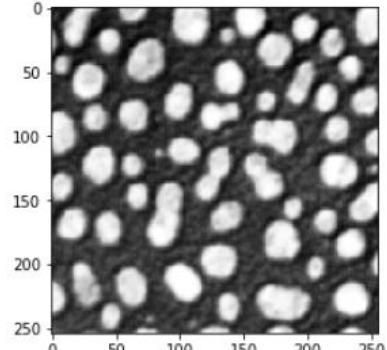
Pixel annotation



Object annotation

Object segmentation

Pixel classification + connected component labeling



Pixel annotation

```
# define features
features = "gaussian_blur=1 gaussian_blur=5 sobel_of_gaussian_blur=1"

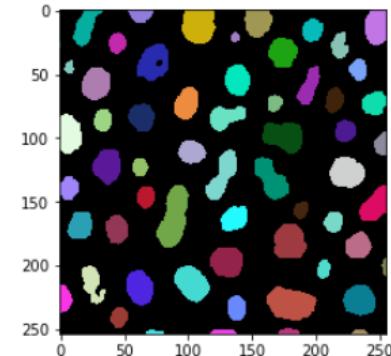
# this is where the model will be saved
cl_filename = 'my_object_segmenter.cl'

# delete classifier in case the file exists already
apoc.erase_classifier(cl_filename)

# train classifier
clf = apoc.ObjectSegmenter(opencl_filename=cl_filename, positive_class_identifier=2)
clf.train(features, manual_annotations, image)

segmentation_result = clf.predict(features=features, image=image)
cle.imshow(segmentation_result, labels=True)
```

Object segmentation



Object label image

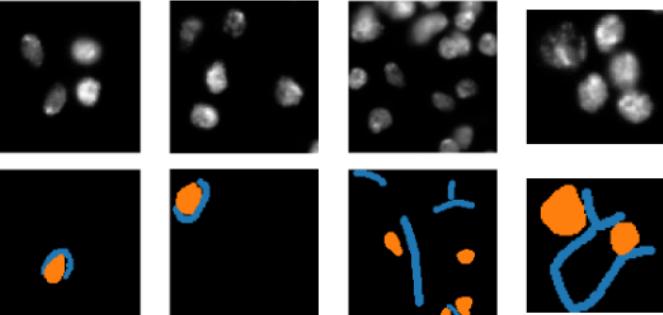
Training on folders of annotated images

```
[2]: image_folder = "data/BBBC007/images/"
masks_folder = "data/BBBC007/masks/"

[3]: file_list = os.listdir(image_folder)

# show all images
fig, axs = plt.subplots(1, 4, figsize=(15,15))
for i, filename in enumerate(file_list):
    image = imread(image_folder + filename)
    stackview.imshow(image, plot=axs[i])
plt.show()

# show corresponding label images
fig, axs = plt.subplots(1, 4, figsize=(15,15))
for i, filename in enumerate(file_list):
    masks = imread(masks_folder + filename)
    stackview.imshow(masks, plot=axs[i], labels=True)
plt.show()
```



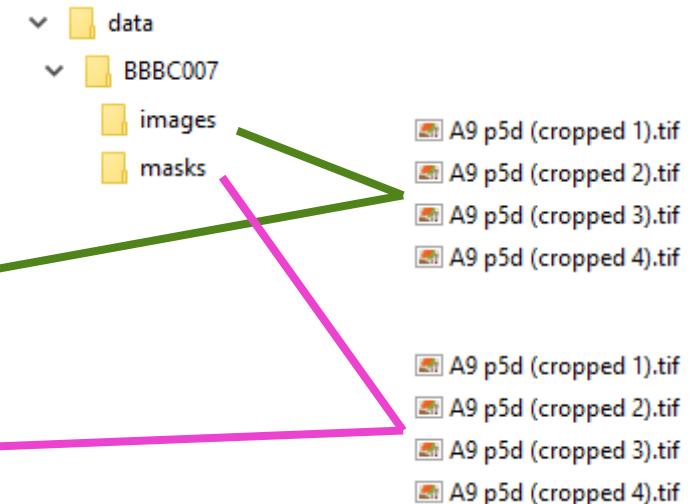
Training

If the folders are setup properly, we can pass the folders to the training.

```
[4]: # setup classifier and where it should be saved
segmenter = apoc.ObjectSegmenter(opencl_filename="data/object_segmenter_trained_on_f

# setup feature set used for training
features = apoc.PredefinedFeatureSet.object_size_1_to_5_px.value

# train classifier on folders
apoc.train_classifier_from_image_folders(
    segmenter,
    features,
    image = image_folder,
    ground_truth = masks_folder)
```



Prediction

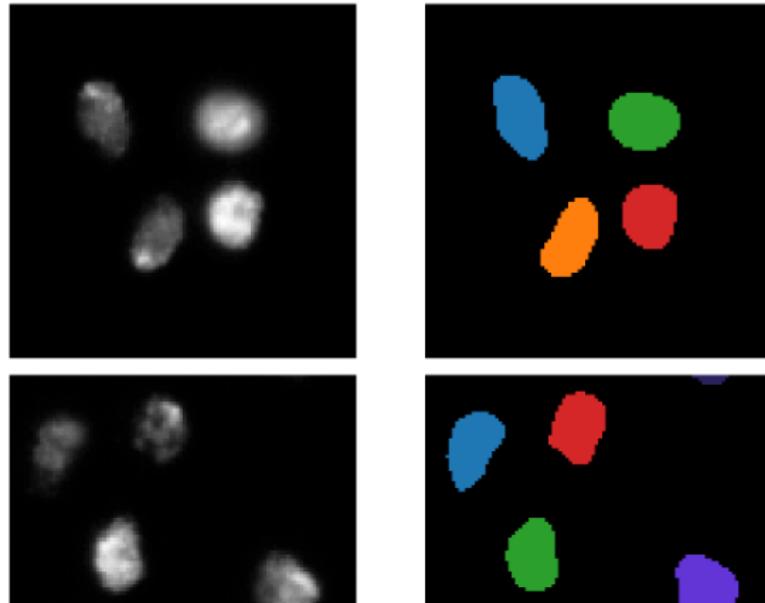
After the training, we can apply the classifier to all images in the image folder. The following line reloads the classifier from disk. In that way we can ensure that it was stored correctly.

```
[5]: segmenter = apoc.ObjectSegmenter(opencl_filename="data/object_segmenter_trained_on_f

[6]: # show all images
for i, filename in enumerate(file_list):
    fig, axs = plt.subplots(1, 2, figsize=(15,15))

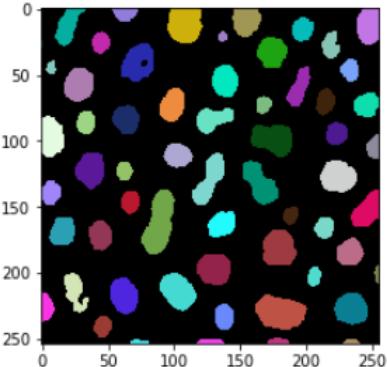
    image = imread(image_folder + filename)
    stackview.imshow(image, plot=axs[0])

    labels = segmenter.predict(image)
    stackview.imshow(labels, plot=axs[1], labels=True)
```

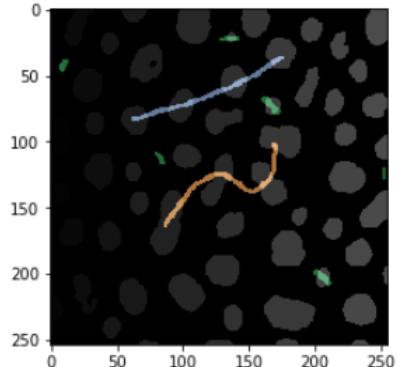


Object classification

Feature extraction + tabular classification



Object label image



Object annotation

```
# for the classification we define size and shape as criteria
features = 'area mean_max_distance_to_centroid_ratio'

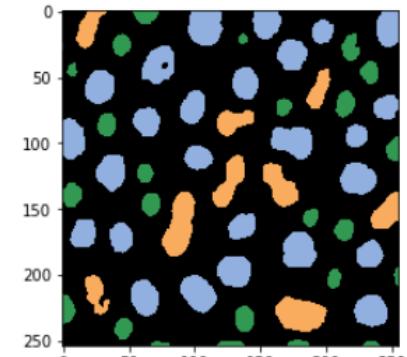
# This is where the model will be saved
cl_filename_object_classifier = "my_object_classifier.cl"

# delete classifier in case the file exists already
apoc.erase_classifier(cl_filename_object_classifier)

# train the classifier
classifier = apoc.ObjectClassifier(cl_filename_object_classifier)
classifier.train(features, segmentation_result, annotation, image)

# determine object classification
classification_result = classifier.predict(segmentation_result, image)
cle.imshow(classification_result, labels=True)
```

