CENTER FOR SCALABLE DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE

Supervised and Unsupervised Machine Learning for Bio-image Analysis

Robert Haase

Funded by

Bundesministerium für Bildung und Forschung

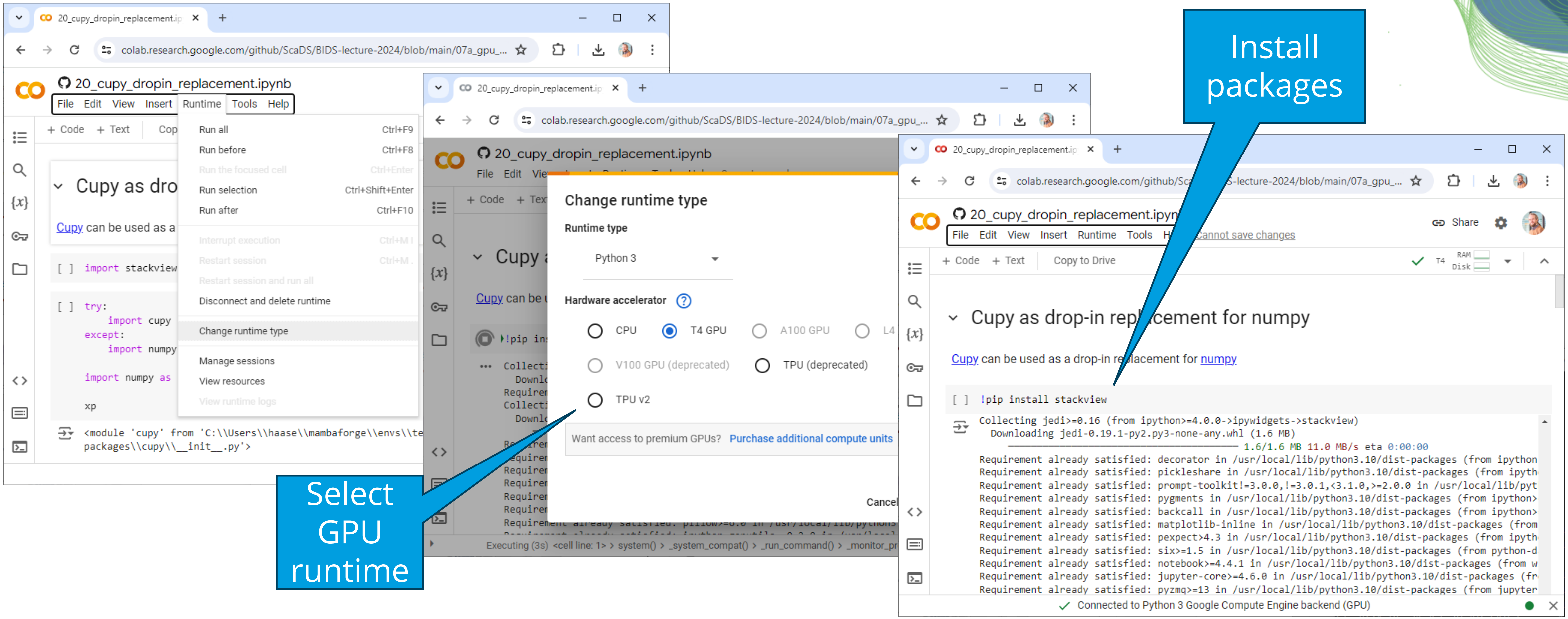
SACHSEN

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.

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.

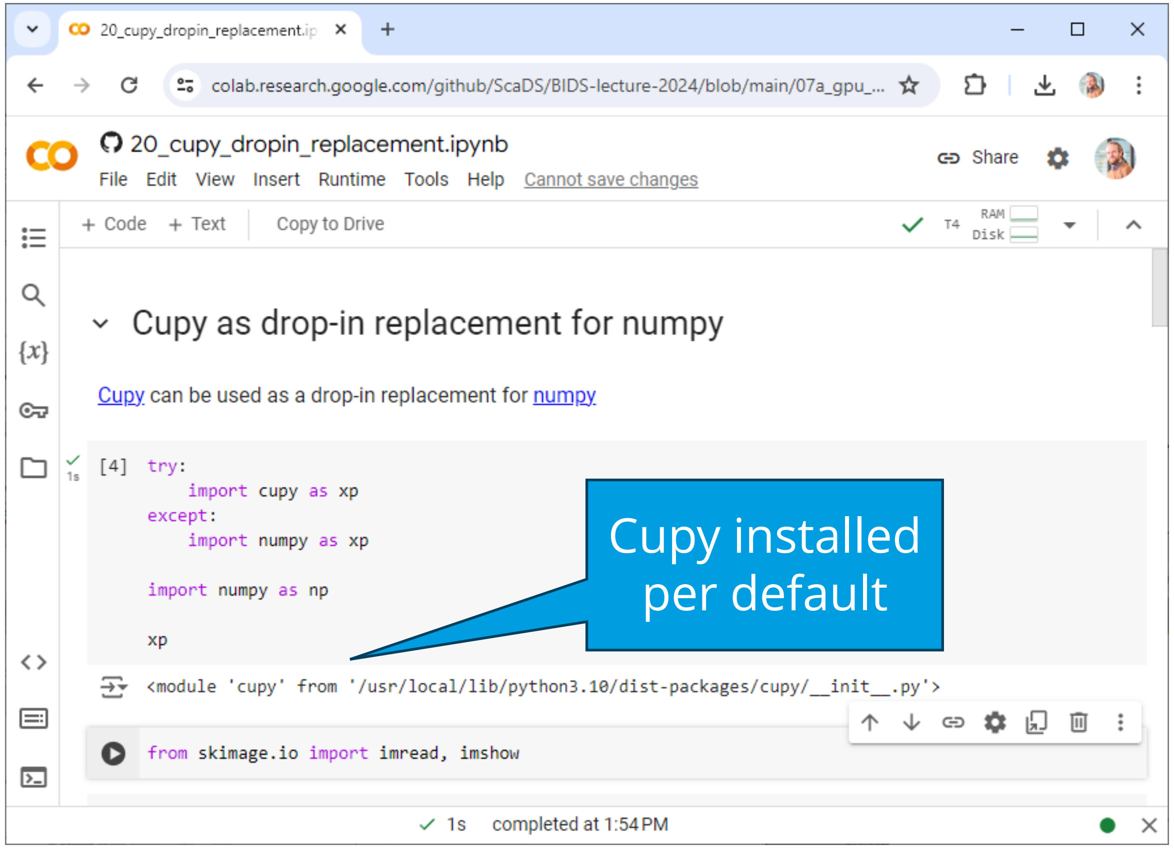
Follow-up: cupy

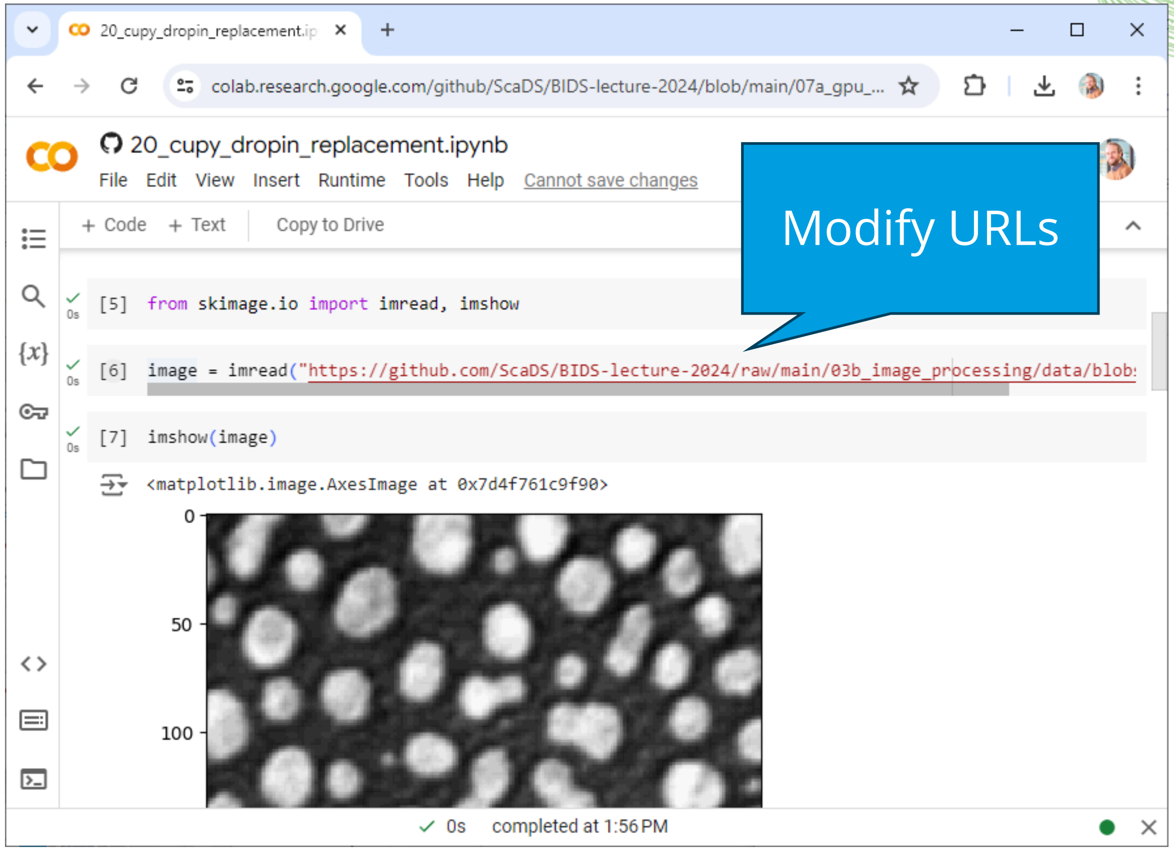
cupy was hard to make work. Consider using a GPU-runtime in Google Colab



Follow-up: cupy

cupy was hard to make work. Consider using a GPU-kernel in Google Colab

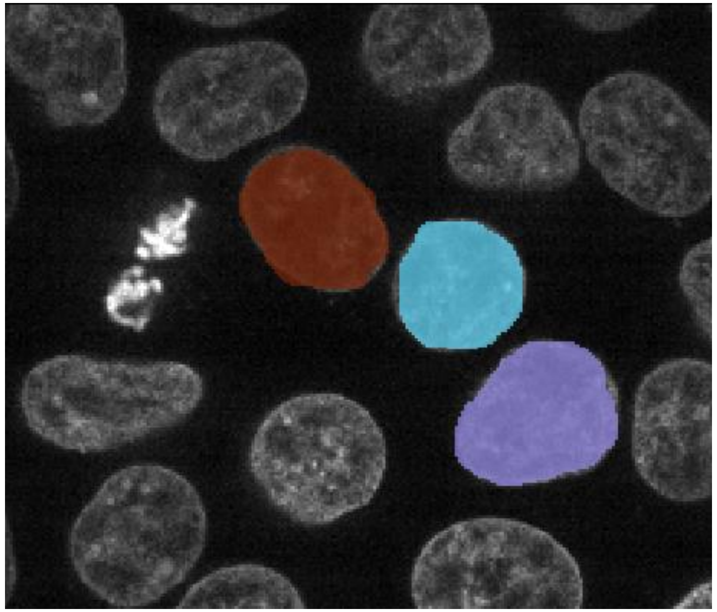




Robert Haase
@haesleinhuepf

Quiz: Recap

What kind of label image is this?



Instance segmentation

Semantic segmentation

Sparse instance segmentation

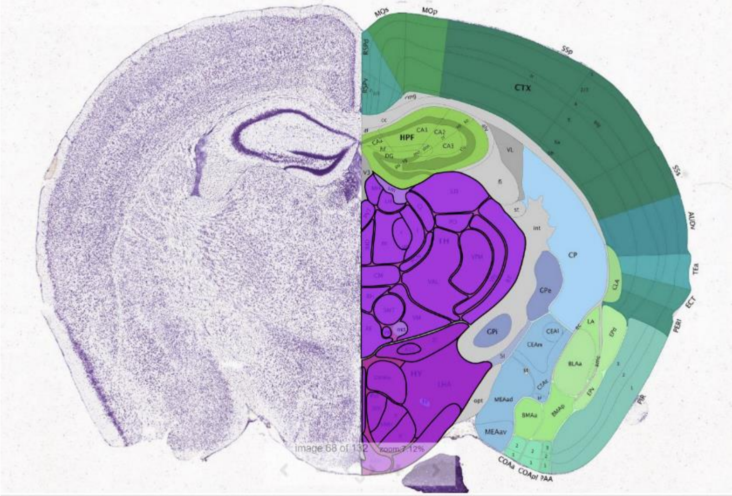
Sparse semantic segmentation

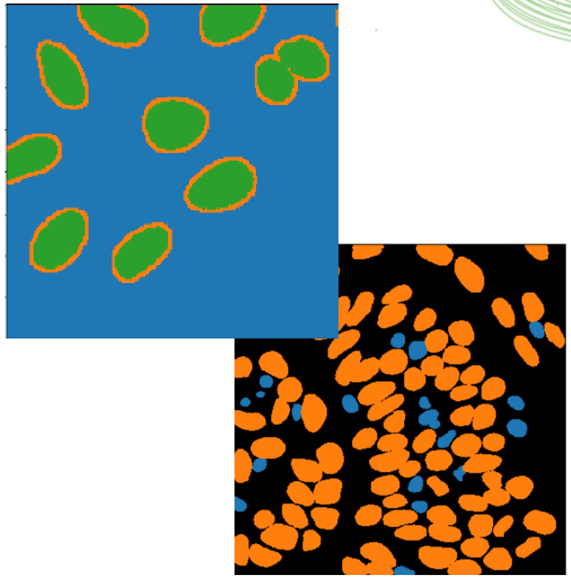
Terminology

Instance segmentation

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

Semantic segmentation



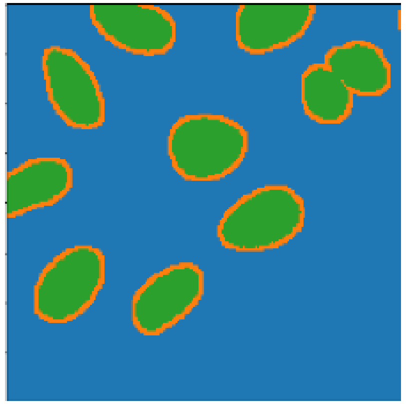


Regions:
- Anatomical, geographical
- All pixels belonging to the same type of object have the same value

Source: Allen Mouse Brain Atlas [dataset]. Available from mouse.brain-map.org. Allen Institute for Brain Science (2011).

Terminology

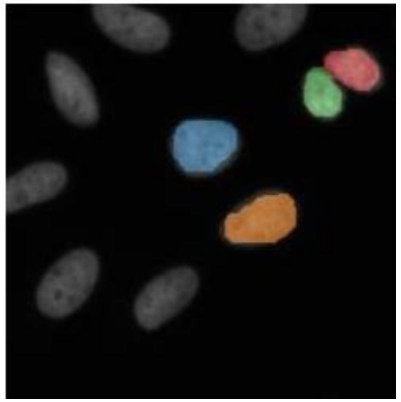
Instance segmentation

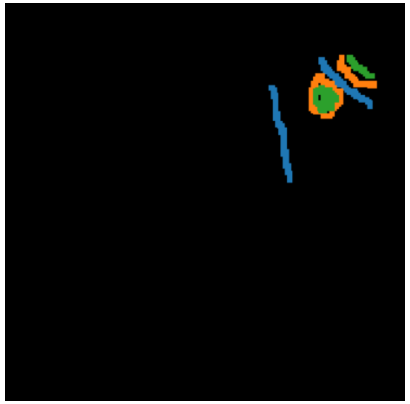


Semantic segmentation

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

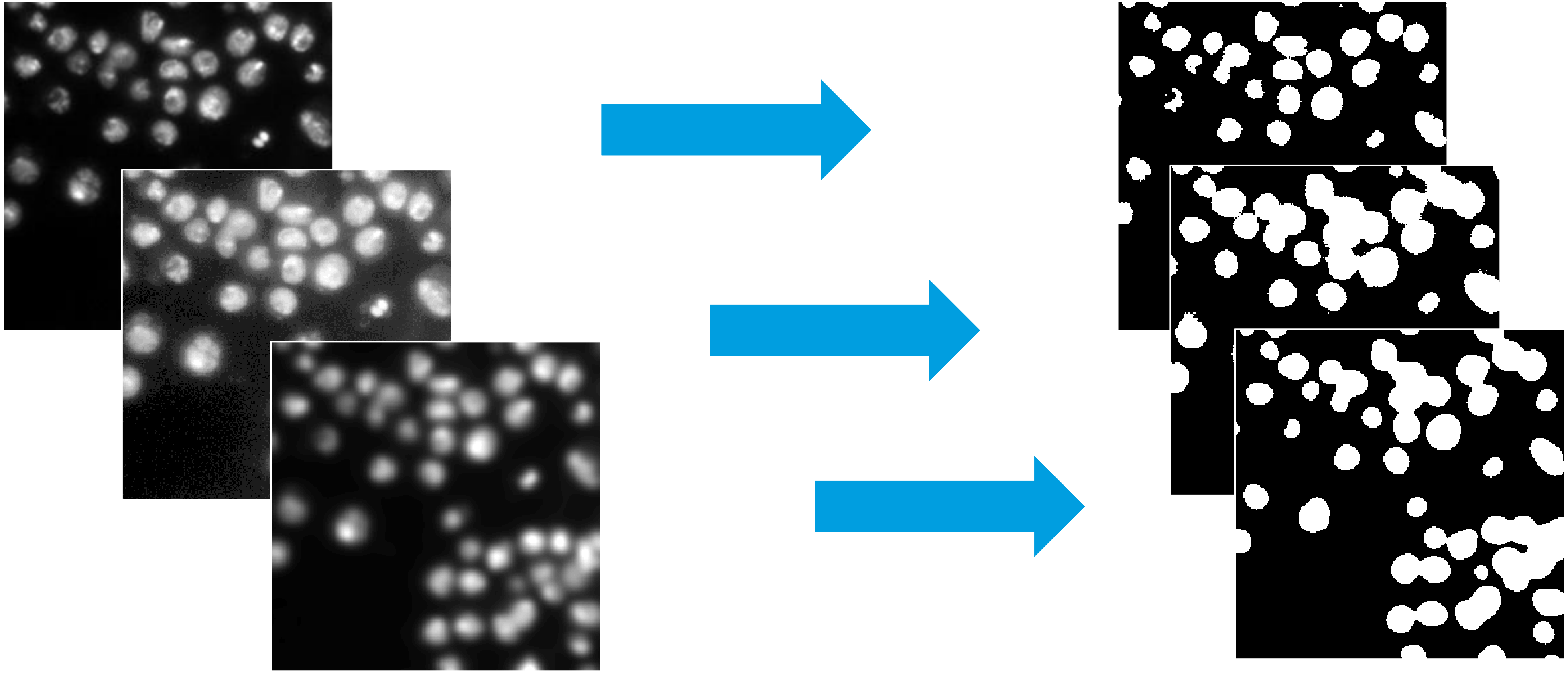


Image data source: BBBC038v1, available from the Broad Bioimage Benchmark Collection (Caicedo et al., Nature Methods, 2019].

Image segmentation using thresholding

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

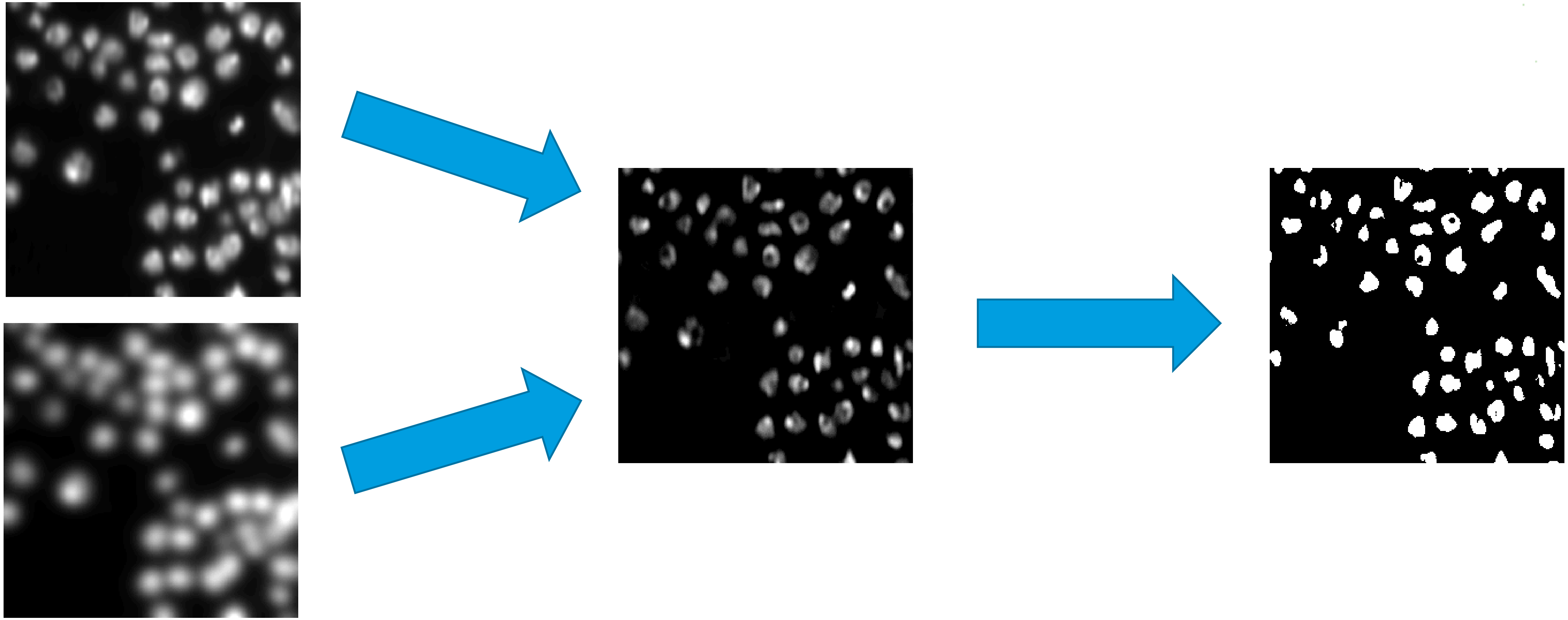


Image data source: BBBC038v1, available from the Broad Bioimage Benchmark Collection (Caicedo et al., Nature Methods, 2019].

Image segmentation using thresholding

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

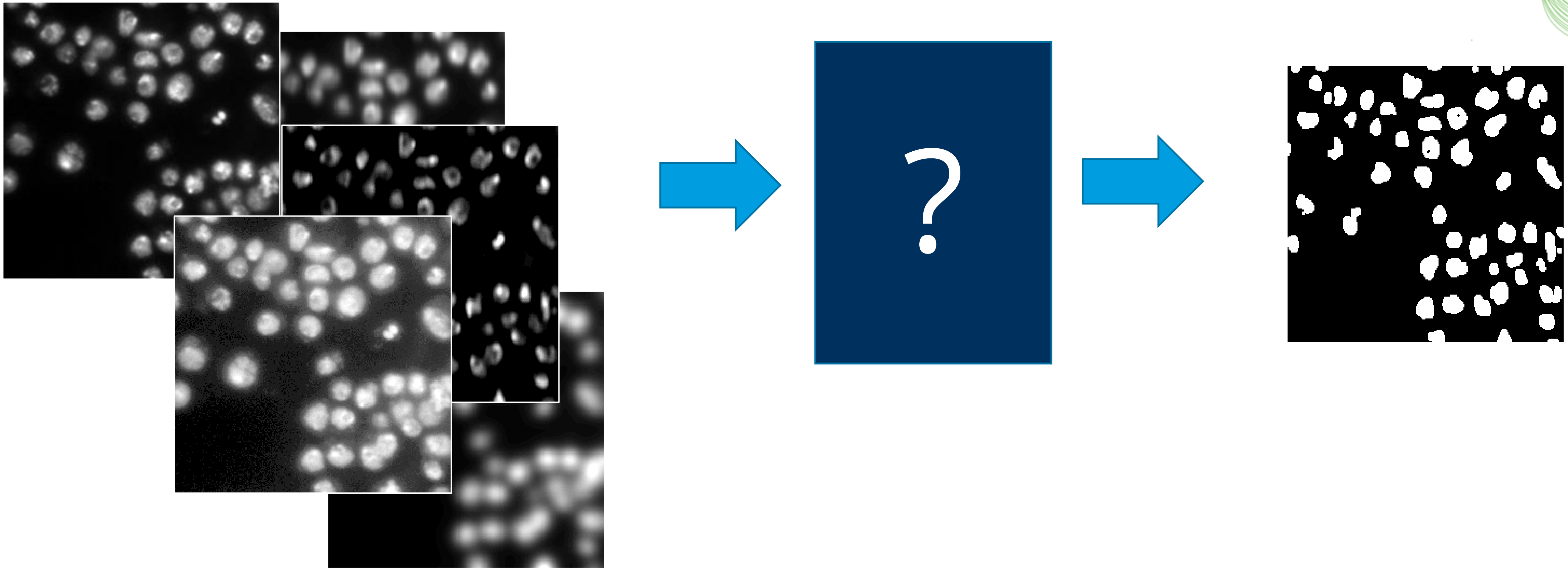
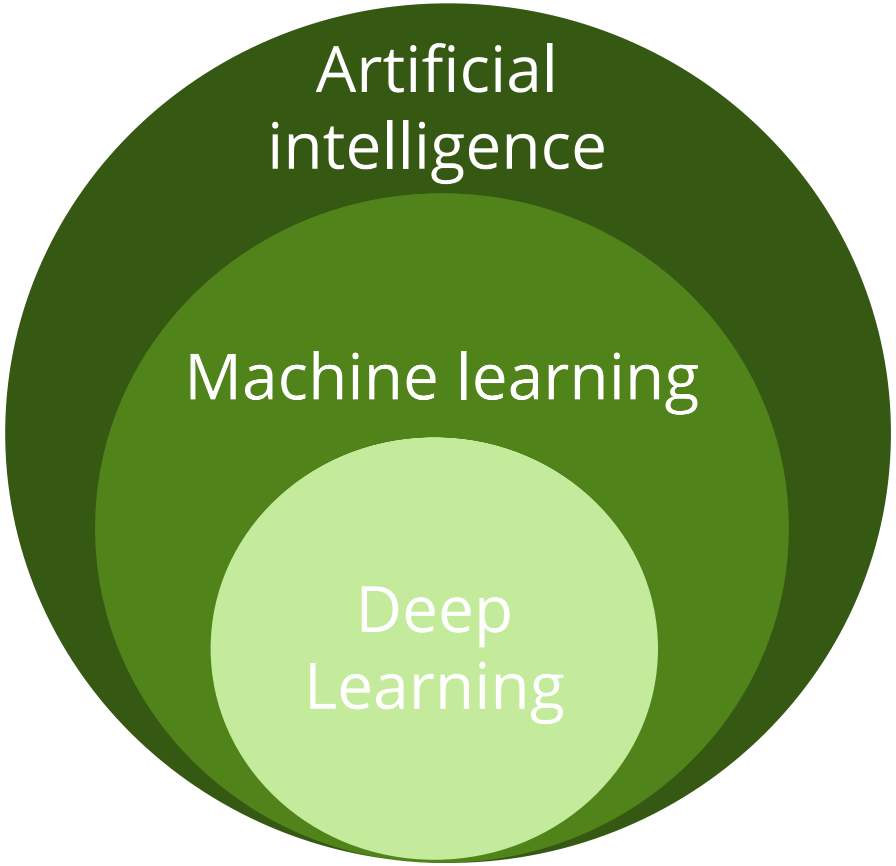
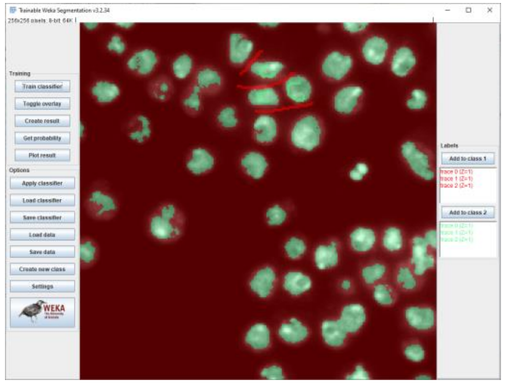


Image data source: BBBC038v1, available from the Broad Bioimage Benchmark Collection (Caicedo et al., Nature Methods, 2019].

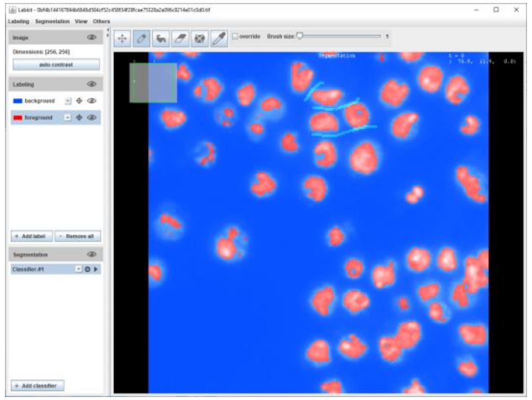
Machine learning

Finds more and more applications, also in life sciences.

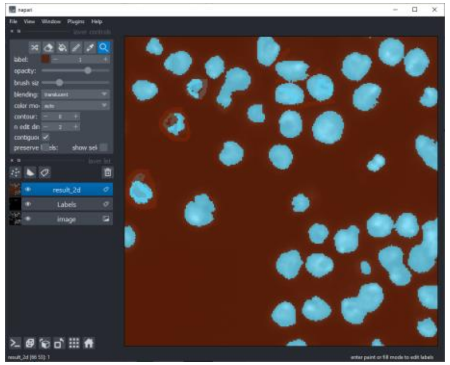




Trainable Weka Segmentation
https://imagej.net/plugins/tws/



LabKit
https://imagej.net/plugins/labkit/



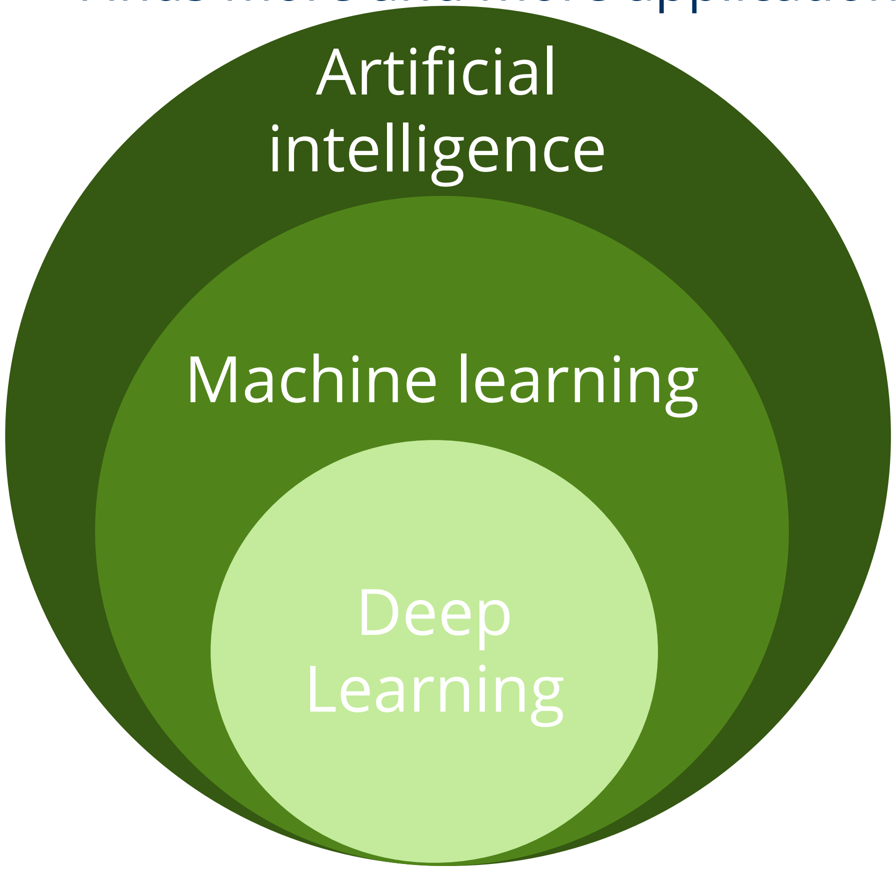
Python / scikit-learn / napari / apoc

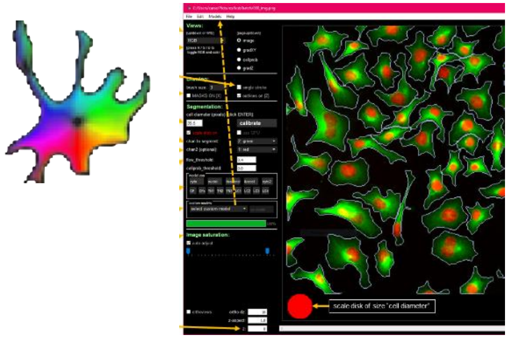
Image data source: BBBC038v1, available from the Broad Bioimage Benchmark Collection (Caicedo et al., Nature Methods, 2019].

Machine learning

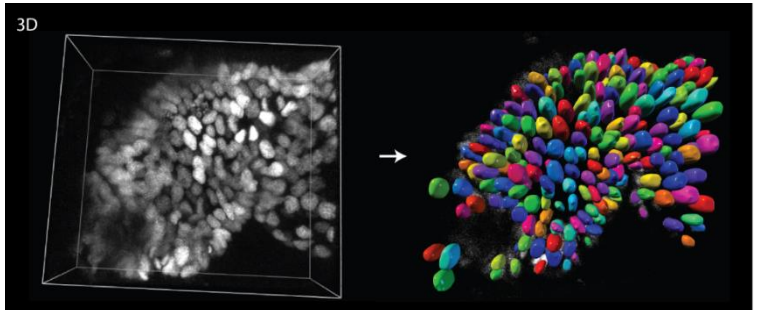
A research field in computer science

Finds more and more applications, also in life sciences.