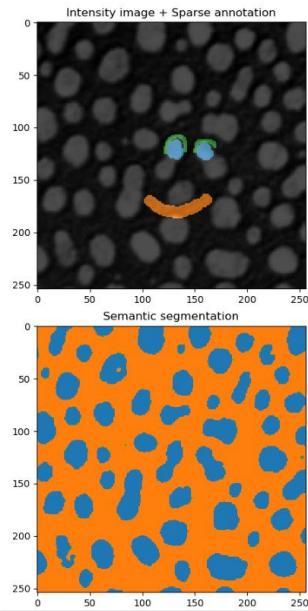
Object classification



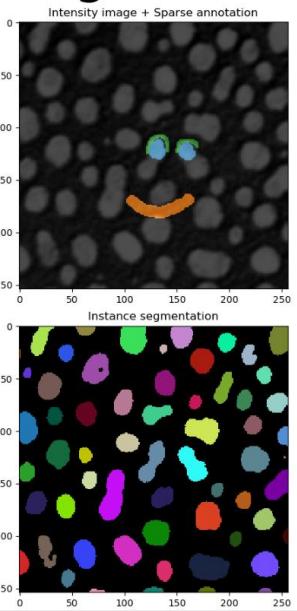
Class label image

Other classification / regression tasks

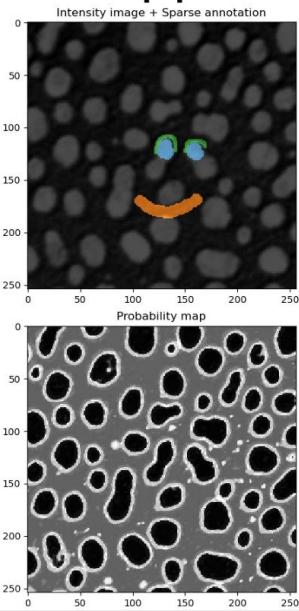
Pixel-
Classifier



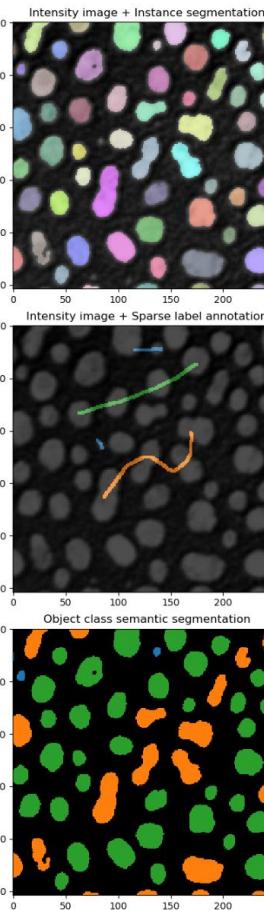
Object-
Segmenter



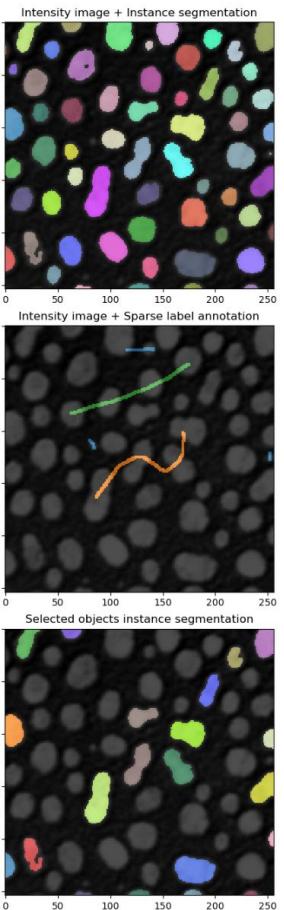
Probability-
Mapper



Object-
Mapper



Object-
Classifier

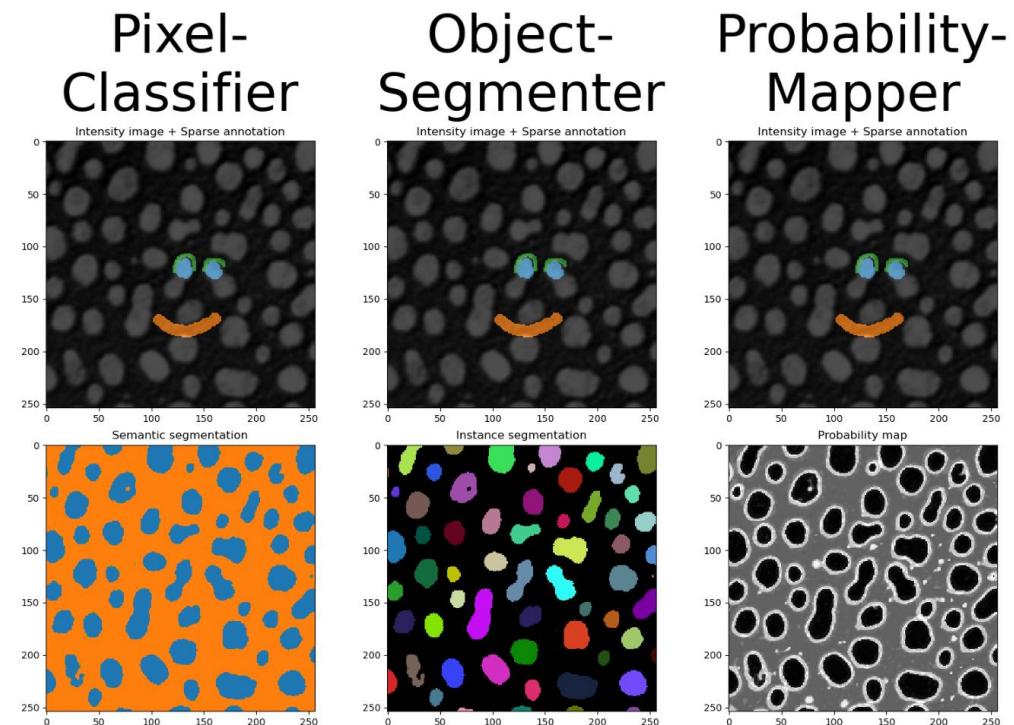


Slide 77



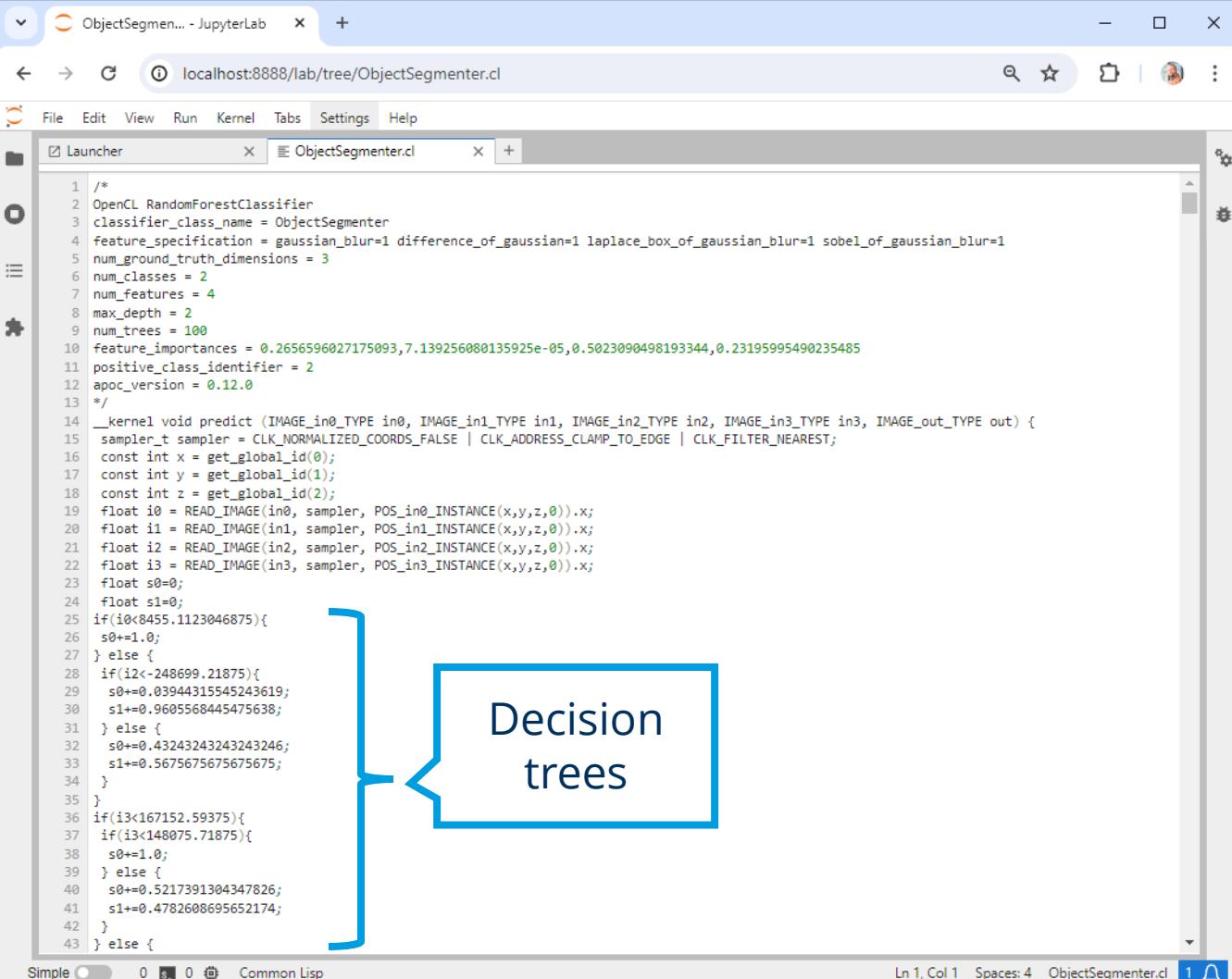
Quiz: Classification versus Regression

Which of these three solves a regression task?



Under the hood: clesperanto / OpenCL

classifier.cl files
can be *read*



```
/*
OpenCL RandomForestClassifier
classifier_class_name = ObjectSegmenter
feature_specification = gaussian_blur=1 difference_of_gaussian=1 laplace_box_of_gaussian_blur=1 sobel_of_gaussian_blur=1
num_ground_truth_dimensions = 3
num_classes = 2
num_features = 4
max_depth = 2
num_trees = 100
feature_importances = 0.2656596027175093, 7.139256080135925e-05, 0.5023090498193344, 0.23195995490235485
positive_class_identifier = 2
apoc_version = 0.12.0
*/
__kernel void predict (IMAGE_in0_TYPE in0, IMAGE_in1_TYPE in1, IMAGE_in2_TYPE in2, IMAGE_in3_TYPE in3, IMAGE_out_TYPE out) {
    sampler_t sampler = CLK_NORMALIZED_COORDS_FALSE | CLK_ADDRESS_CLAMP_TO_EDGE | CLK_FILTER_NEAREST;
    const int x = get_global_id(0);
    const int y = get_global_id(1);
    const int z = get_global_id(2);
    float i0 = READ_IMAGE(in0, sampler, POS_in0_INSTANCE(x,y,z,0)).x;
    float i1 = READ_IMAGE(in1, sampler, POS_in1_INSTANCE(x,y,z,0)).x;
    float i2 = READ_IMAGE(in2, sampler, POS_in2_INSTANCE(x,y,z,0)).x;
    float i3 = READ_IMAGE(in3, sampler, POS_in3_INSTANCE(x,y,z,0)).x;
    float s0=0;
    float s1=0;
    if(i0>8455.1123046875){
        s0+=1.0;
    } else {
        if(i2<-248699.21875){
            s0+=0.03944315545243619;
            s1+=0.9605568445475638;
        } else {
            s0+=0.43243243243243246;
            s1+=0.5675675675675675;
        }
    }
    if(i3<167152.59375){
        if(i3<148075.71875){
            s0+=1.0;
        } else {
            s0+=0.5217391304347826;
            s1+=0.4782608695652174;
        }
    } else {

```

Decision trees

Explainable Machine Learning

Robert Haase

These slides can be reused under the terms of the [CC-BY4.0](#) license.

Explainable Artificial Intelligence (XAI)

- “Es gibt derzeit noch keine allgemein akzeptierte Definition von XAI.”