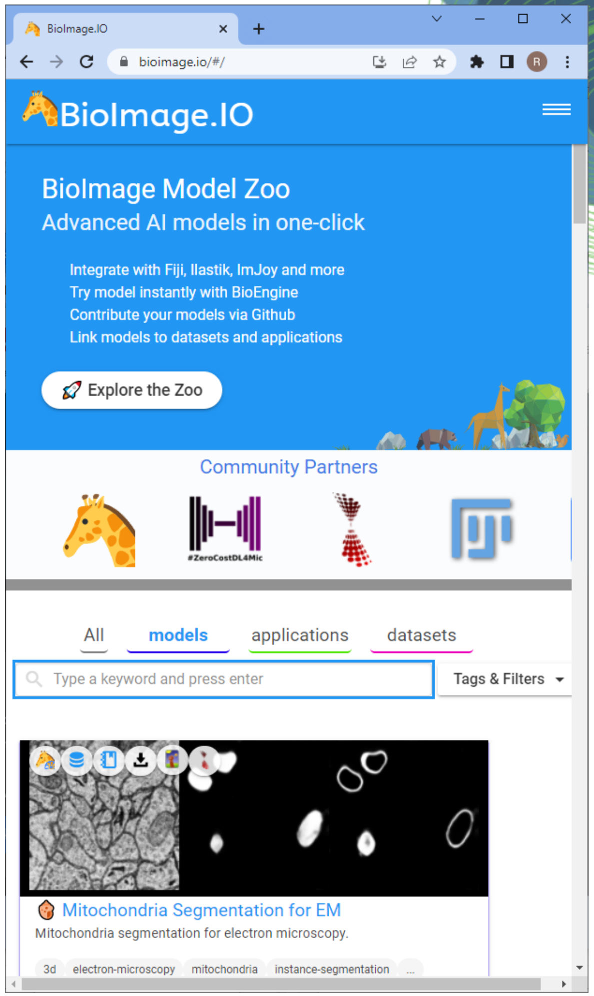




www.cellpose.org/



https://github.com/stardist/stardist



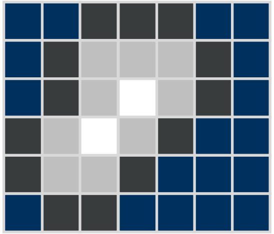
https://bioimage.io/

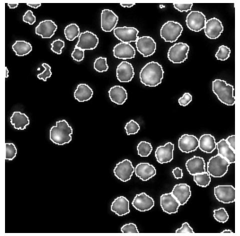
Logos and screenshots are taken from the github repositories / websites provided under BSD and MIT licenses.

Machine learning

Automatic construction of predictive models from given data

Pixels, Objects, Images, Audio, Text, Measurements, …

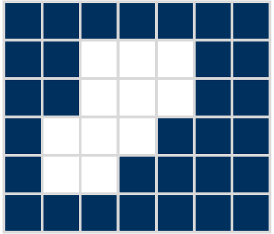


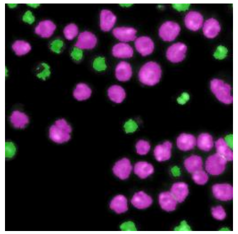


Dense Segmentation / Binarization

Object classification

Image classification

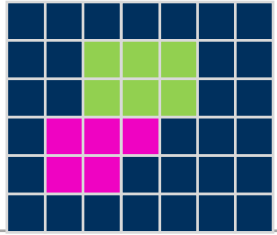




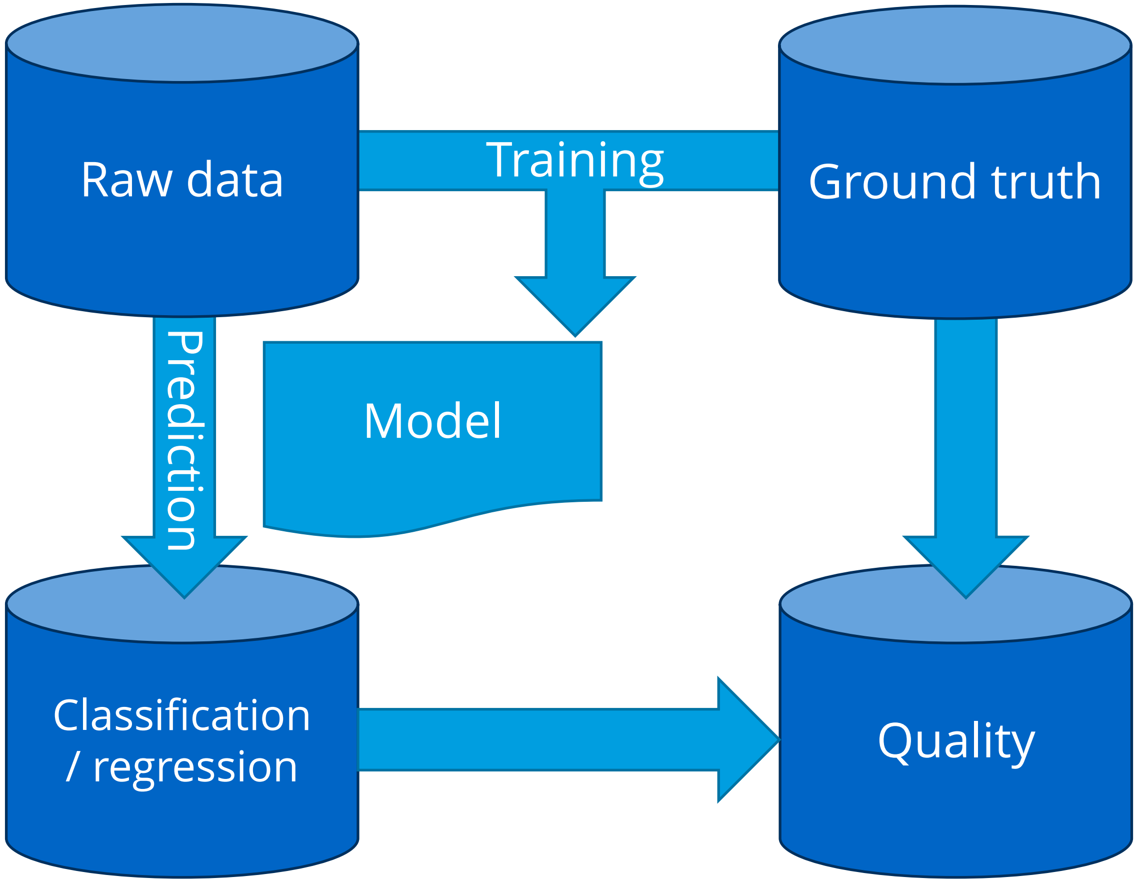
"Cat"

Instance segmentation

Cont. quantity



P_{Cat}= 0.5
$P_{Microscope}$= 0.4
Height = 80 cm

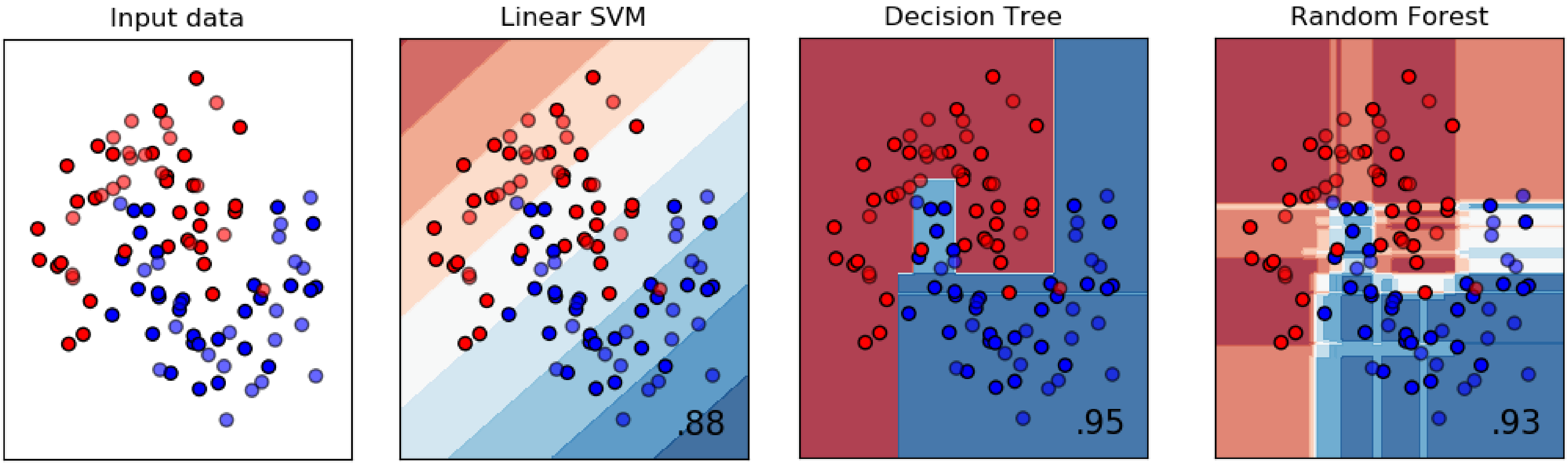


Annotated raw data, usually generated by humans

Precision, Recall

Goal

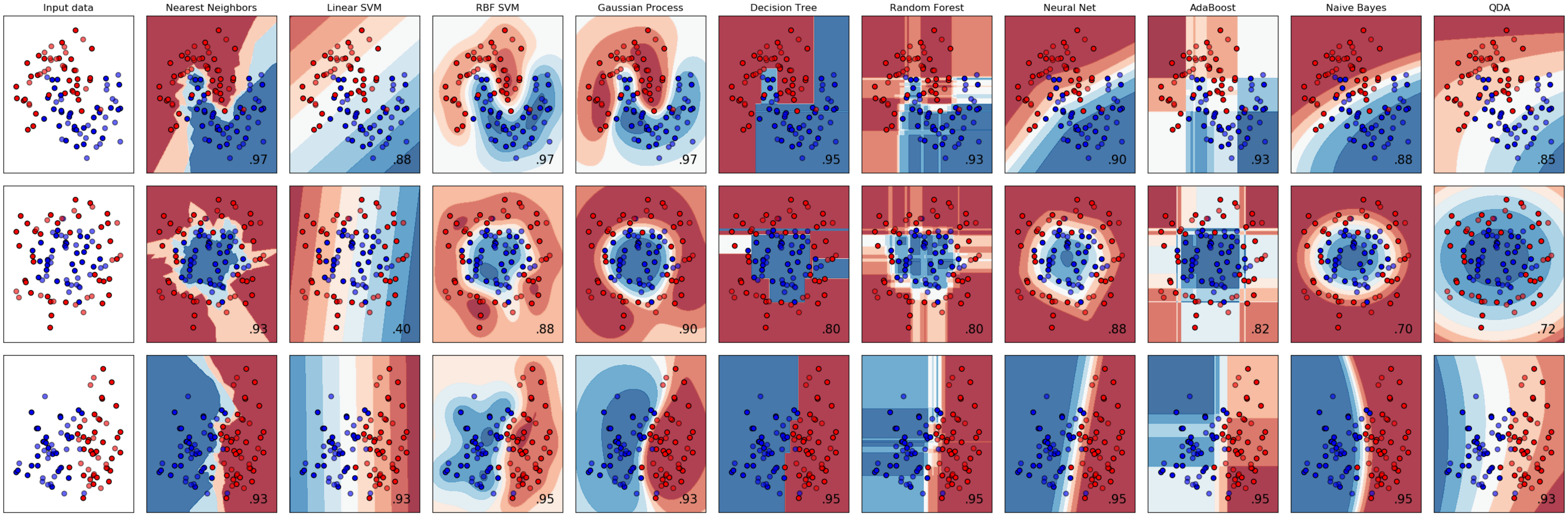
Guess classification (color) from position of a sample in parameter space.



Adapted from https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

Approaches

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



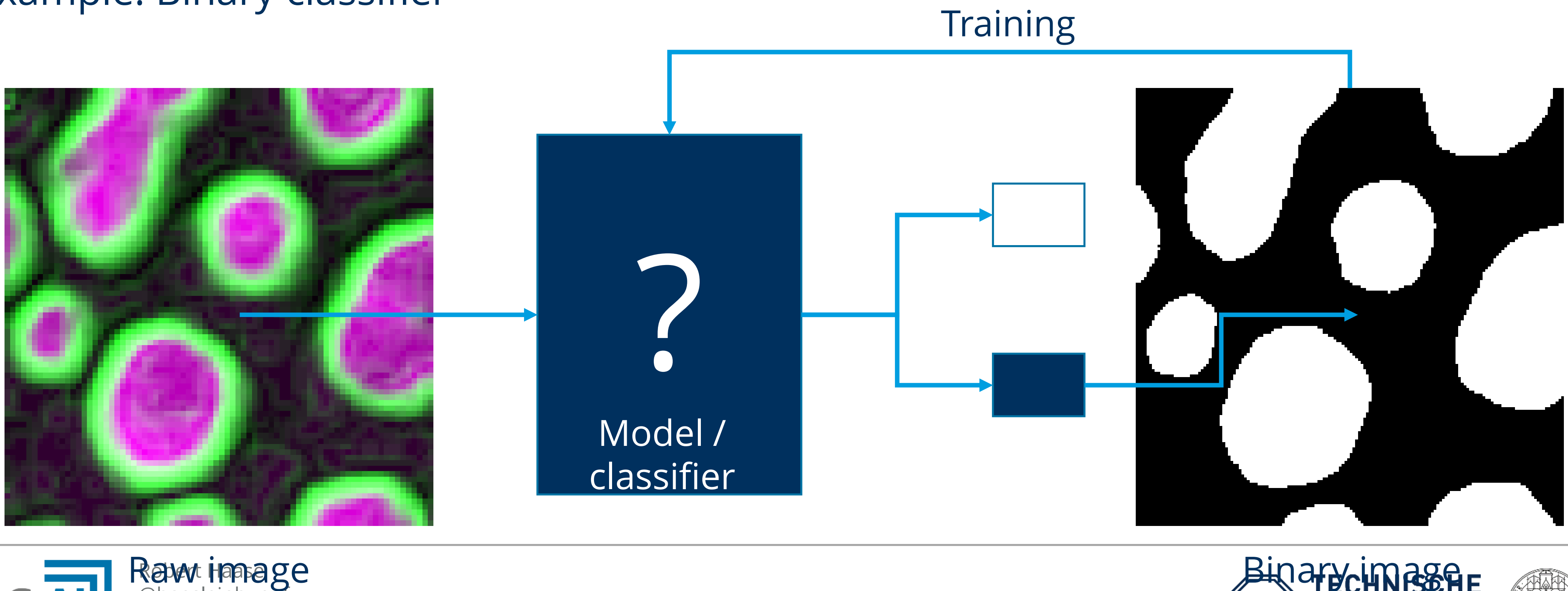
Adapted from https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

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



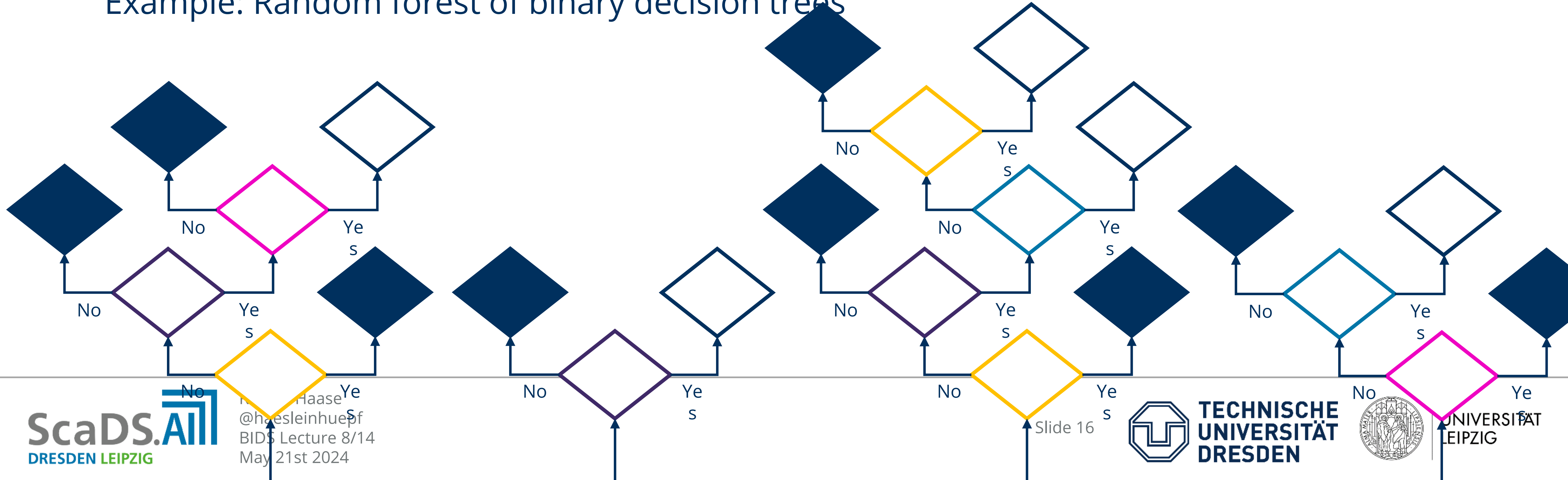
Random forest based image segmentation

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

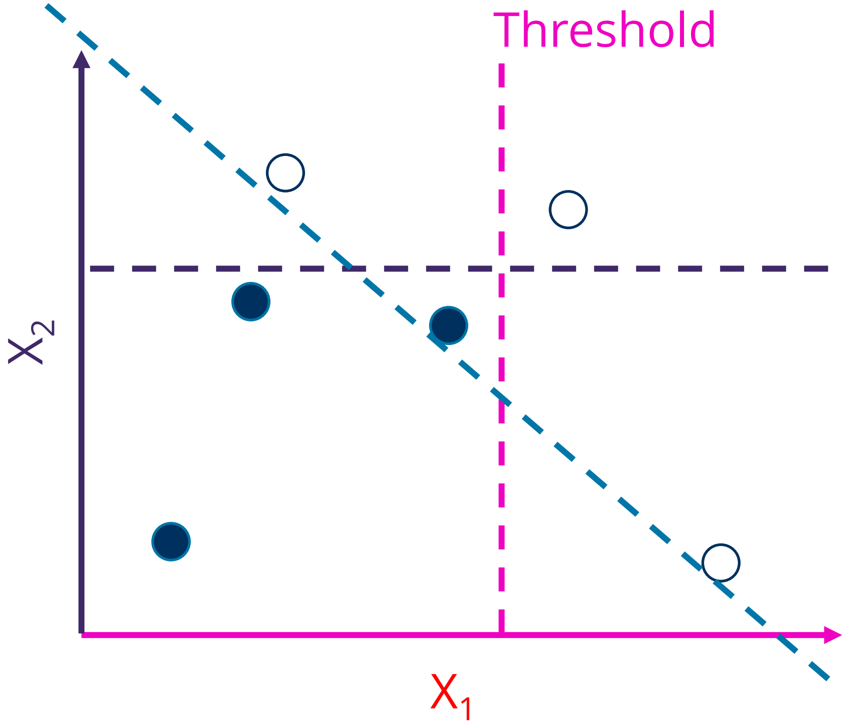
Example: Random forest of binary decision trees

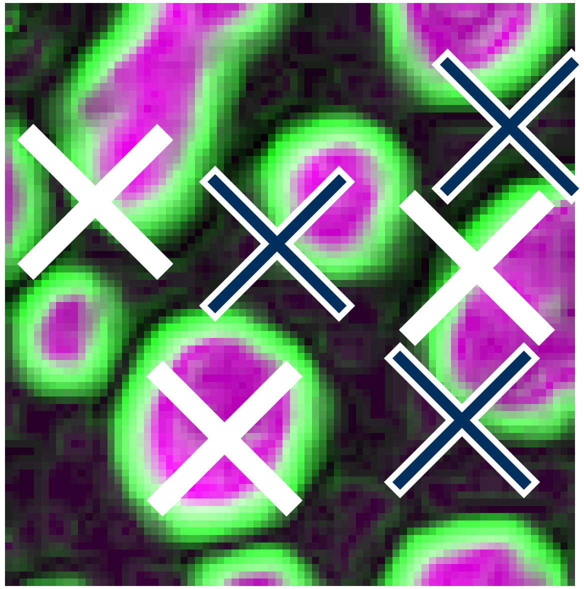


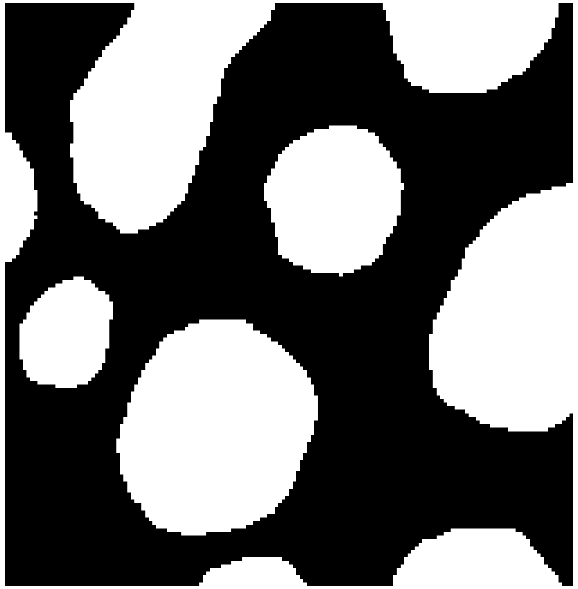
Deriving random decision trees

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

- Individual pixels, their intensity and their classification



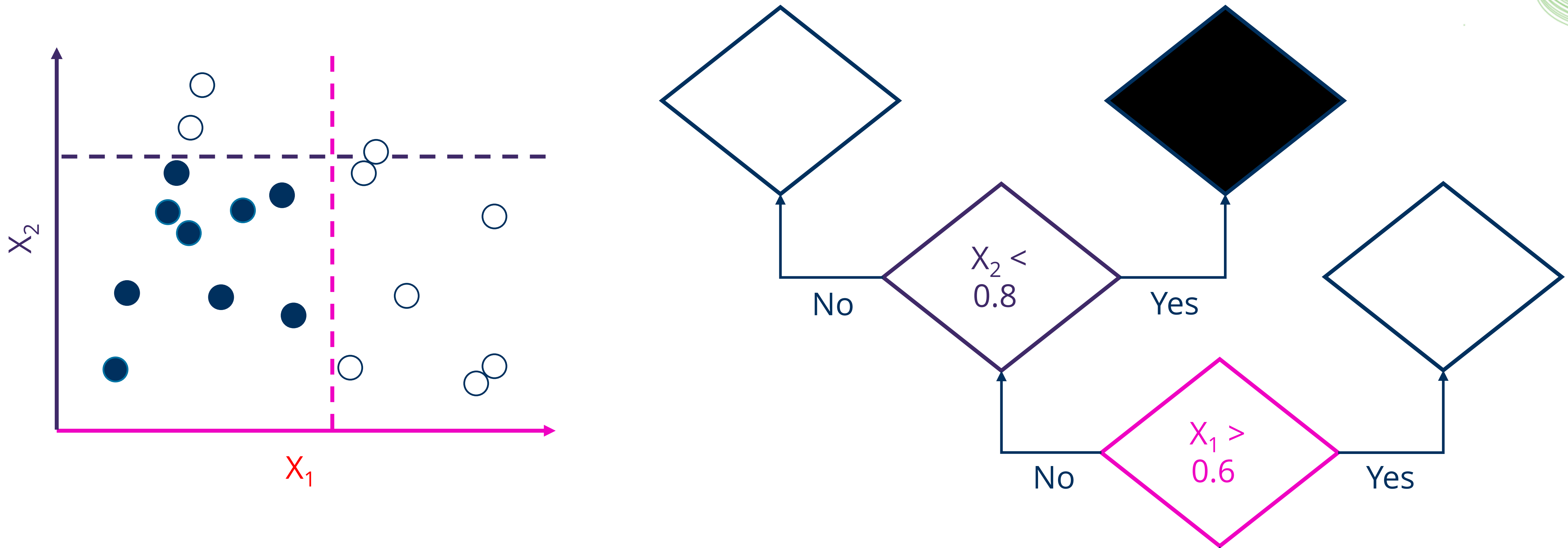




Note: You cannot use a single threshold to make the decision correctly

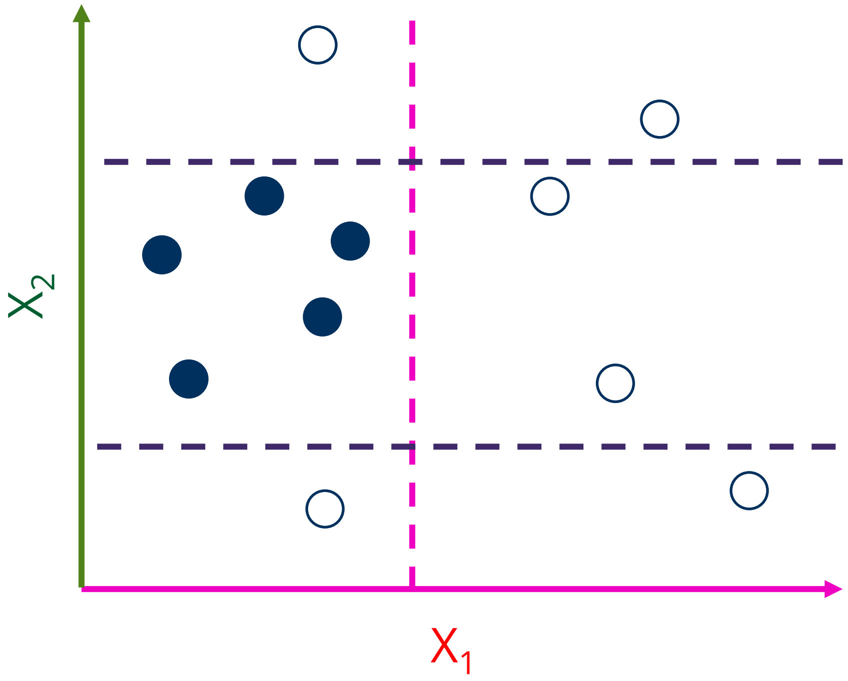
Deriving random decision trees

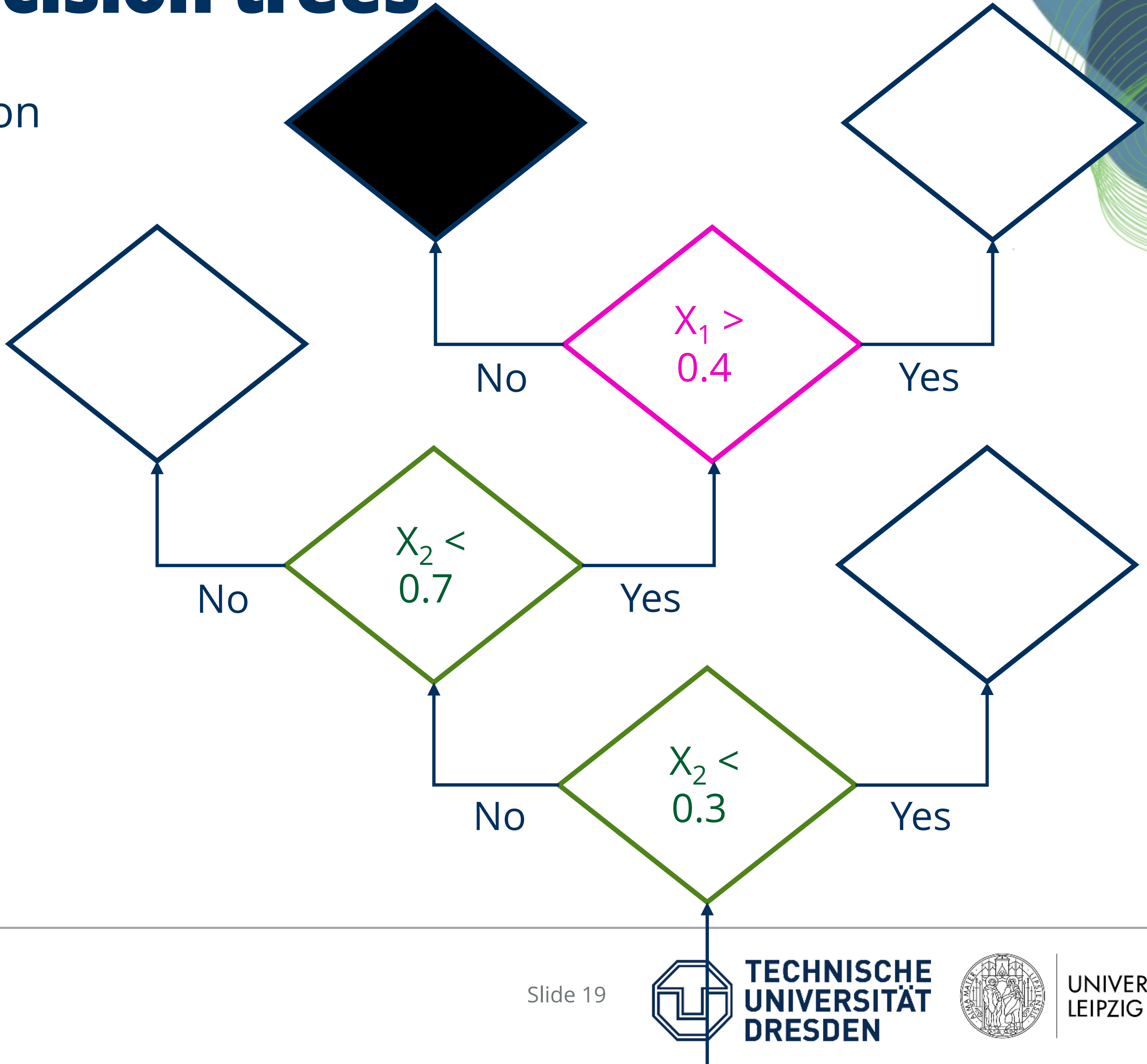
Decision trees combine several thresholds on several parameters



Deriving random decision trees

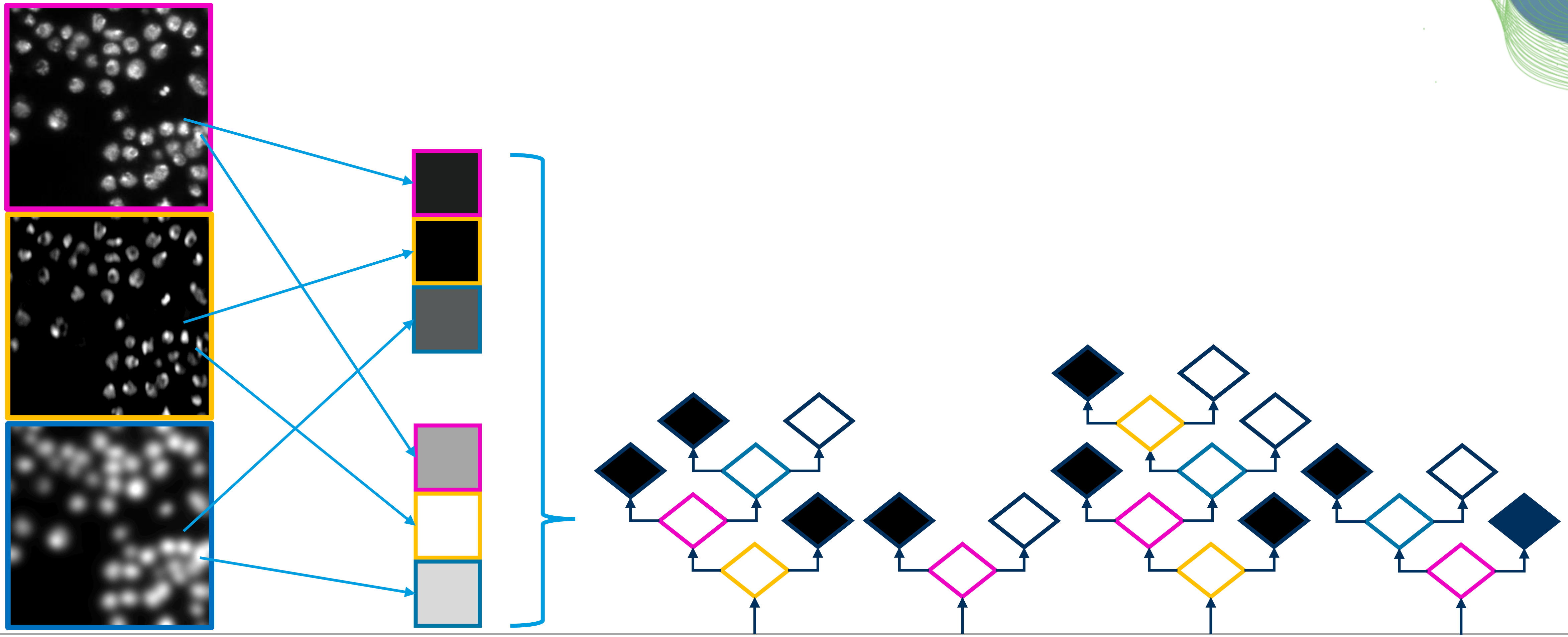
Depending on sampling, the decision trees are different





Random Forest Pixel Classifiers

By training many decision trees, errors are equilibrated



Sampling

Robert Haase
@haesleinhuepf
BIDS Lecture 8/14
May 21st 2024

ScaDS.AI DRESDEN LEIPZIG

TECHNISCHE UNIVERSITÄT DRESDEN

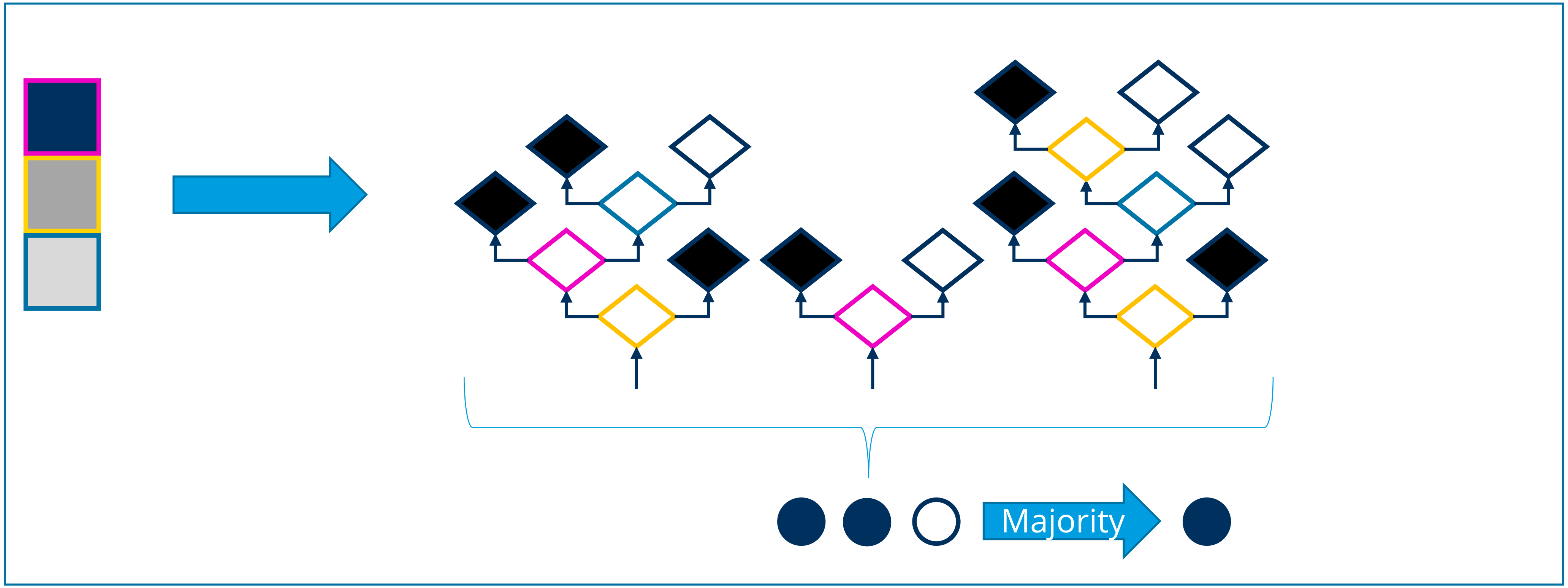
UNIVERSITÄT LEIPZIG

Image data source: BBBC038v1, available from the Broad Bioimage Benchmark Collection (Caicedo et al., Nature Methods, 2019)