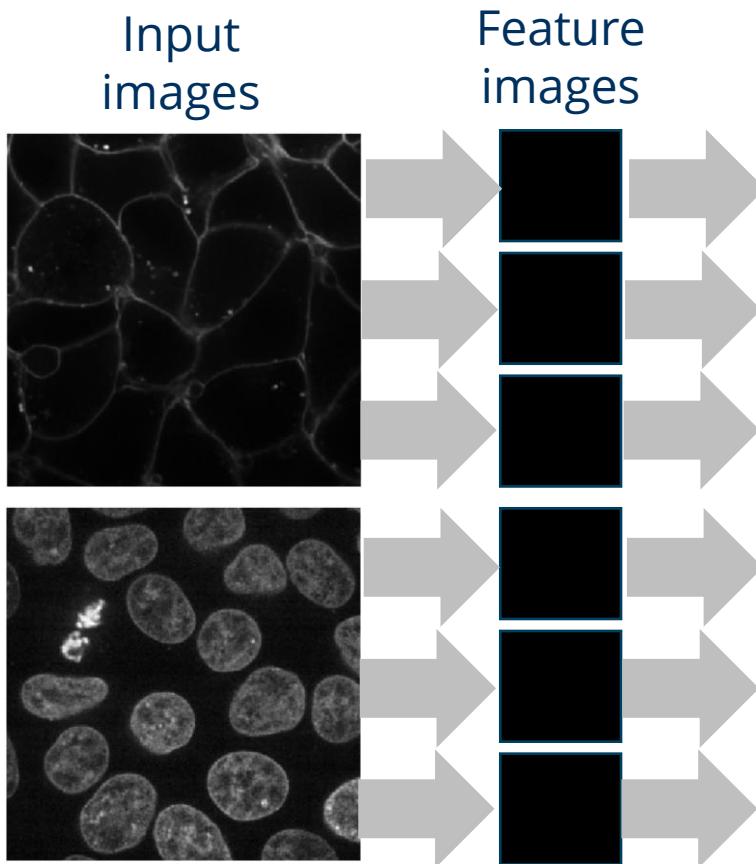
Wikipedia [1]

Relevant Aspects:

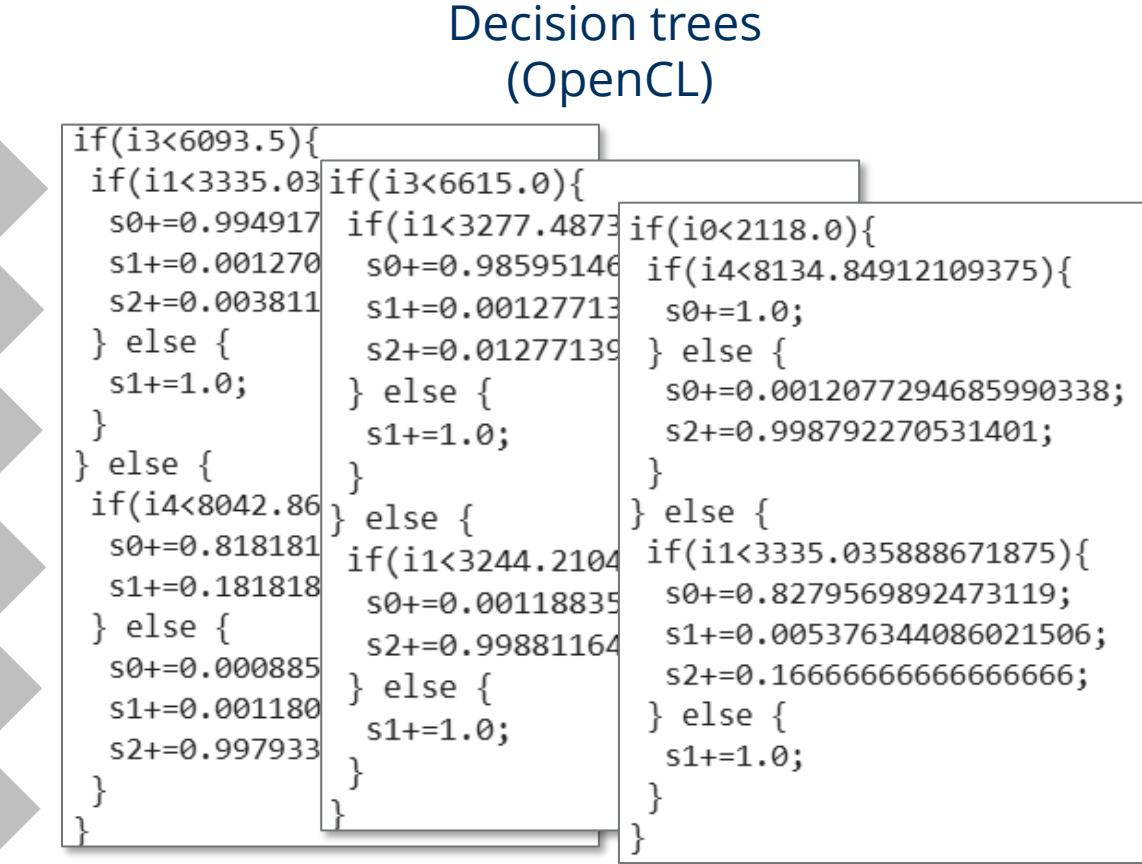
- Explainability vs. Interpretability of AI-algorithms
- We seek to enable humans to
 - predict results of AI Systems,
 - trust AI-Systems and
 - using AI-Systems effectively.

Explanation of Random Forest Classifiers

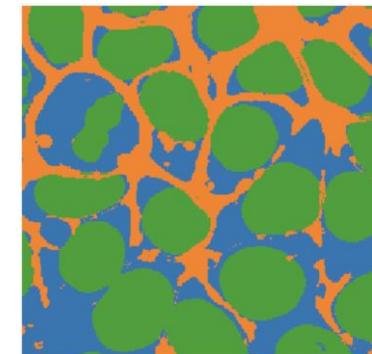
... by reading code



... is quite useless



Classification result images



Explainability

A logically consistent line of argumentation that depicts a situation or an algorithm with complete transparency.

Intrinsically explainable AI-algorithms

- Example: Linear Regression

$$f(x_1, x_2) = w_1 x_1 + w_2 x_2$$

If w_1 is much bigger than w_2 , the result depends much more on x_1 compared to x_2 .

Model
explainable

Results
predictable

Explainability

A logically consistent line of argumentation that depicts a situation or an algorithm with complete transparency.

Intrinsically explainable AI-algorithms

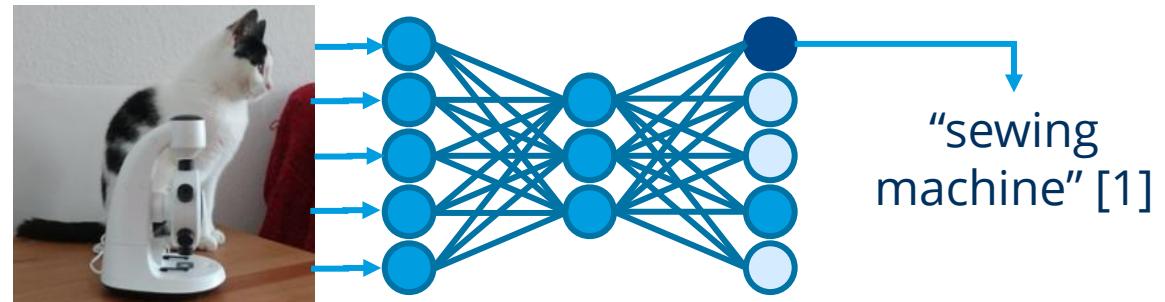
- Example: Linear Regression

$$f(x_1, x_2) = w_1 x_1 + w_2 x_2$$

If w_1 is much bigger than w_2 , the result depends much more on x_1 compared to x_2 .

Black-Box AI-algorithms

- Example: Deep Neural Networks (DNN)



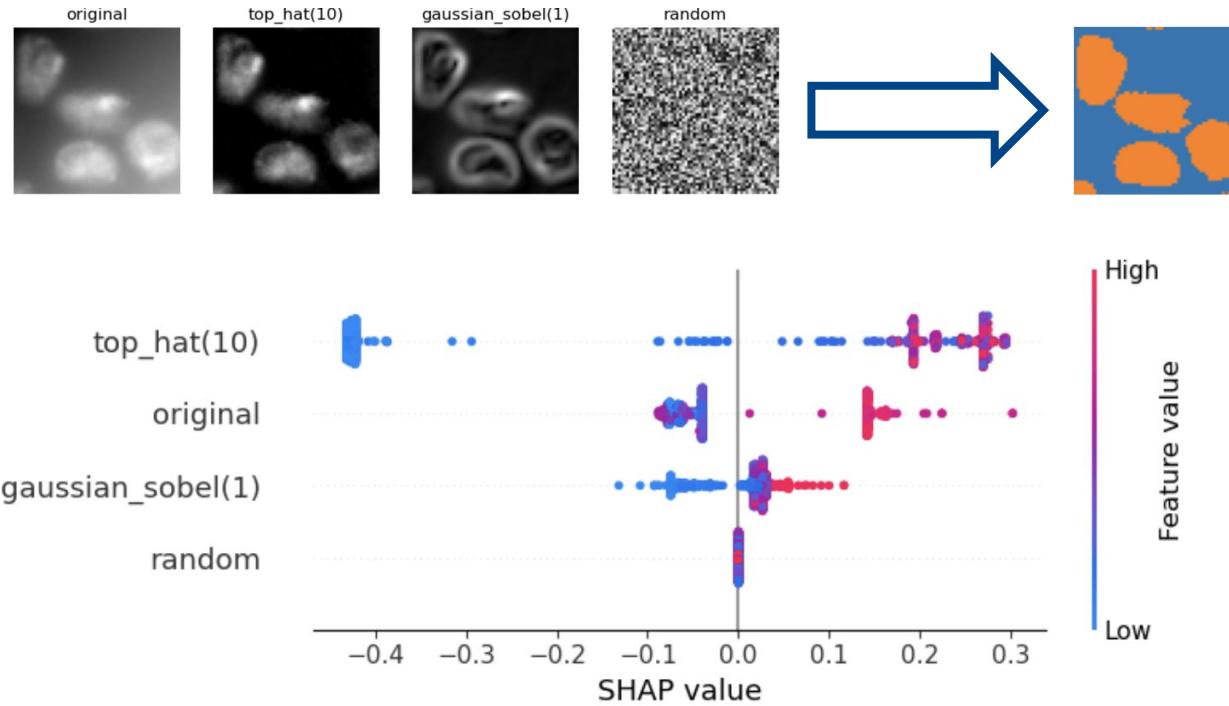
Not easily explainable and predictable

Interpretability

Visualization of intermediate results and their influence on results

Model-agnostic methods

Example: Shapley's Additive exPlanations (SHAP)