Random Forest Pixel Classifiers

Combination of individual tree decisions by voting or max / mean

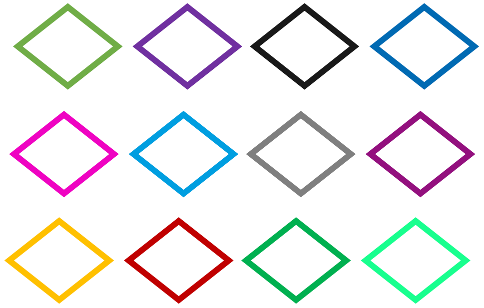
Prediction



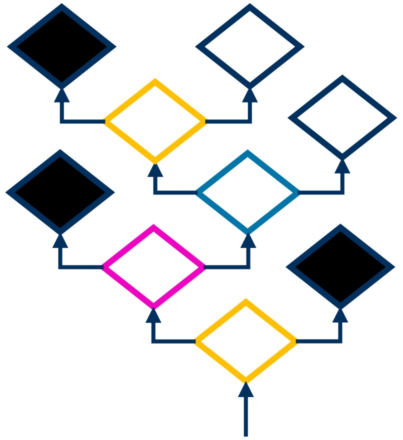
Random Forest Pixel Classifiers

Typical numbers for pixel classifiers in microscopy

Available features:

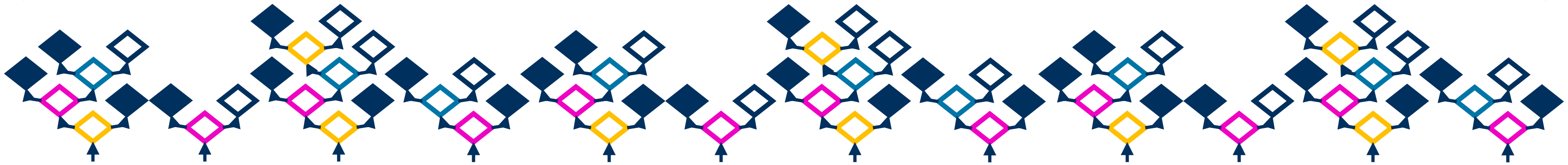


- Gaussian blur image
- DoG image
- LoG image
- Hessian
- ….



Depth: 4

Number of trees: > 100



Model validation

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

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

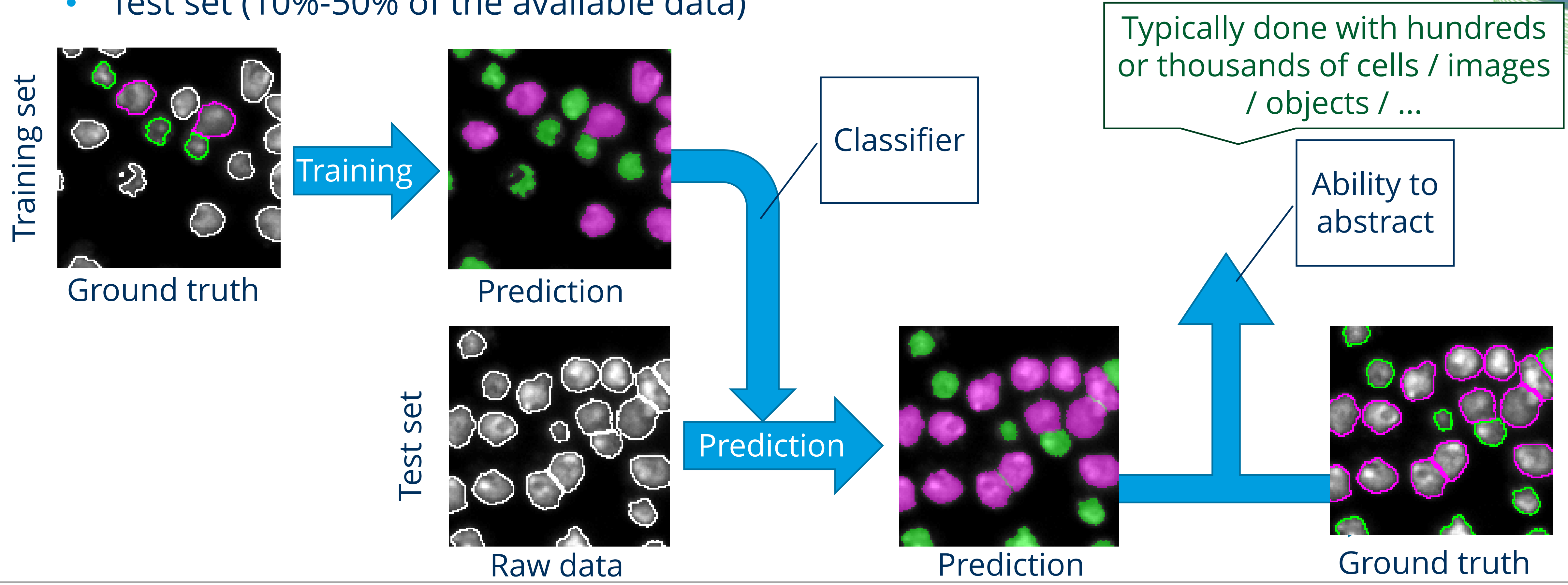
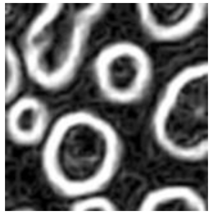


Image data source: BBBC038v1, available from the Broad Bioimage Benchmark Collection (Caicedo et al., Nature Methods, 2019)

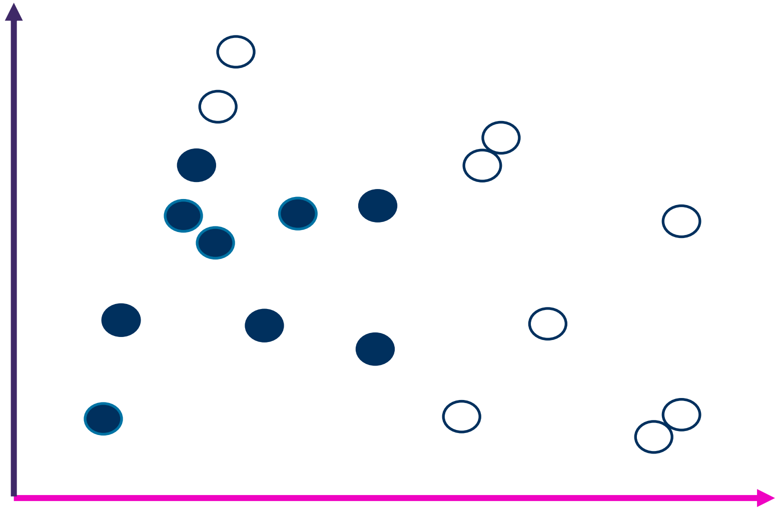
Object classification

What if we exchange pixel features with object features?

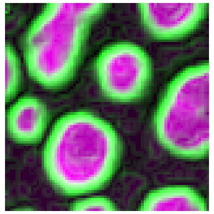


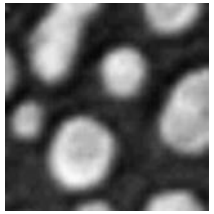
Pixel classification

Intensity in Sobel filtered image



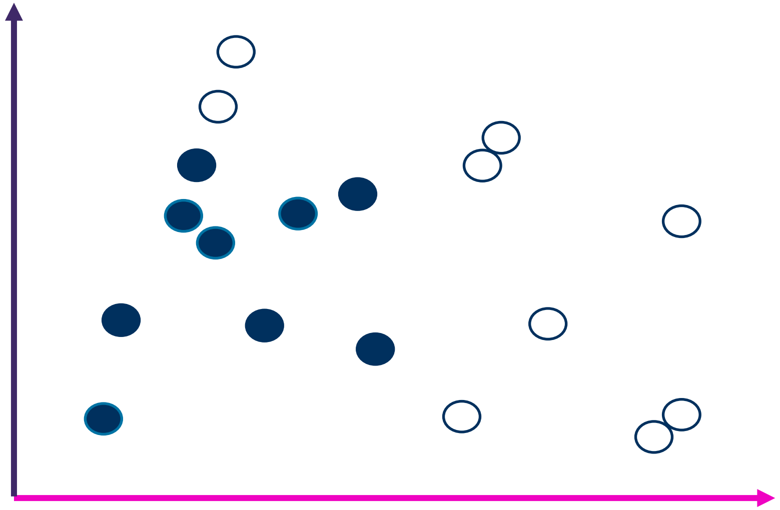
Intensity in raw image





Object classification

Circularity



Aspect ratio

- The algorithms work the same using with different features

ScaDS.AI
DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE

Using Python

Supervised and Unsupervised Machine Learning for Bio-image Analysis

Robert Haase

Funded by

Bundesministerium für Bildung und Forschung

SACHSEN

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.

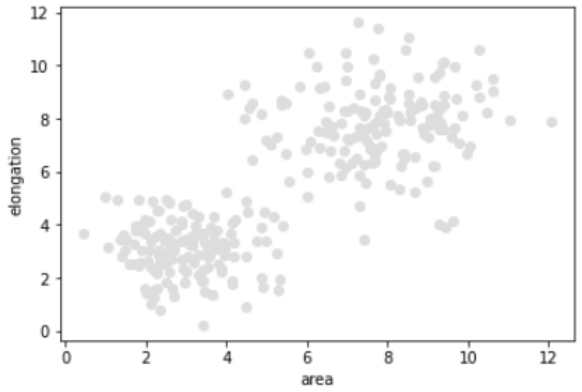
Reusing materials from Johannes Soltwedel, Till Korten, Johannes Müller, Laura Žigutytė (TU Dresden), Ryan Savill (MPI-CBG), Matthias Täschner (ScaDS.AI/Uni Leipzig) and the Scikit-learn community.

Tabular object classification

Classify objects starting from feature vectors (table columns)

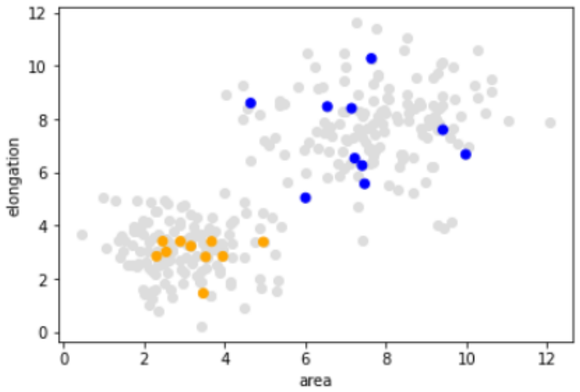
Raw data

	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,
```

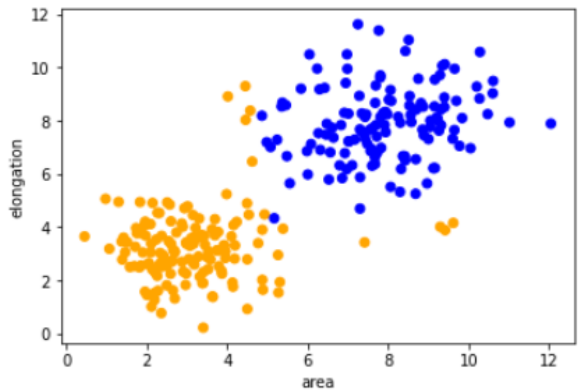


Classifier training

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

Classifier prediction

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



https://github.com/BiAPoL/Bio-

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)
```

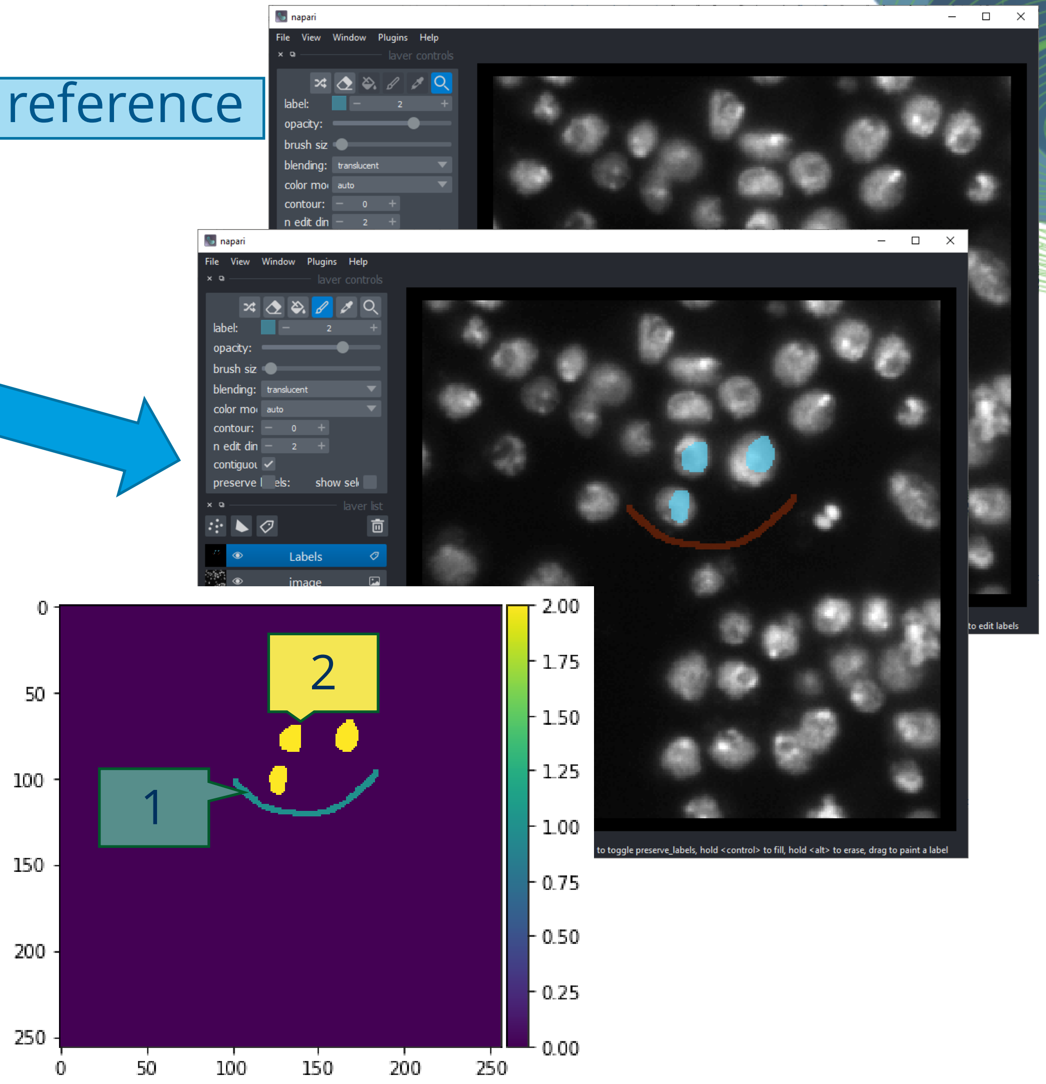
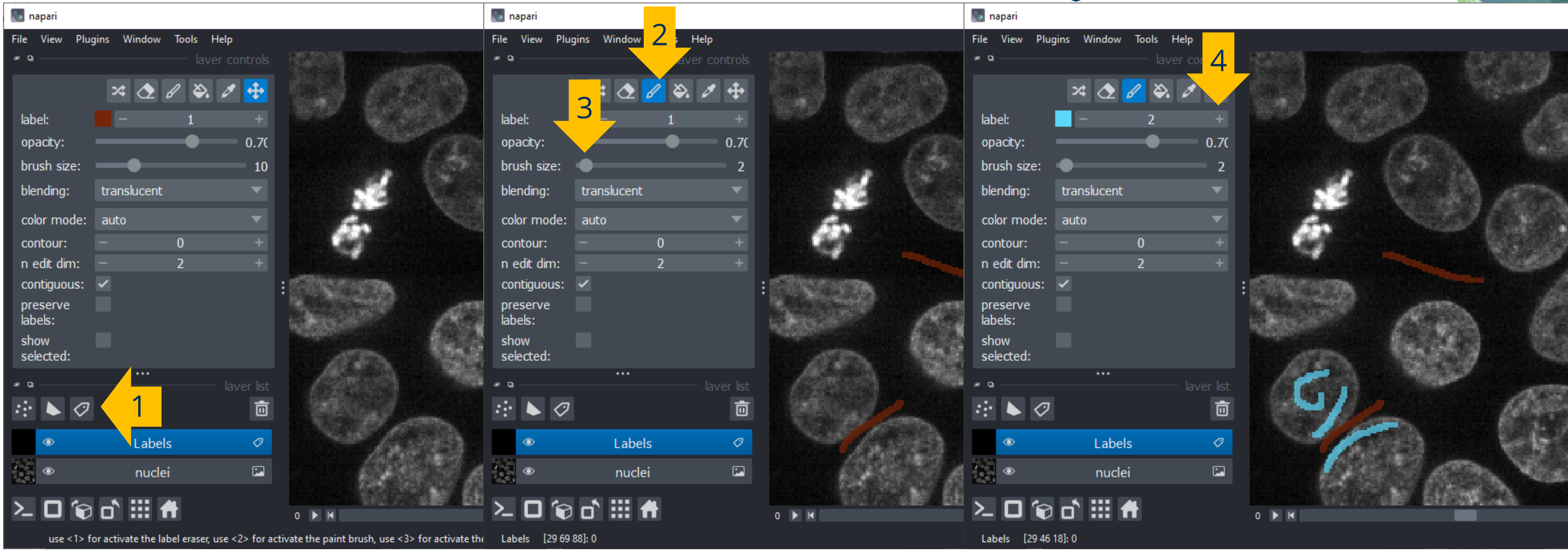


Image data source: BBBC038v1, available from the Broad Bioimage Benchmark Collection (Caicedo et al., Nature Methods, 2019).