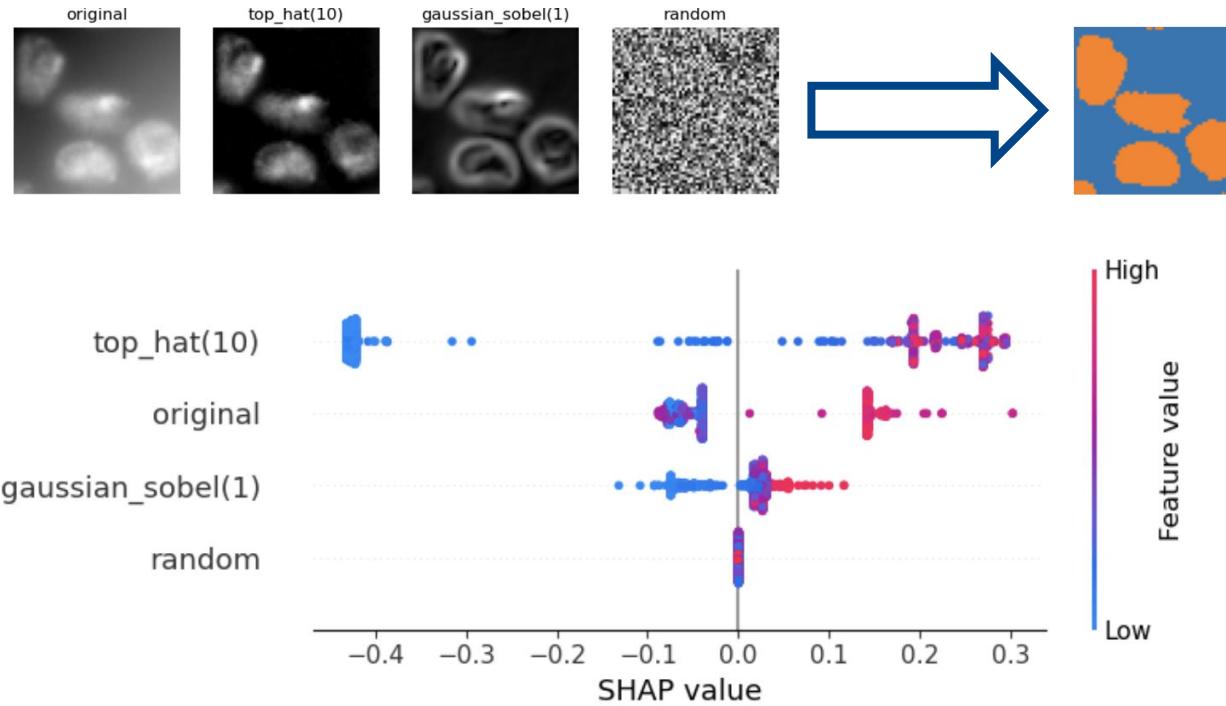


Interpretability

Visualization of intermediate results and their influence on results

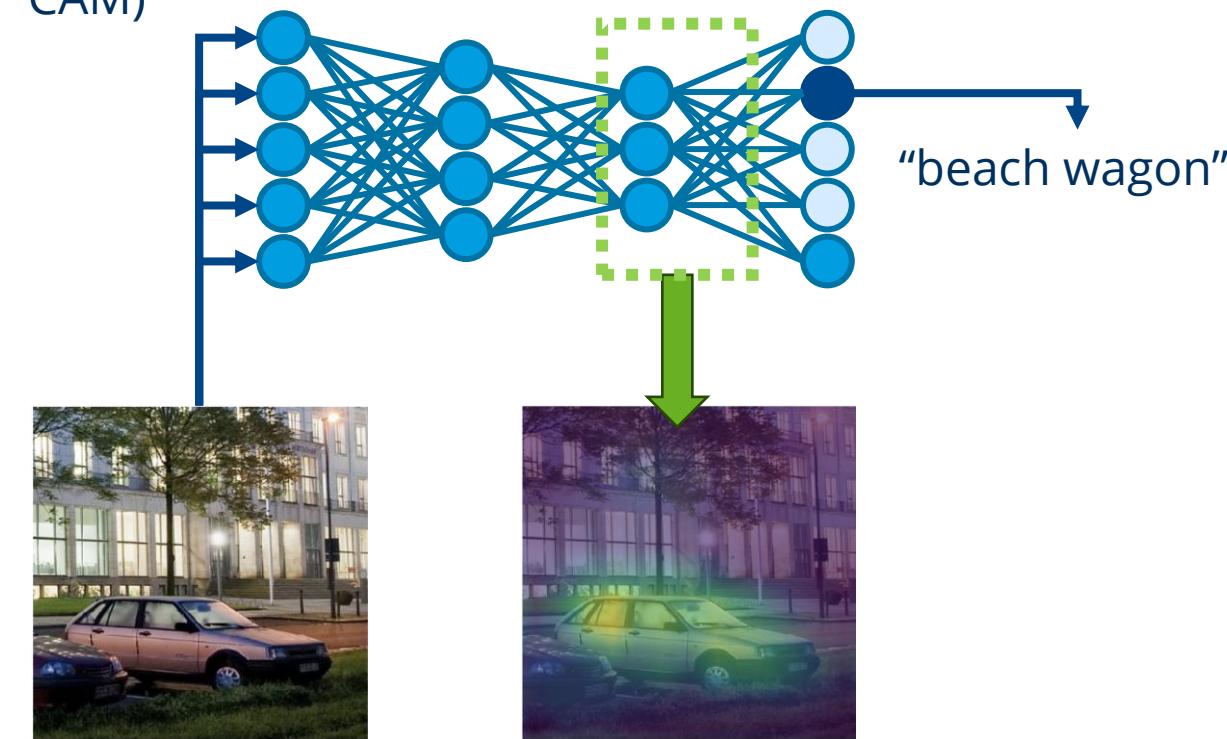
Model-agnostic methods

Example: Shapley's Additive exPlanations (SHAP)



Model-specific methods

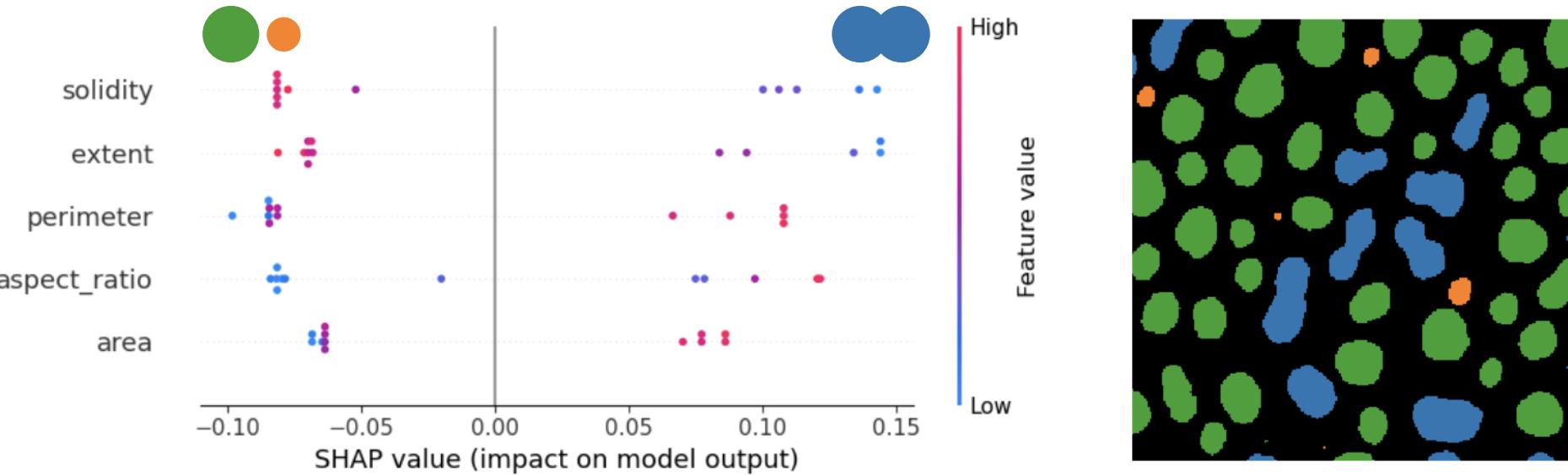
Example: Gradient Class Activation Maps (Grad-CAM)



Explainable AI

Depending on the target group [for the explanation], the influence of data is more important than how AI algorithms work.

- Many computer scientists want to explain and understand AI methods.
- Biologists use AI as a method to explain biological processes.
- Example: "What parameters distinguish **round objects** from **elongated ones**?"



Recap: Feature selection

- Which measurement / parameter / feature is related to the effect I'm investigating?
- Example goals:

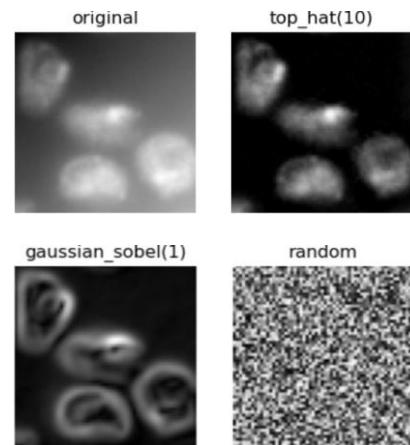


- Amplitude
- Energy
- Duration
- ...

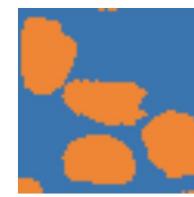


- Noise
- Tourists jumping on a sensor
- Earthquake approaching

Signal classification



Pixel classification



- Area
- Perimeter
- Aspect ratio
- ...



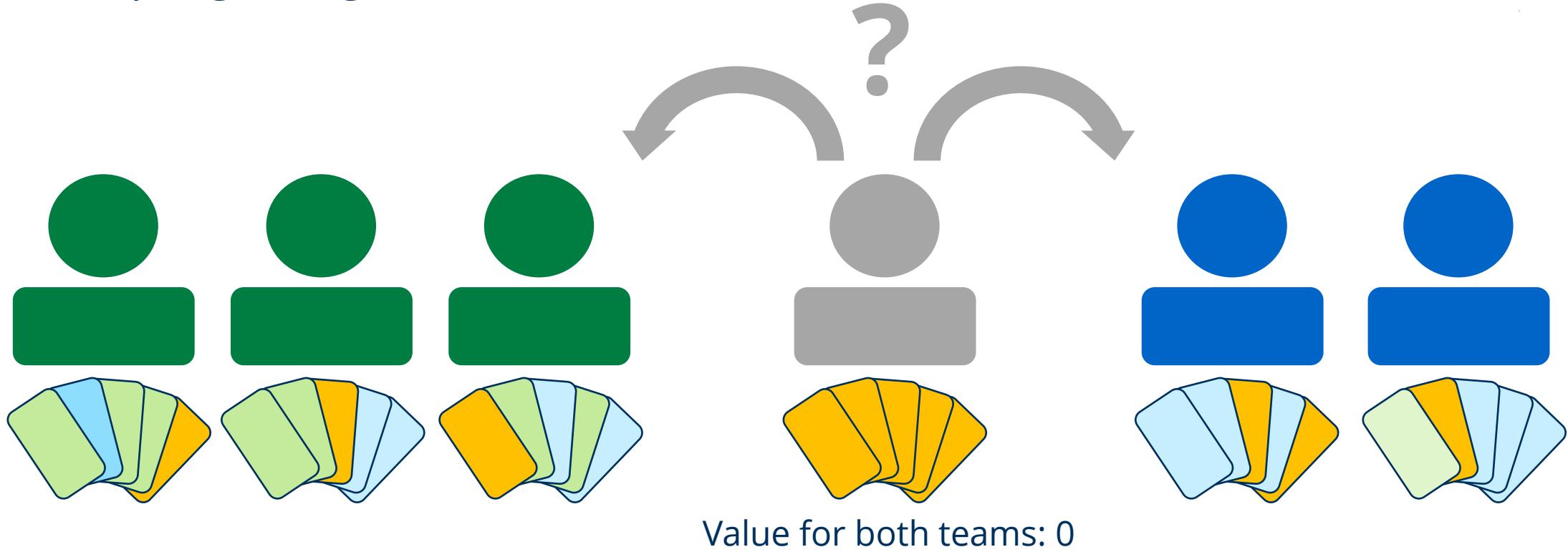
- Round
- Elongated

Object classification

Collaborative game theory

If players collaborate, how is the impact on a team if another player joins?

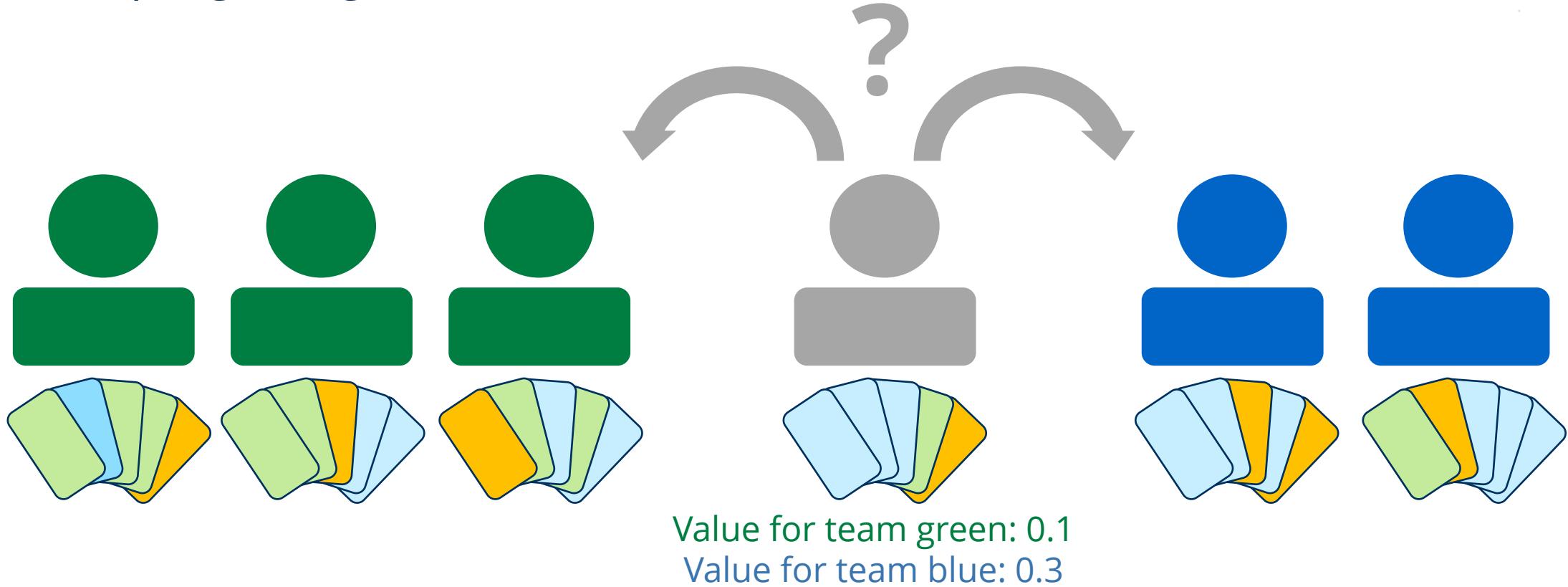
Example game goal: maximize cards of the same colour.



Collaborative game theory

If players collaborate, how is the impact on a team if another player joins?

Example game goal: maximize cards of the same colour.



SHAP

SHapley's Additive exPlanations

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F|-|S|-1)!}{|F|!} [f_x(S \cup \{i\}) - f_x(S)]$$

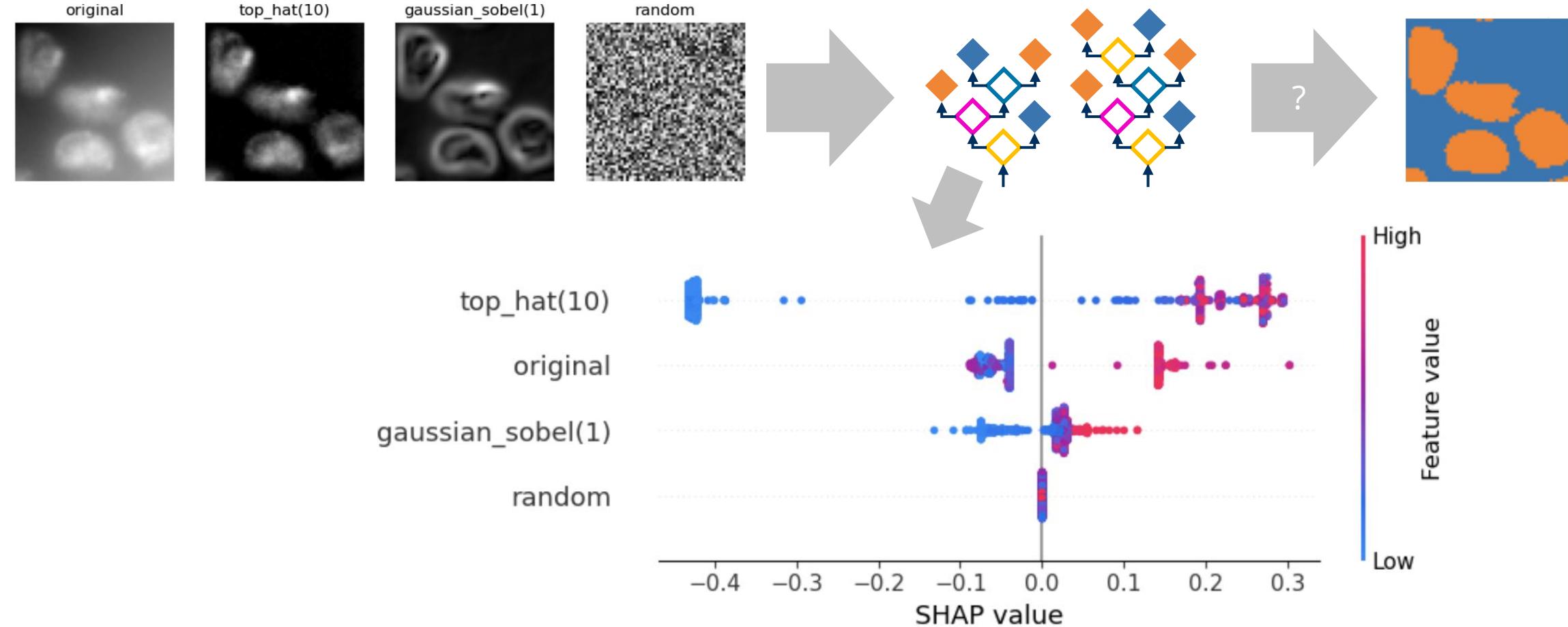
SHAP value of feature i	Sum over all Subsets of Features not including i	Weight related to number of used features in relation all players	Quality of classifier using feature i	Quality of classifier <i>not</i> using feature i
-------------------------	--	---	---------------------------------------	--

Game theory	SHAP value of player i	Sum over all Subsets of Players not including i	Weight related to number of <i>players in a coalition</i> in relation to <i>undecided players and all players</i>	Chance to win game of coalition <i>without</i> player i	Chance to win game of coalition <i>including</i> player i
-------------	------------------------	---	---	---	---

Analogously, this can be done with data points instead of features.

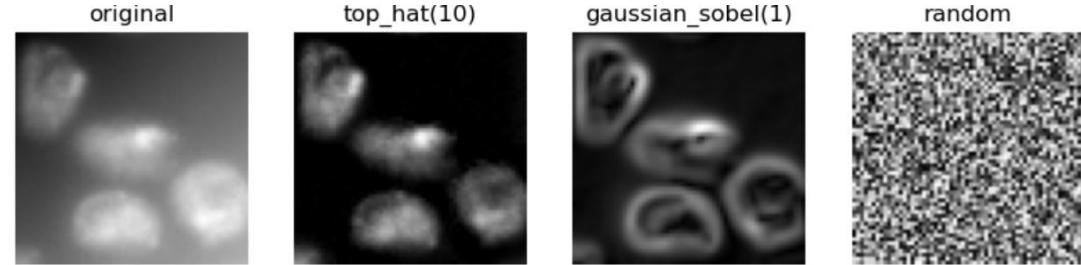
SHAP

Allows interpreting [pixel] classification results

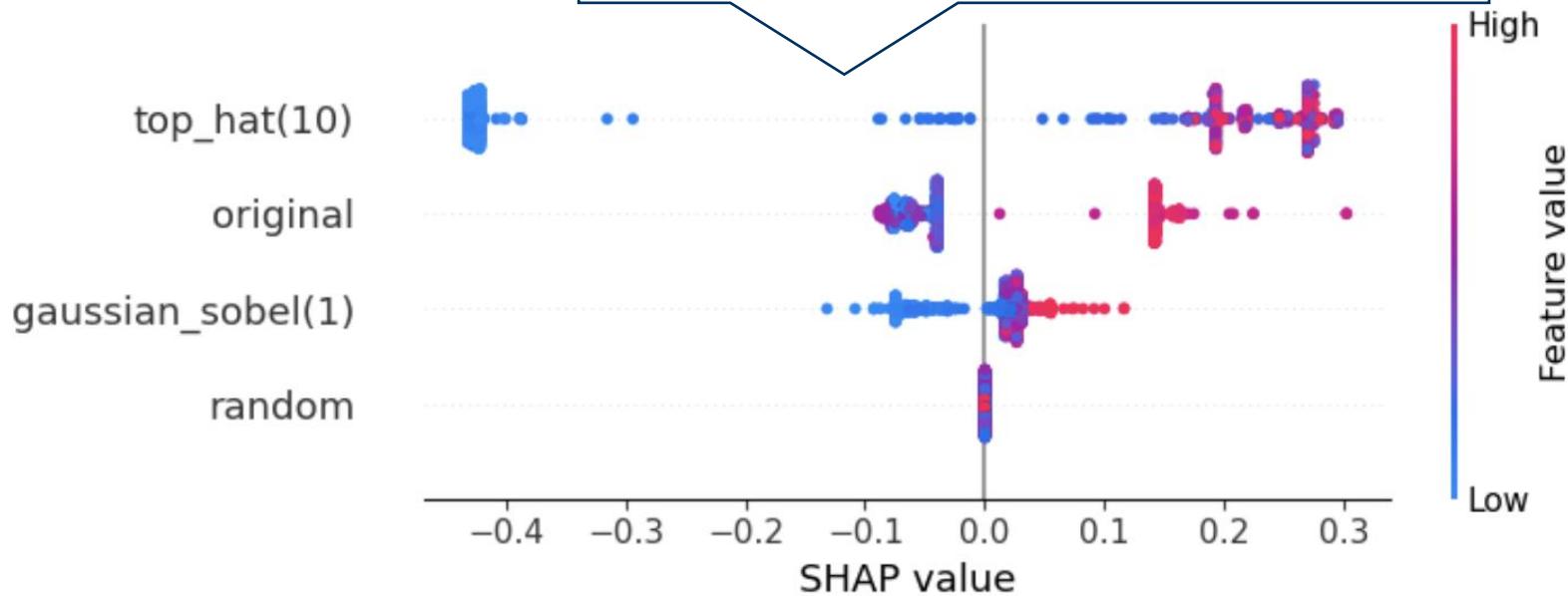
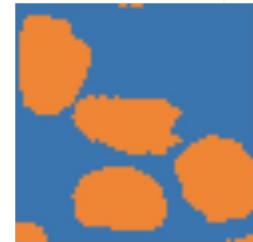


SHAP

Allows interpreting [pixel] classification

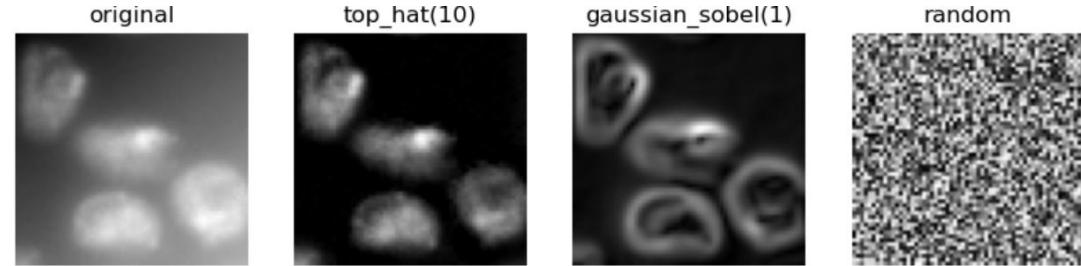


"If intensity in the top-hat image is high, the classifier tends to select the positive class (orange)."