Napari – common workflows

Pixel / object annotation drawing

[1: Create empty labels layer]
2: Select paint brush tool
3: Decreaese 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)
```

Image data

Ground truth / annotation

```
y_ = classifier.predict(X)
```

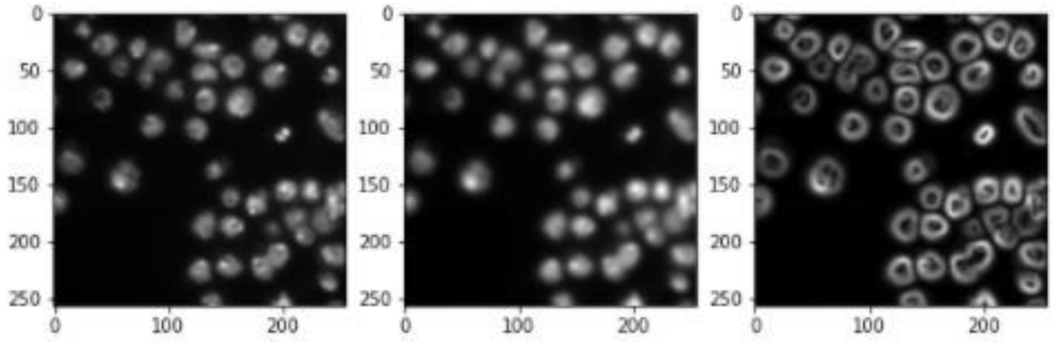
Image data

prediction

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)
```

Image data source: BBBC038v1, available from the Broad Bioimage Benchmark Collection (Caicedo et al., Nature Methods, 2019)

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

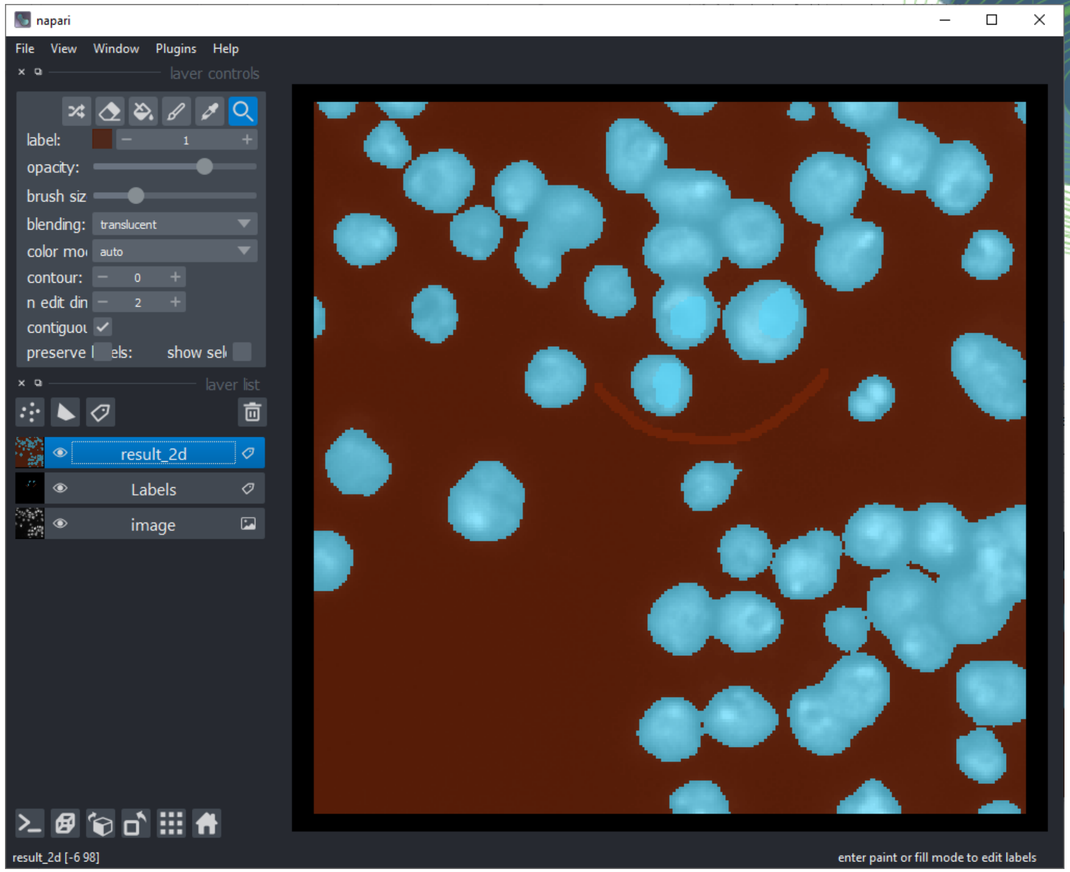
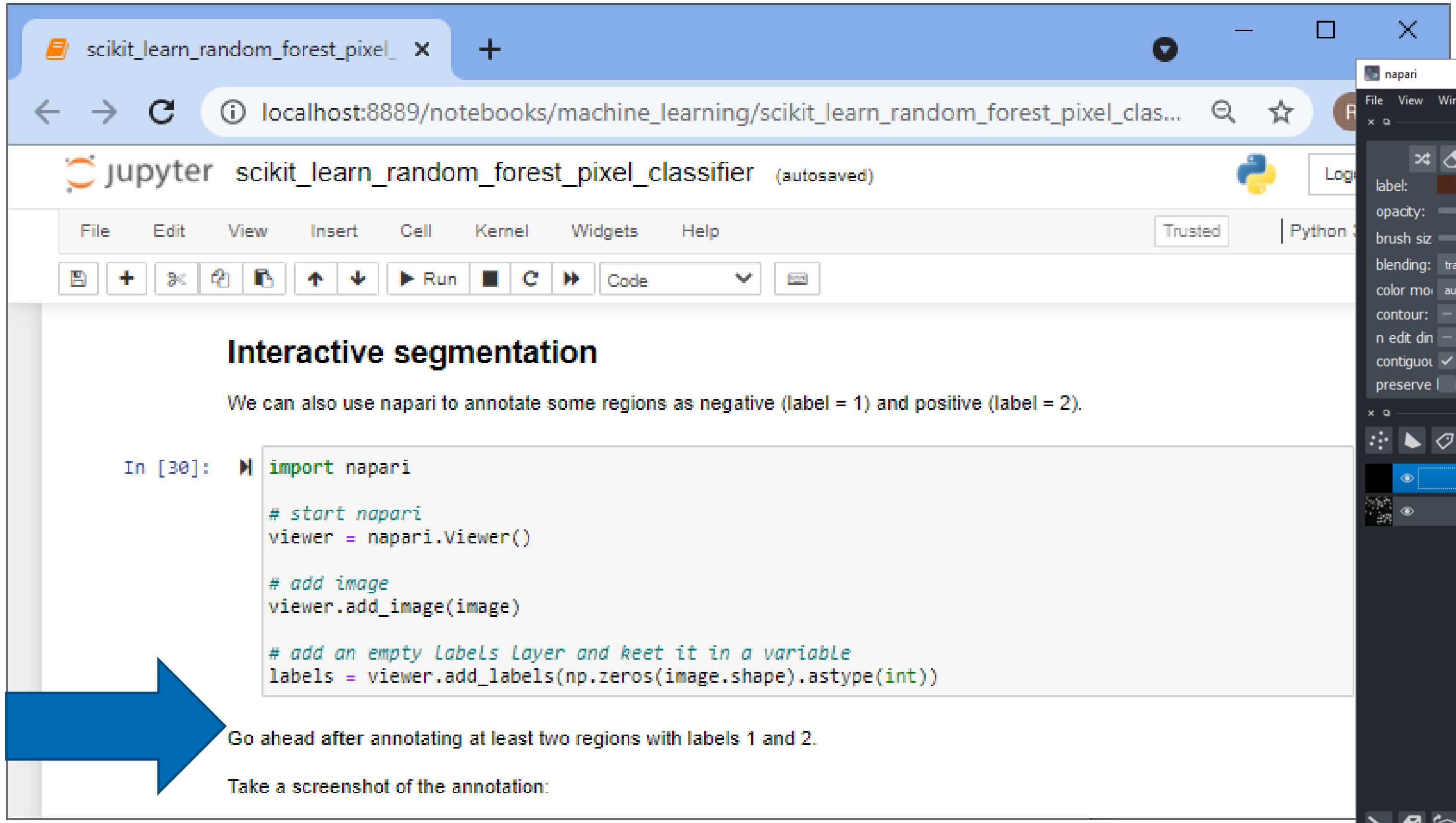


Image data source: BBBC038v1, available from the Broad Bioimage Benchmark Collection (Caicedo et al., Nature Methods, 2019)

Interactive pixel classification

Jupyter notebooks and napari side-by-side



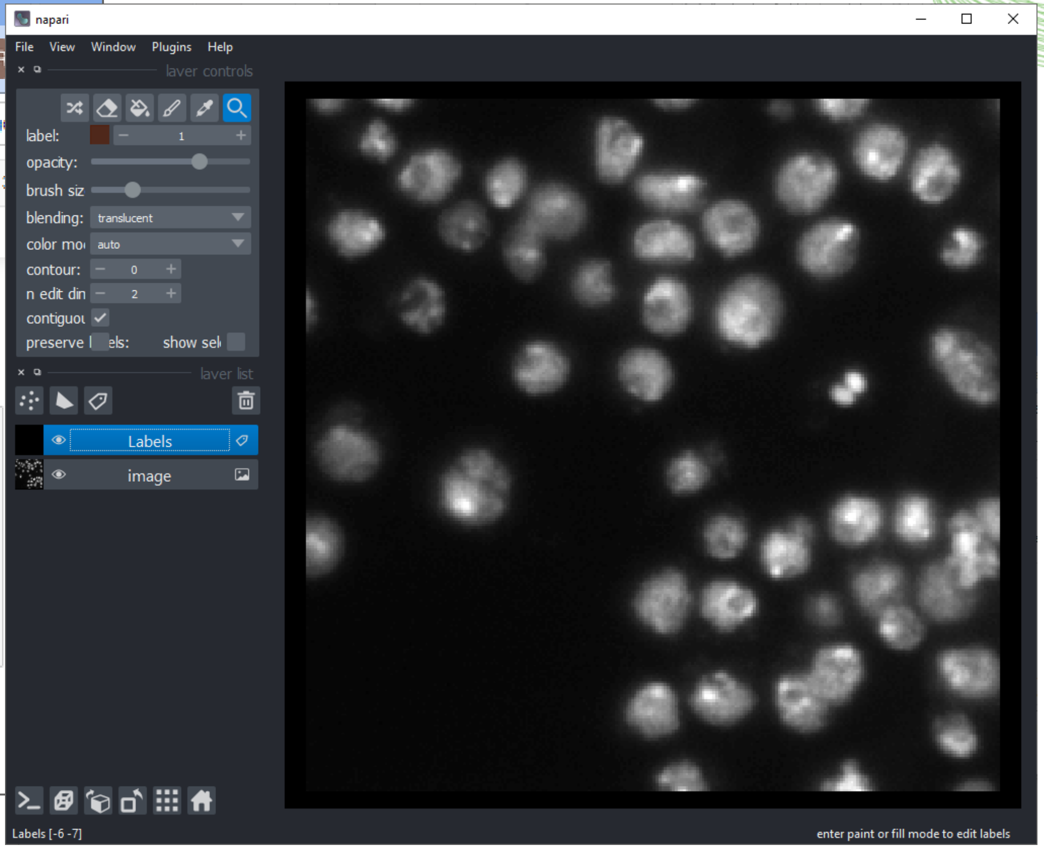


Image data source: BBBC038v1, available from the Broad Bioimage Benchmark Collection (Caicedo et al, Nature Methods, 2019)

Interactive pixel classification

Jupyter notebooks and napari side-by-side

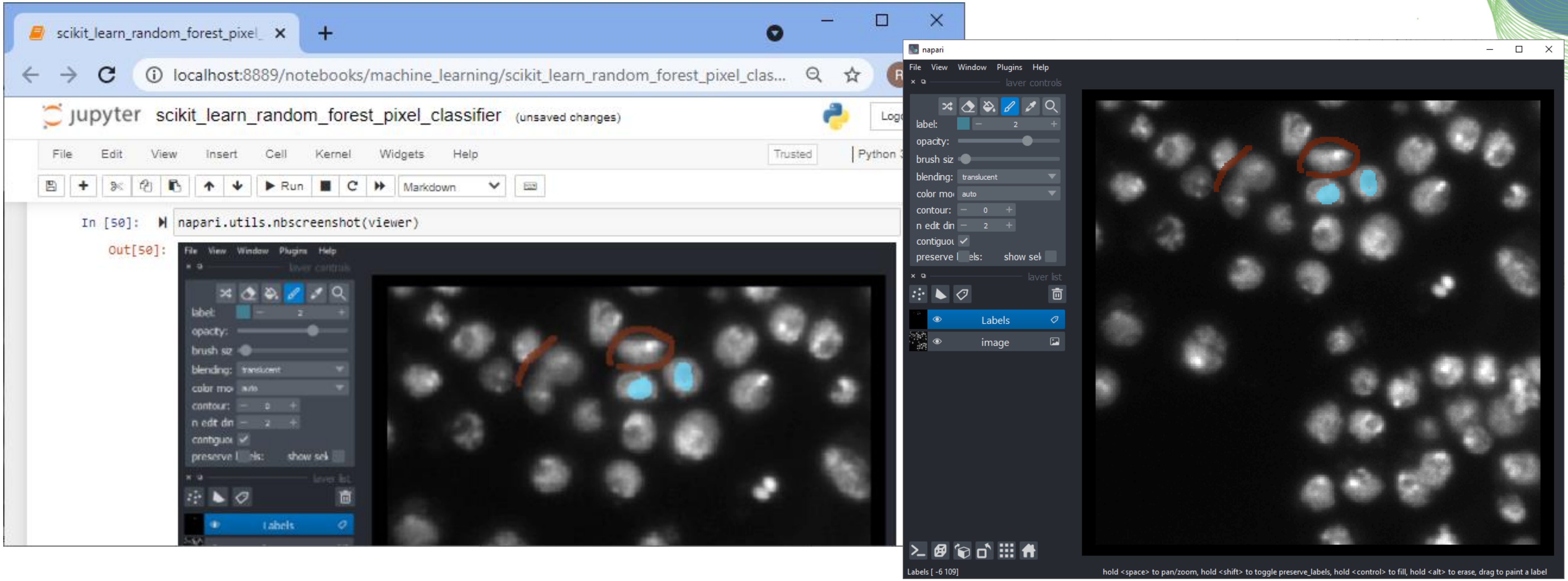


Image data source: BBBC038v1, available from the Broad Bioimage Benchmark Collection (Caicedo et al. Nature Methods, 2019)

Interactive pixel classification

Jupyter notebooks and napari side-by-side

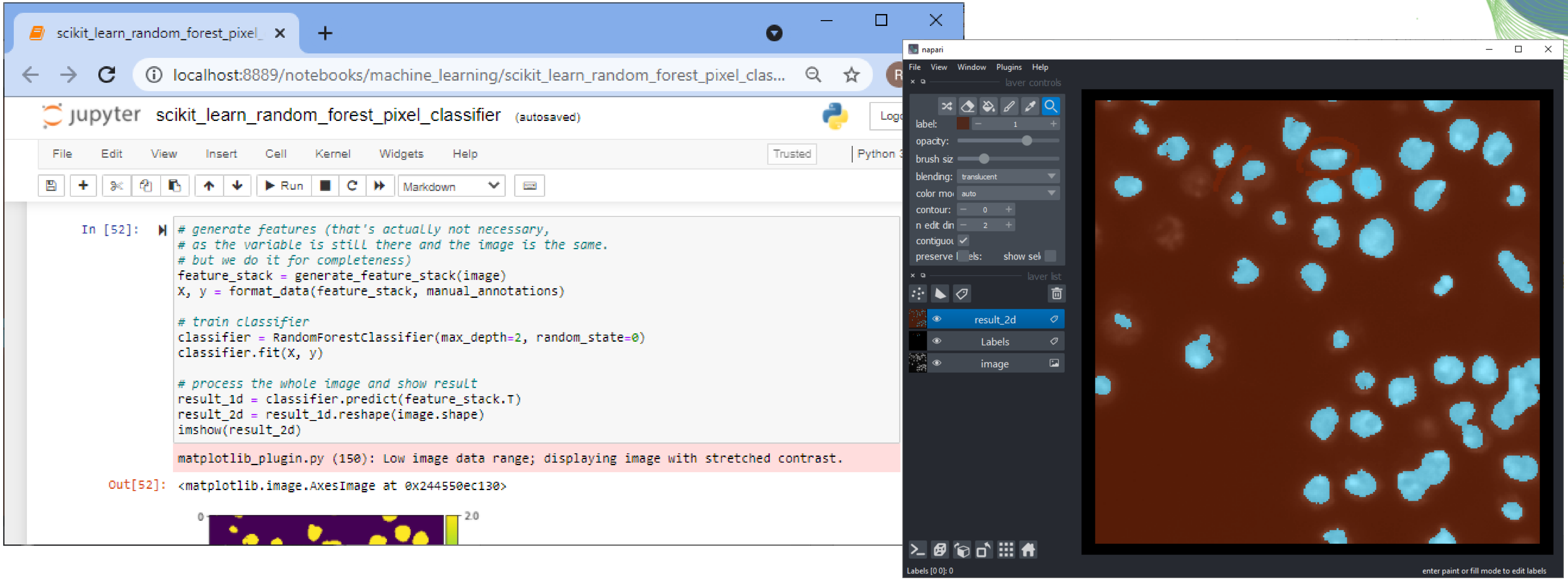


Image data source: BBBC038v1, available from the Broad Bioimage Benchmark Collection (Caicedo et al., Nature Methods, 2019)

CENTER FOR SCALABLE DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE

GPU-accelerated

Using Python

Supervised and Unsupervised Machine Learning for Bio-image Analysis

Robert Haase

Funded by

Bundesministerium für Bildung und Forschung

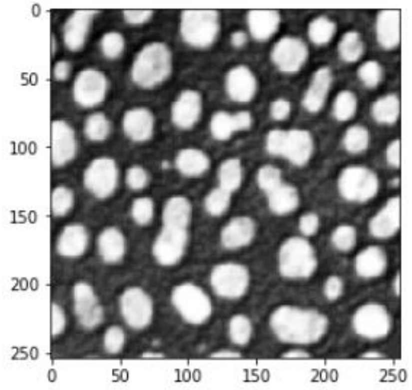
SACHSEN

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.