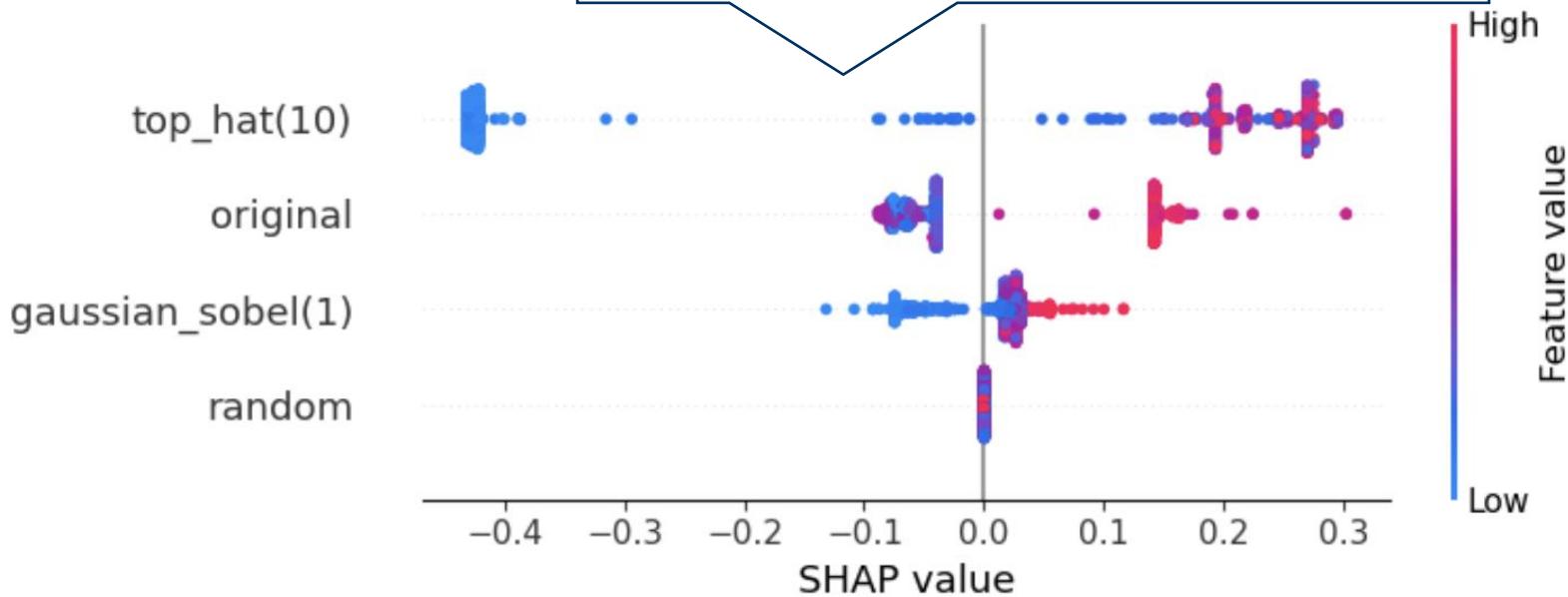
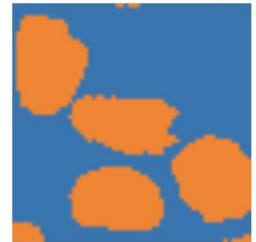


SHAP

Allows interpreting [pixel] classification

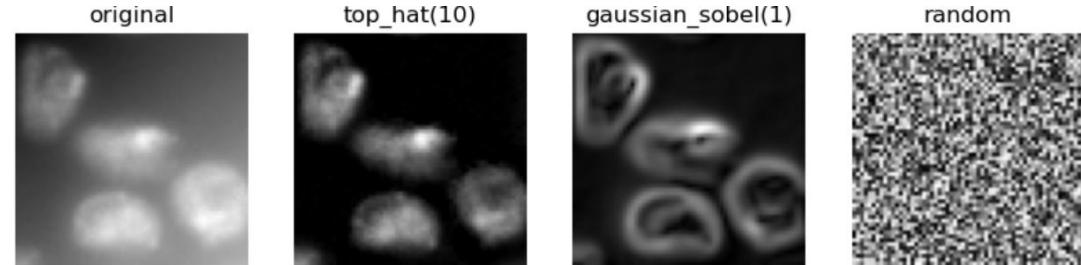


"If intensity in the top-hat image is low, the classifier needs to take other features into account."

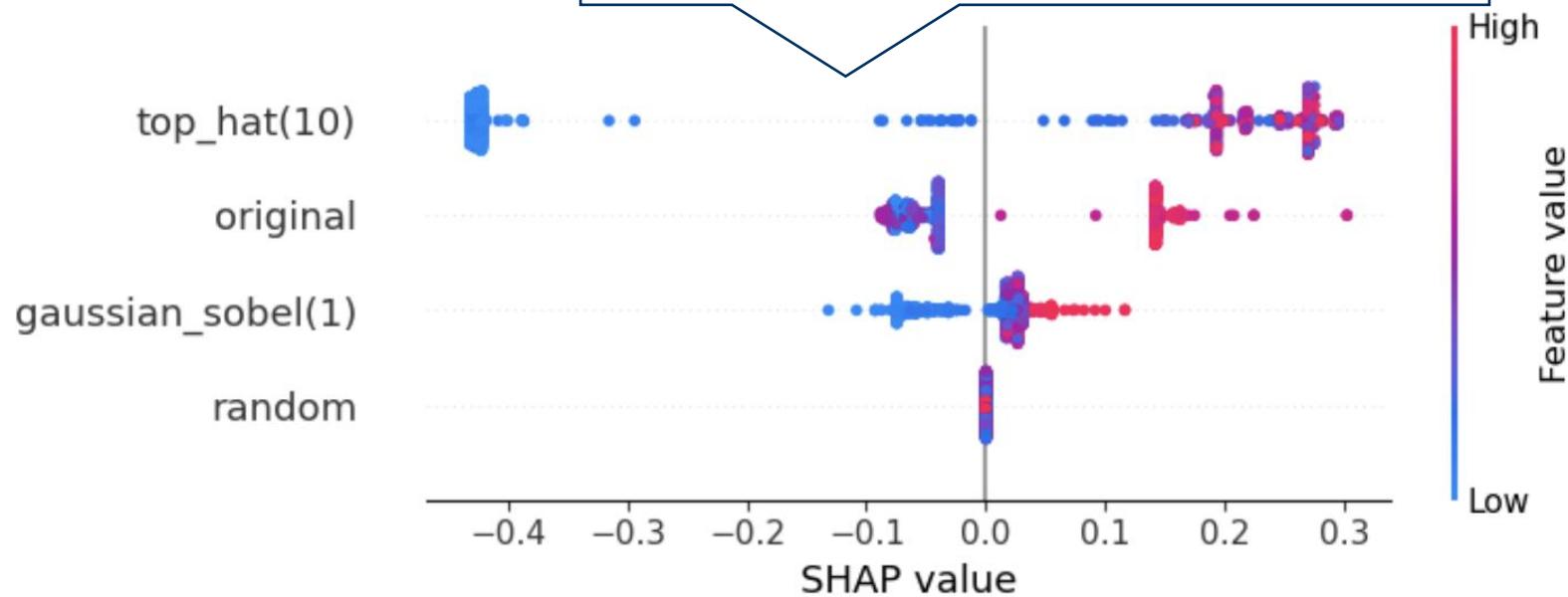


SHAP

Allows interpreting [pixel] classification

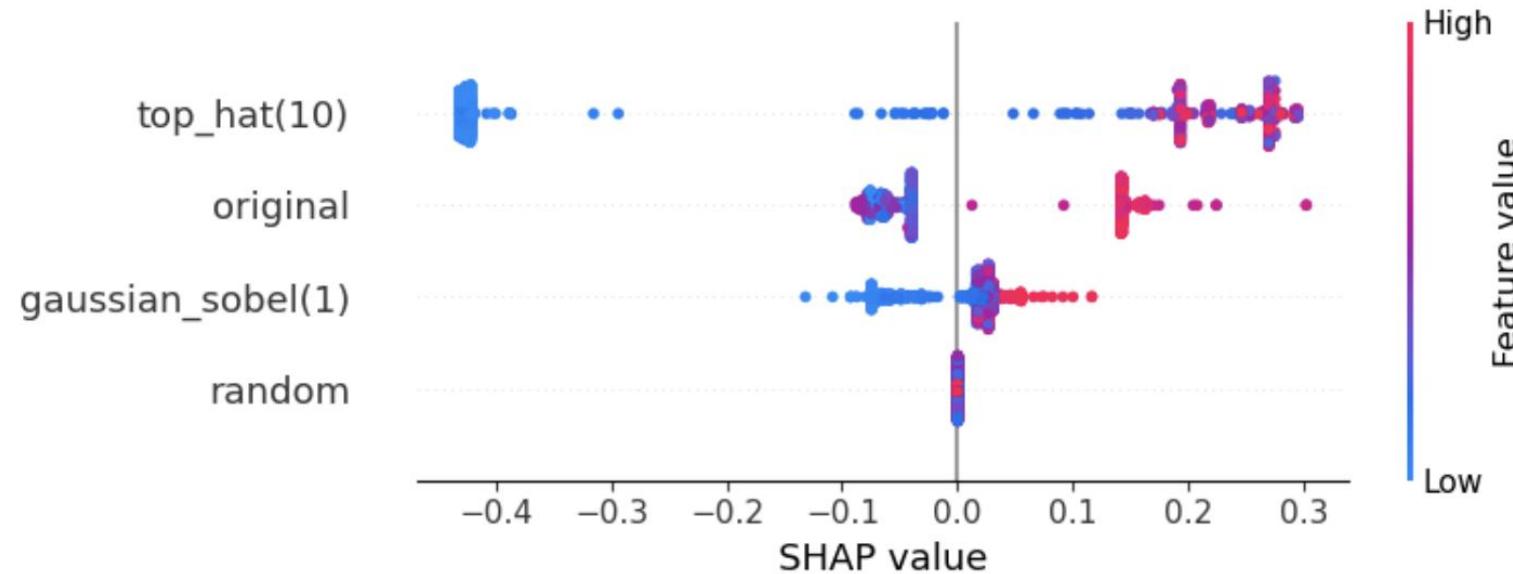
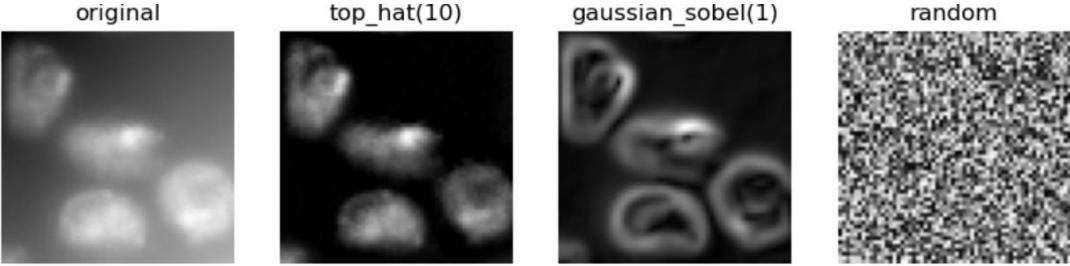


"The random feature has no value for classification."



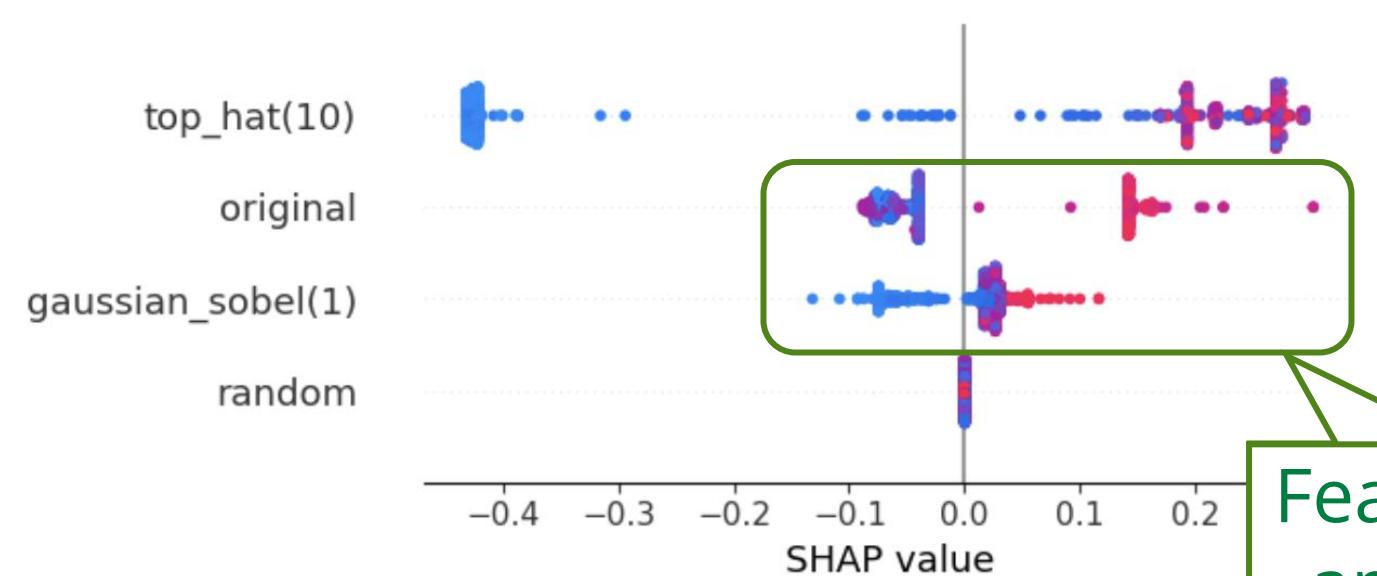
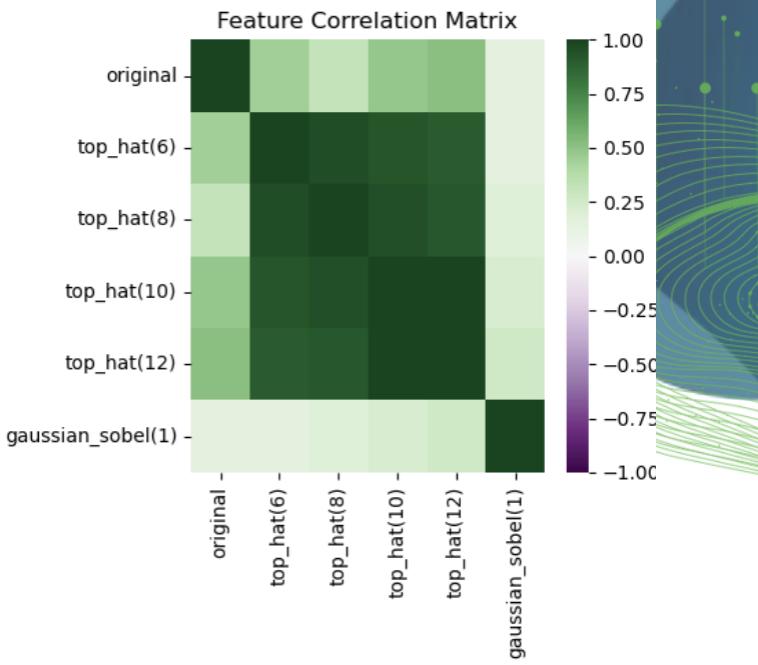
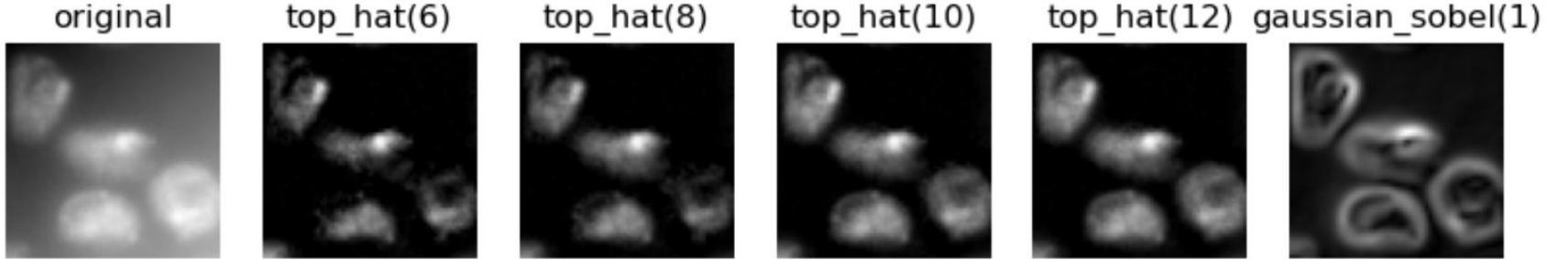
Pitfall: Correlation

Correlated features may harm interpretability

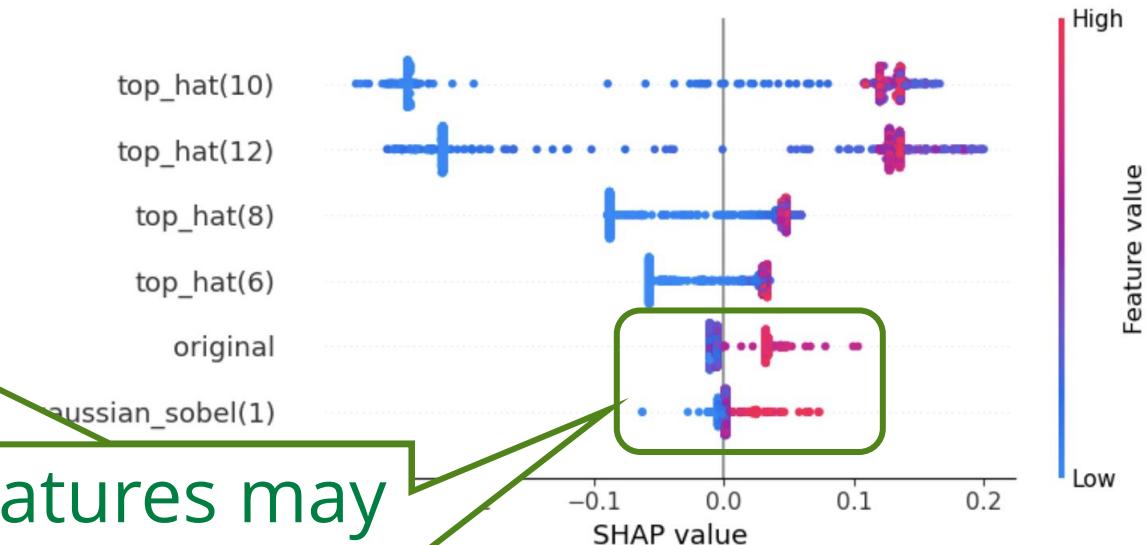


Pitfall: Correlation

Correlated features may harm interpretability

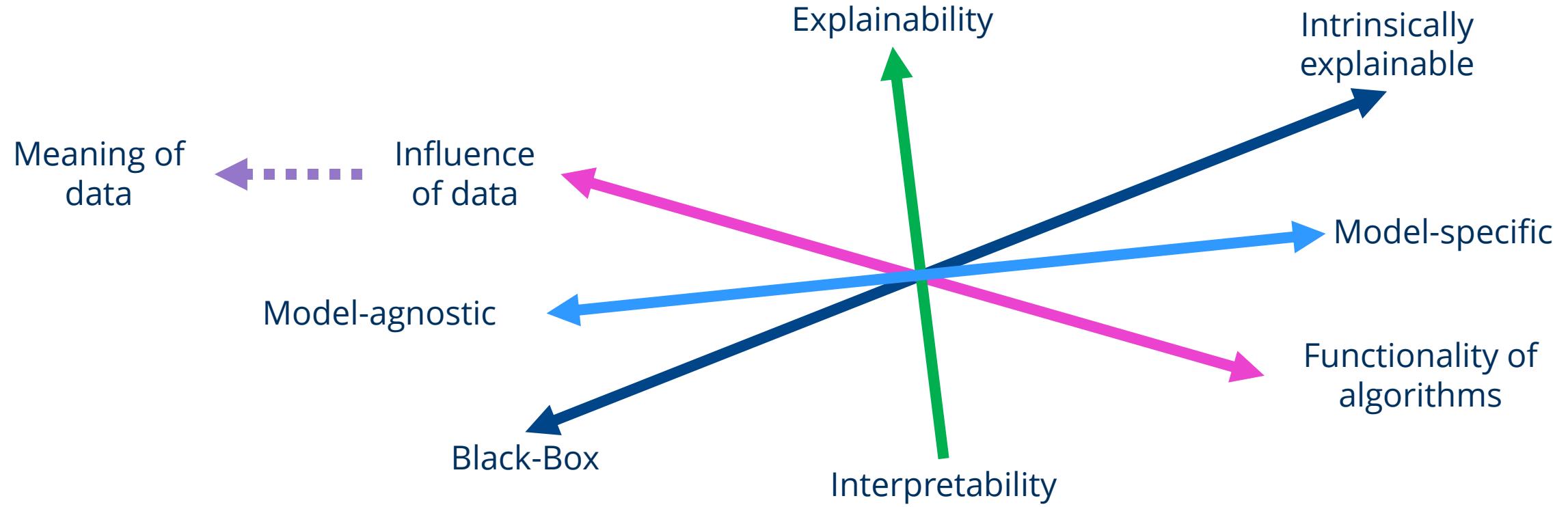


Features may appear less valuable.



Summary: Explainable AI

Methods of XAI can be classified on different scales



Exercises

Robert Haase

Funded by



Bundesministerium
für Bildung
und Forschung



Diese Maßnahme wird gefördert durch die Bundesregierung
aufgrund eines Beschlusses des Deutschen Bundestages.
Diese Maßnahme wird mitfinanziert durch Steuermittel auf
der Grundlage des von den Abgeordneten des Sächsischen
Landtags beschlossenen Haushaltes.

Exercise: Feature exploration

Use dimensionality reduction to elaborate features that might allow round and elongated objects

The image shows a Jupyter Notebook interface with several windows open, illustrating the process of feature exploration for identifying round and elongated objects.