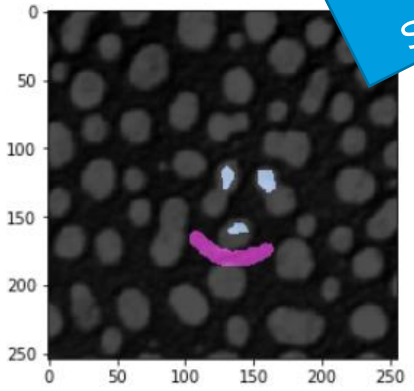
Reusing materials from Johannes Soltwedel, Till Korten, Johannes Müller, Laura Žigutyte (TU Dresden), Ryan Savill (MPI-CBG), Matthias Täschner (ScaDS.AI/Uni Leipzig) and the Scikit-learn community.

Accelerated pixel and object classification

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

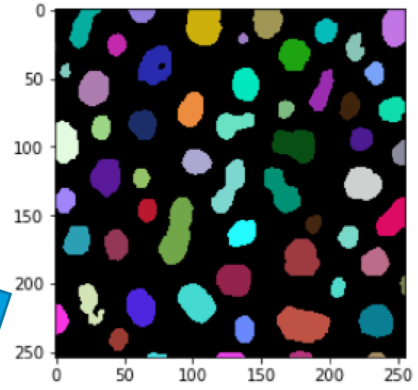


Raw image

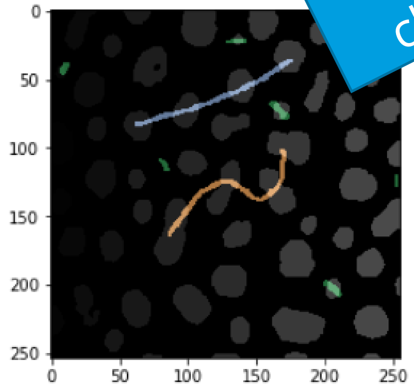


Pixel annotation

Object segmentation

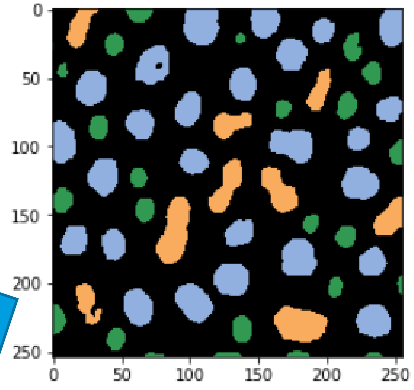


Object label image



Object annotation

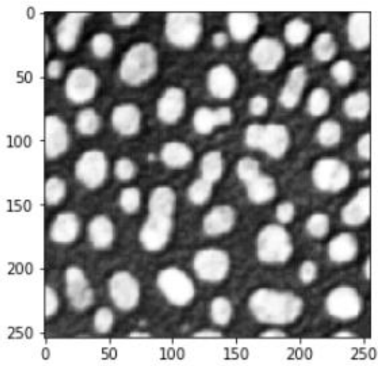
Object classification



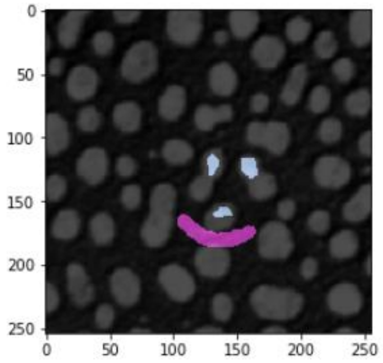
Class label image

Object segmentation

Pixel classification + connected component labeling



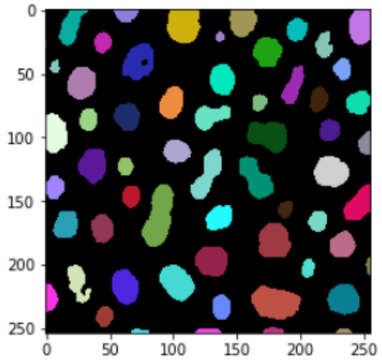
Raw image



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

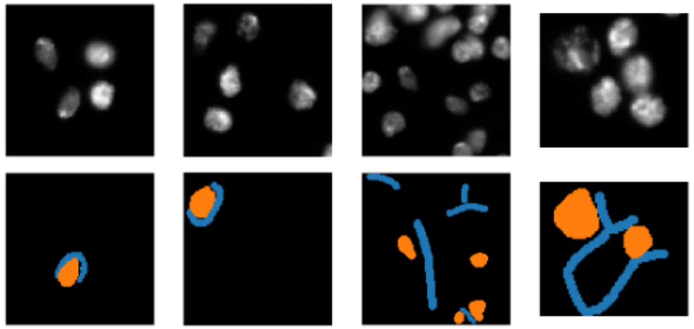


Object label image

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 classifer 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)
```

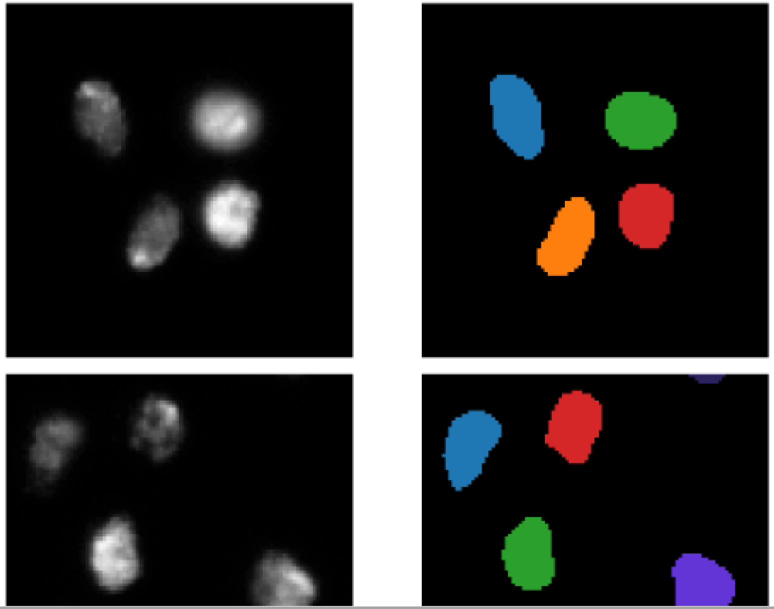
- data
  - BBBC007
    - images
      - A9 p5d (cropped 1).tif
      - A9 p5d (cropped 2).tif
      - A9 p5d (cropped 3).tif
      - A9 p5d (cropped 4).tif
    - masks
      - A9 p5d (cropped 1).tif
      - A9 p5d (cropped 2).tif
      - A9 p5d (cropped 3).tif
      - A9 p5d (cropped 4).tif

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.

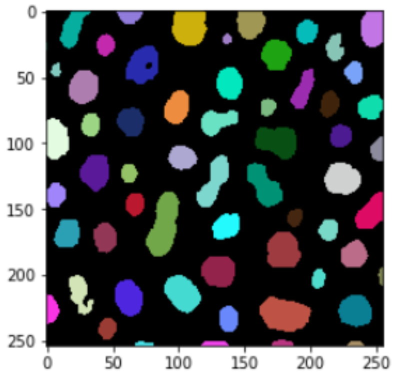
```
[5]: segmenter = apoc.ObjectSegmenter(opencl_filename="data/object_segmenter_trained_on_f
```

```
[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)
```

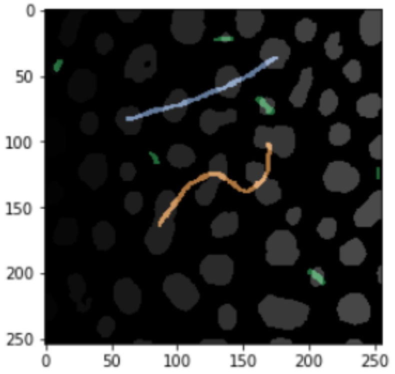


Object classification

Feature extraction + tabular classification



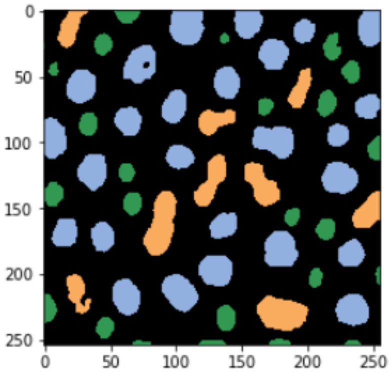
Object label image



Object annotation

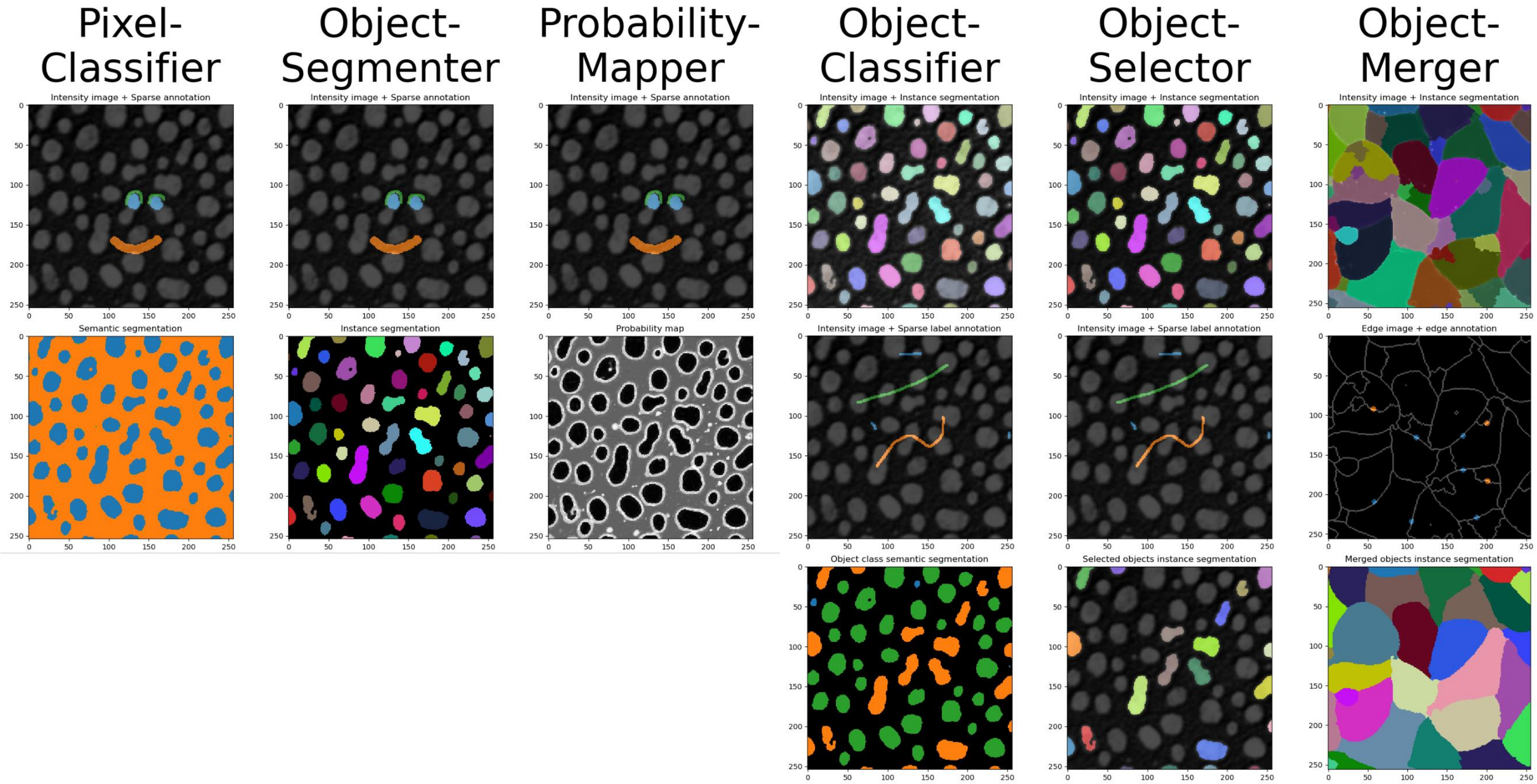
```
# for the classification we define size and shape as criteria
features = 'area mean_max_distance_to_centroid_ratio'