- Window 1:** Displays a scatter plot of colored blobs (blobs.tif) with axes labeled from 0 to 250.
- Window 2:** Shows correlation statistics for various parameters (label, area, bbox_area, equivalent_diameter, convex_area, max_intensity, mean_intensity). The correlation matrix is as follows:

	label	area	bbox_area	equivalent_diameter	convex_area	max_intensity	mean_intensity
label	1.00000	0.261682	0.223070	0.249249	0.250594	0.110791	
area	0.261682	1.00000	0.973718	0.978723	0.997560	0.511730	
bbox_area	0.223070	0.973718	1.00000	0.948328	0.985584	0.481524	
equivalent_diameter	0.249249	0.978723	0.948328	1.00000	0.974614	0.633984	
convex_area	0.250594	0.997560	0.985584	0.974614	1.00000	0.506730	
max_intensity	0.110791	0.511730	0.481524	0.633984	0.506730	1.00000	
mean_intensity	0.235692	0.530250	0.476951	0.618553	0.517356	0.825115	
min_intensity	nan	nan	nan	nan	nan	nan	

The notebook also contains code for colorizing styler and plotting the correlation matrix.

TERACTION: A note about using a lasso-annotation to identify clusters corresponding to 8-shaped objects.

Exercise: The UMAP-generation above is done without parameters such as centroid and orientation. Why?

Exercise: Repeat the procedure above with the dataset `human_mitosis`. Identify parameters for differentiating the small bright cells from the others. (hint)

```
[ ]: image = human_mitosis()  
stackview.insight(image)
```

Pixel classification / object segmentation

Use Napari to segment objects

Interactive pixel classification and object segmentation in Napari

In this exercise we will train a [Random Forest Classifier](#) for pixel classification and convert the result in an instance segmentation. We will use the napari plugin [napari-accelerated-pixel-and-object-classification](#).

Getting started

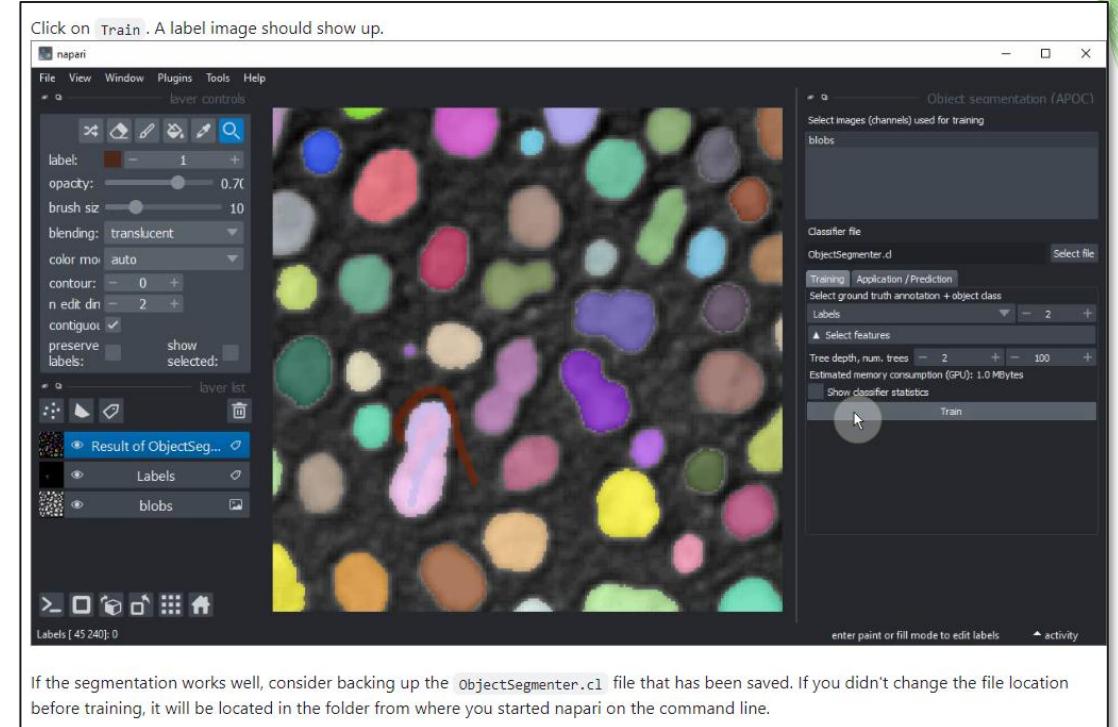
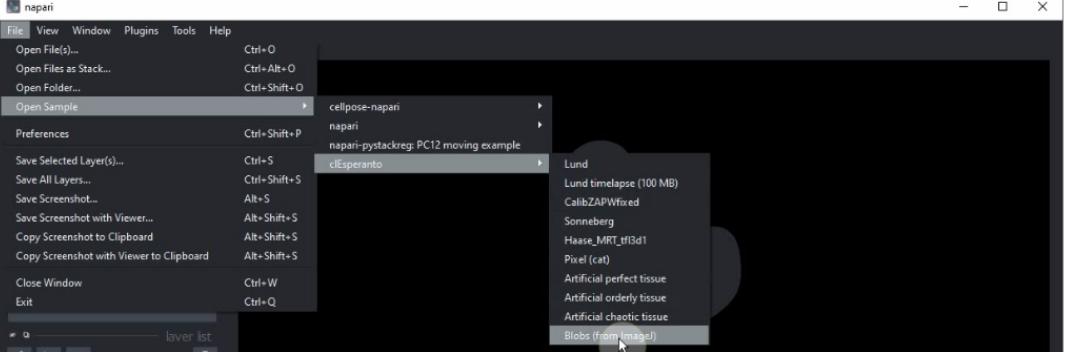
Open a terminal window and activate your conda environment:

```
conda activate devbio-napari-env
```

Afterwards, start up Napari:

```
napari
```

Load the "Blobs" example dataset from the menu File > Open Sample > c1Esperanto > Blobs (from ImageJ)



Object classification

Use Napari to classify round and elongated objects

Interactive object classification in Napari

In this exercise we will train a [Random Forest Classifiers](#) for classifying segmented objects. We will use the napari plugin [napari-accelerated-pixel-and-object-classification](#).

Getting started

Open a terminal window and activate your conda environment:

```
conda activate devbio-napari-env
```

Afterwards, start up Napari:

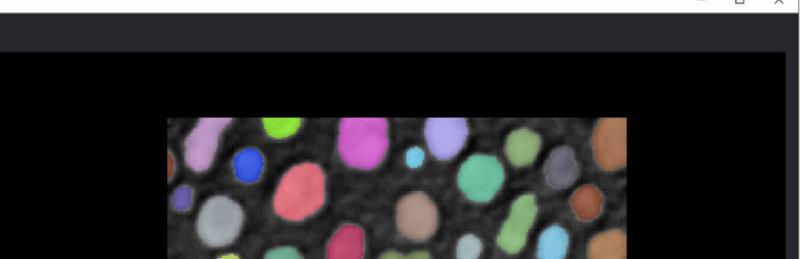
```
napari
```

Load the "Blobs" example dataset from the menu [File > Open Sample > c1Esperanto > Blobs \(from Image\)](#)

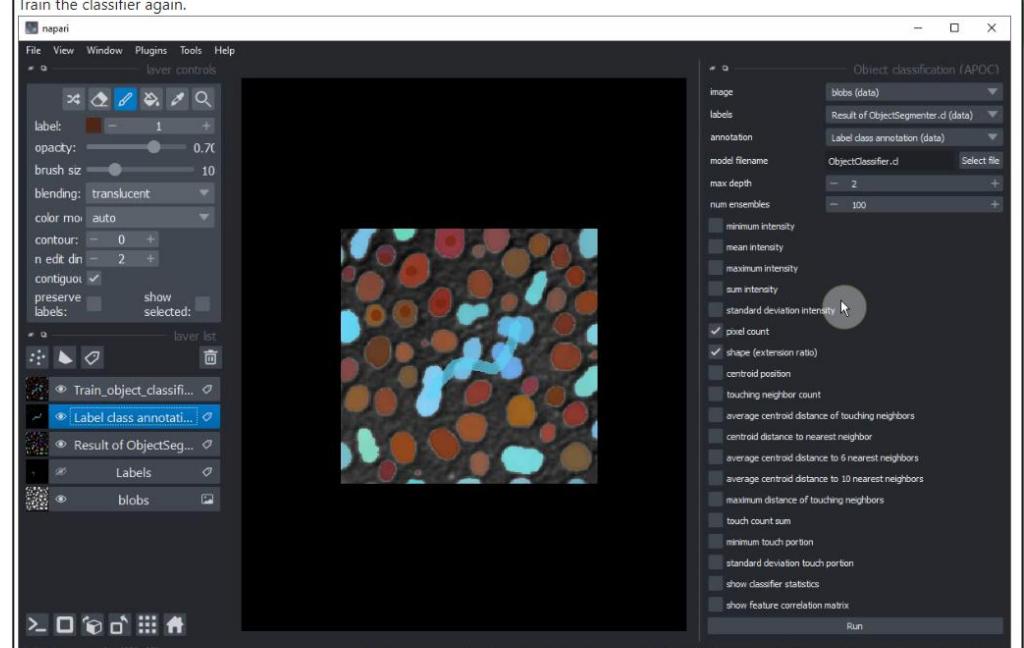
We furthermore need a label image. You can create it using the pixel classifier trained earlier or using the menu [Tools > Segmentation / labeling > Gauss-Otsu Labeling \(clesperanto\)](#).

Object classification

Our starting point is a loaded image and a label image with segmented objects. The following procedure is also shown in [this video](#).



Train the classifier again.



If you are happy with the trained classifier, copy the file to a safe place. When training the next classifier this one might be overwritten.

Extra exercise

Retrain the classifier so that it can differentiate three different classes:

- Small round objects
- Large round objects
- Large elongated objects

Supervised machine learning using Python

Use scikit-learn in Jupyter Notebooks to train and apply Random Forest Classifiers

The screenshot shows a Jupyter Notebook interface with two tabs: 'interactive_parameter_explor' and '01_scikit_learn_random_forest'. The main content area displays a section titled 'Pixel classification using Scikit-learn'. It includes a brief description of pixel classification, a 'See also' section with links to 'Scikit-learn random forest' and 'Classification of land cover by Chris Holden', and a code cell [3] containing the following Python code:

```
from sklearn.ensemble import RandomForestClassifier  
  
from skimage.io import imread  
import numpy as np  
import napari  
import stackview  
import matplotlib.pyplot as plt
```

Below the code, a note states: 'We need an example image and a corresponding annotation.' A code cell [6] shows the following code:

```
image = imread('data/blobs.tif')  
  
annotation = imread('data/blobs_annotations.tif')
```

The status bar at the bottom indicates: 'Simple 0 2 Python 3 (ipykernel) | ... Mode: Comma... Ln 1, Co... 01_scikit_learn_random_forest_pixel_classifier.ip... 1 1'

The screenshot shows a Jupyter Notebook interface with two tabs: 'interactive_parameter_explor' and '01_scikit_learn_random_forest'. The main content area displays a note: 'Go ahead after annotating at least two regions with labels 1 and 2. Take a screenshot of the annotation:' followed by a yellow arrow pointing to the napari viewer window. The napari viewer window shows a grayscale image of blobs with two regions annotated: one blue and one red. The status bar at the bottom indicates: 'Simple 0 2 Python 3 (ipykernel) | ... Mode: Comma... Ln 1, Co... 01_scikit_learn_random_forest_pixel_classifier.ip... 1 1'

SHAP Analysis in Python

Use the opportunity
and explain SHAP
plots like this one!