# 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



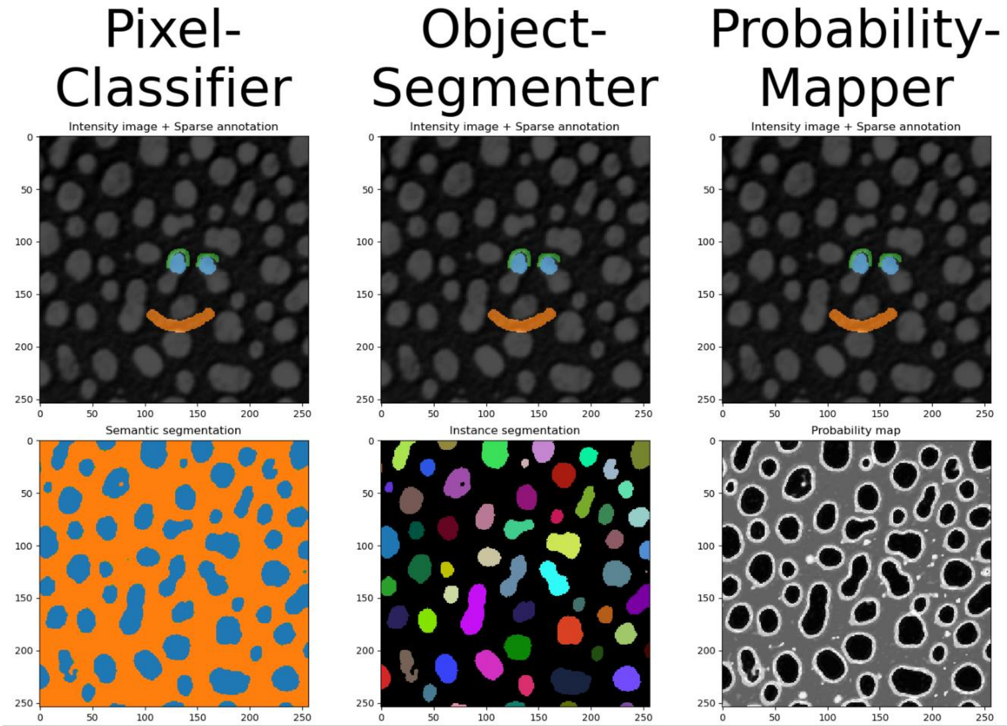
Class label image

Other classification / regression tasks



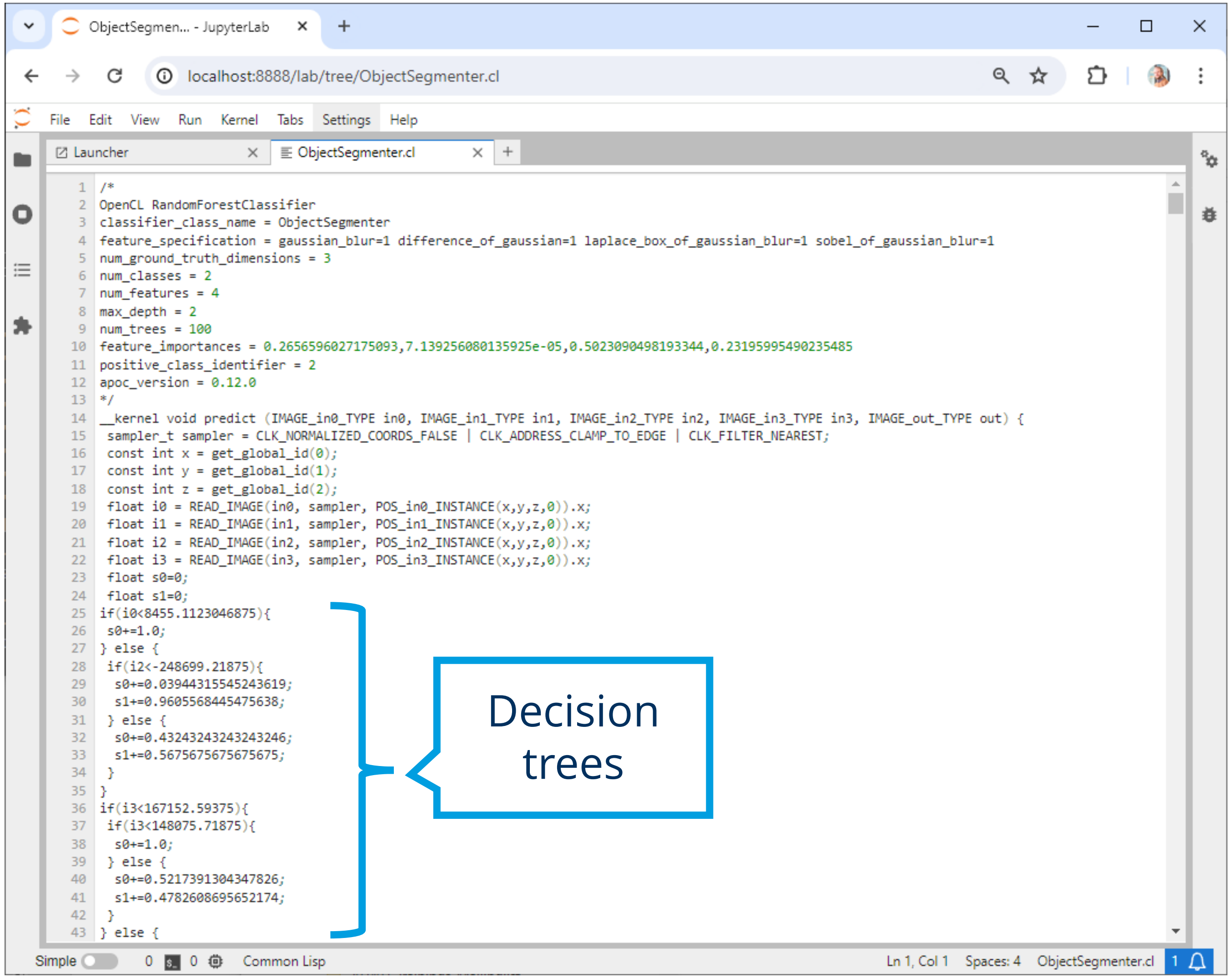
Quiz: Classification versus Regression

Which of these three solves a regression task?



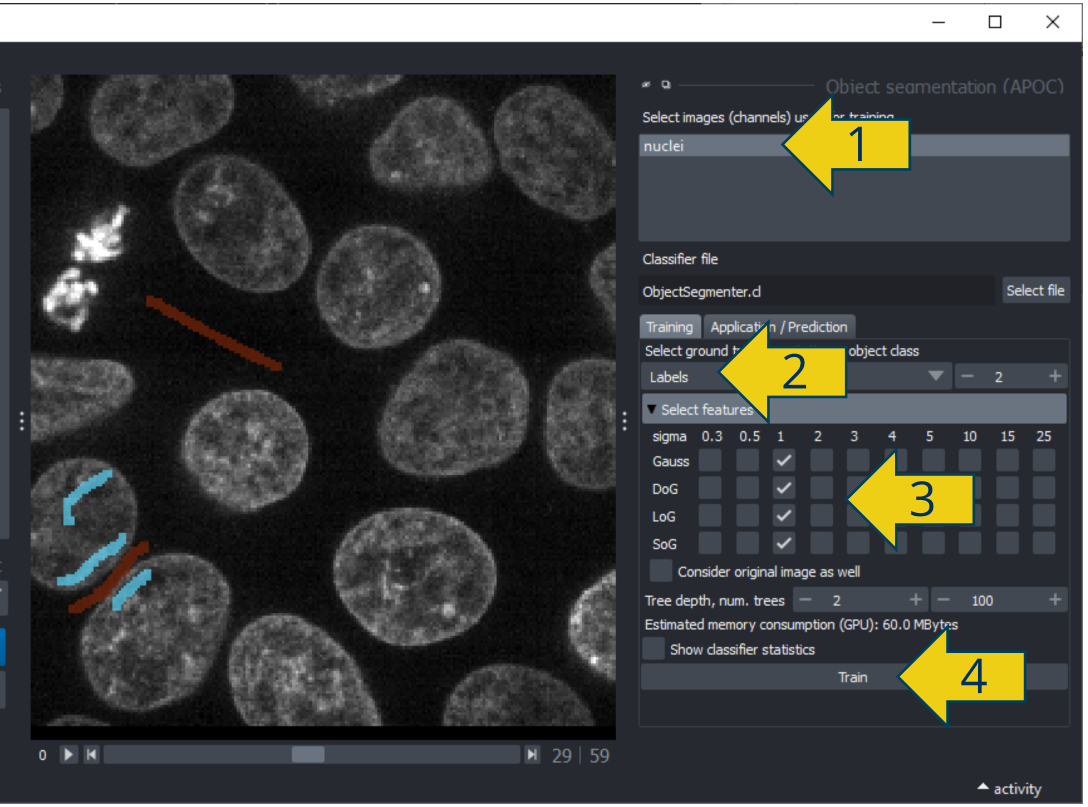
Under the hood: clesperanto / OpenCL

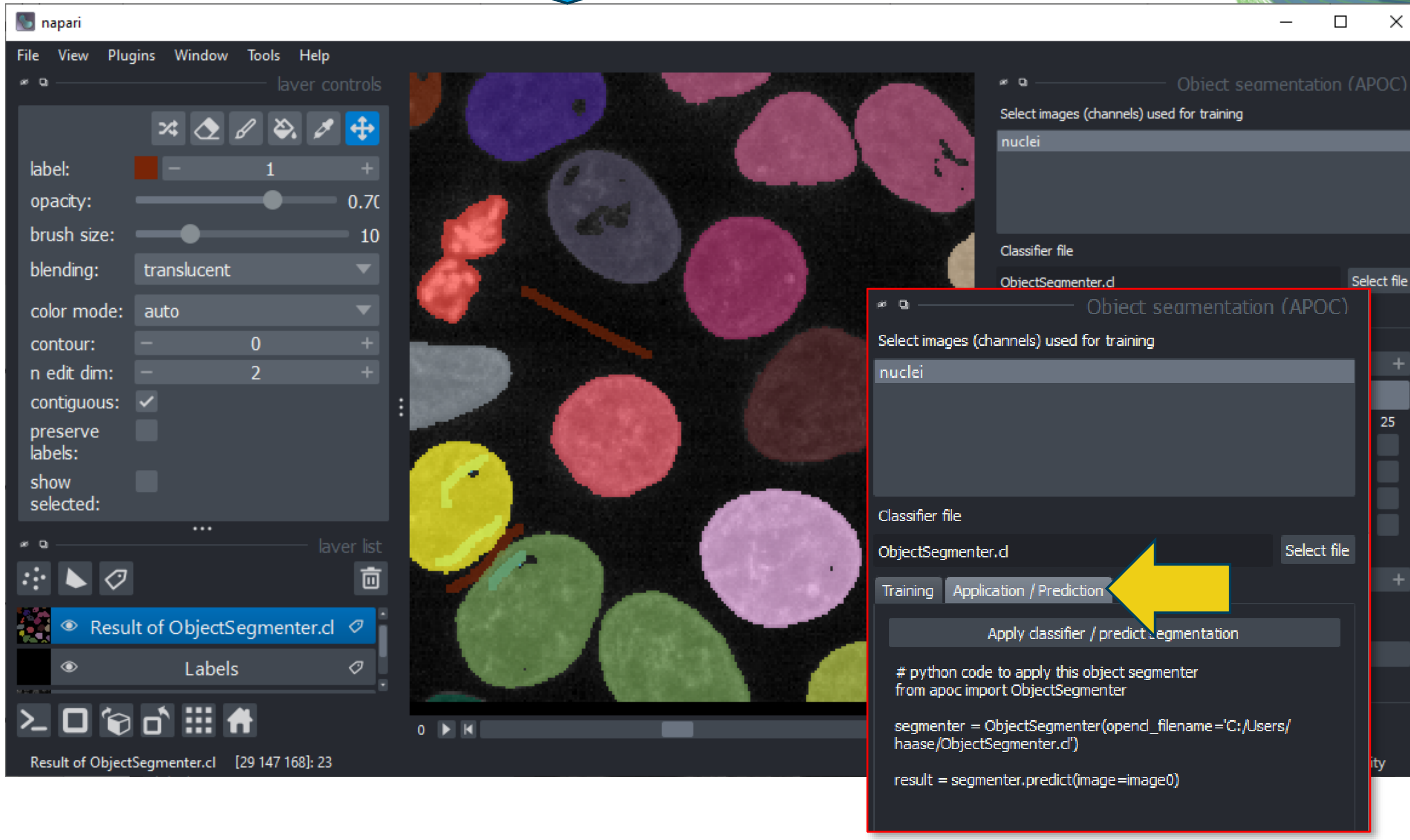
classifier.cl files
can be *read*



Graphical user interface: Object segmentation

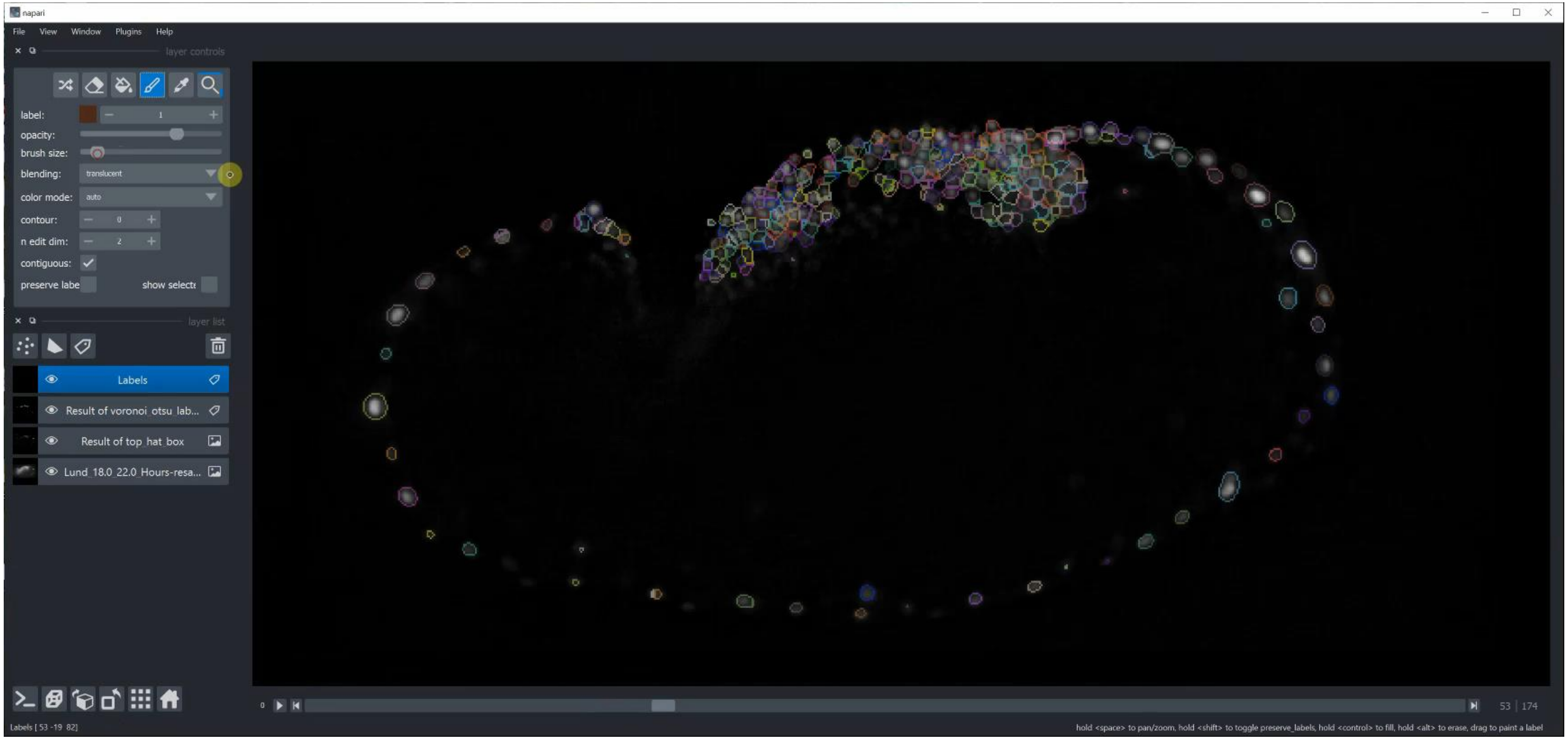
1: Select image[s]
2: Select ground truth annotation
[3: Select features]
4: Train / predict





Graphical user interface: Object classification

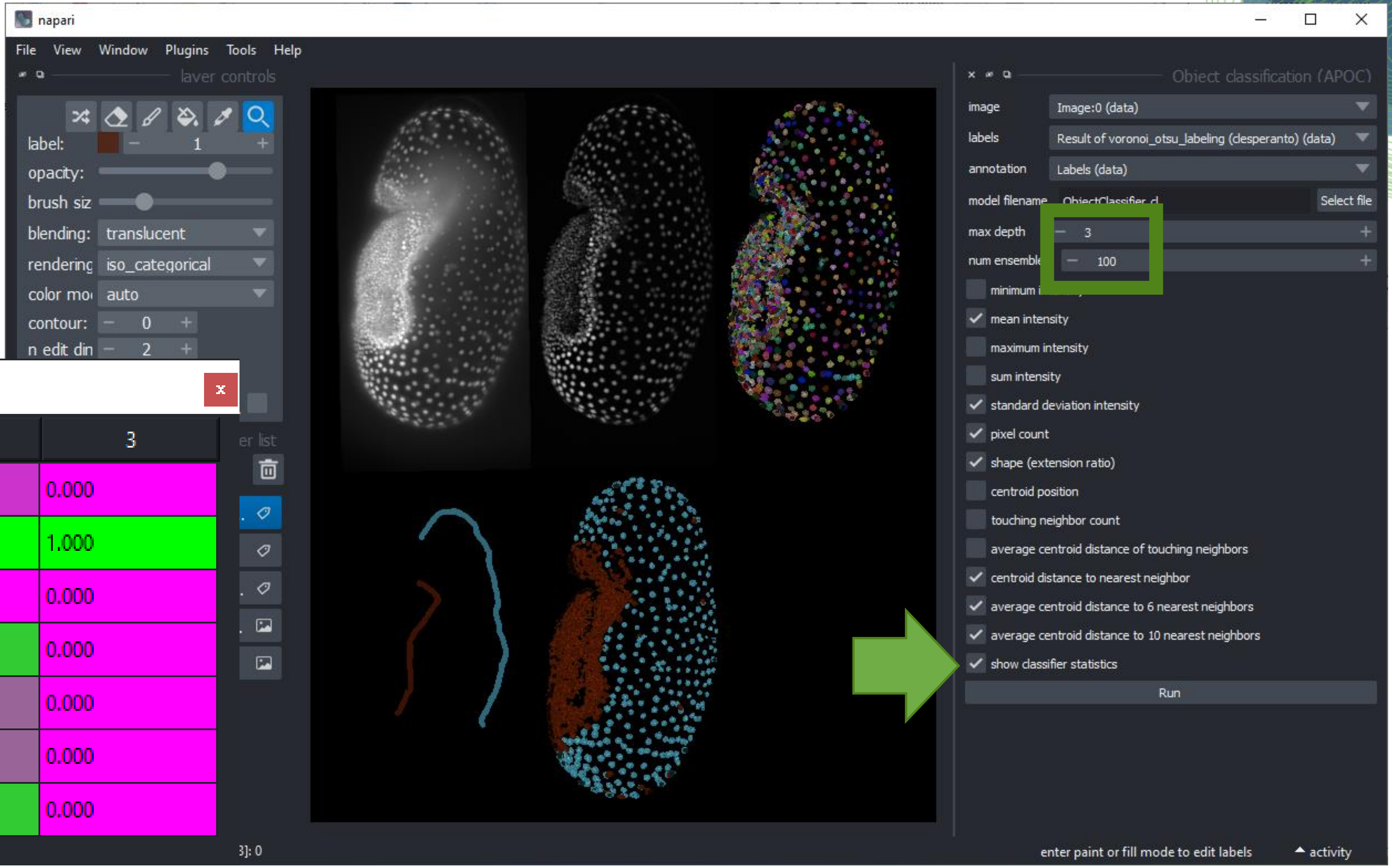
Annotation / classification of segmented objects



Graphical user interface: Object classification

Inspect how the random forest classifier makes decisions

Note: Beware of correlated parameters!



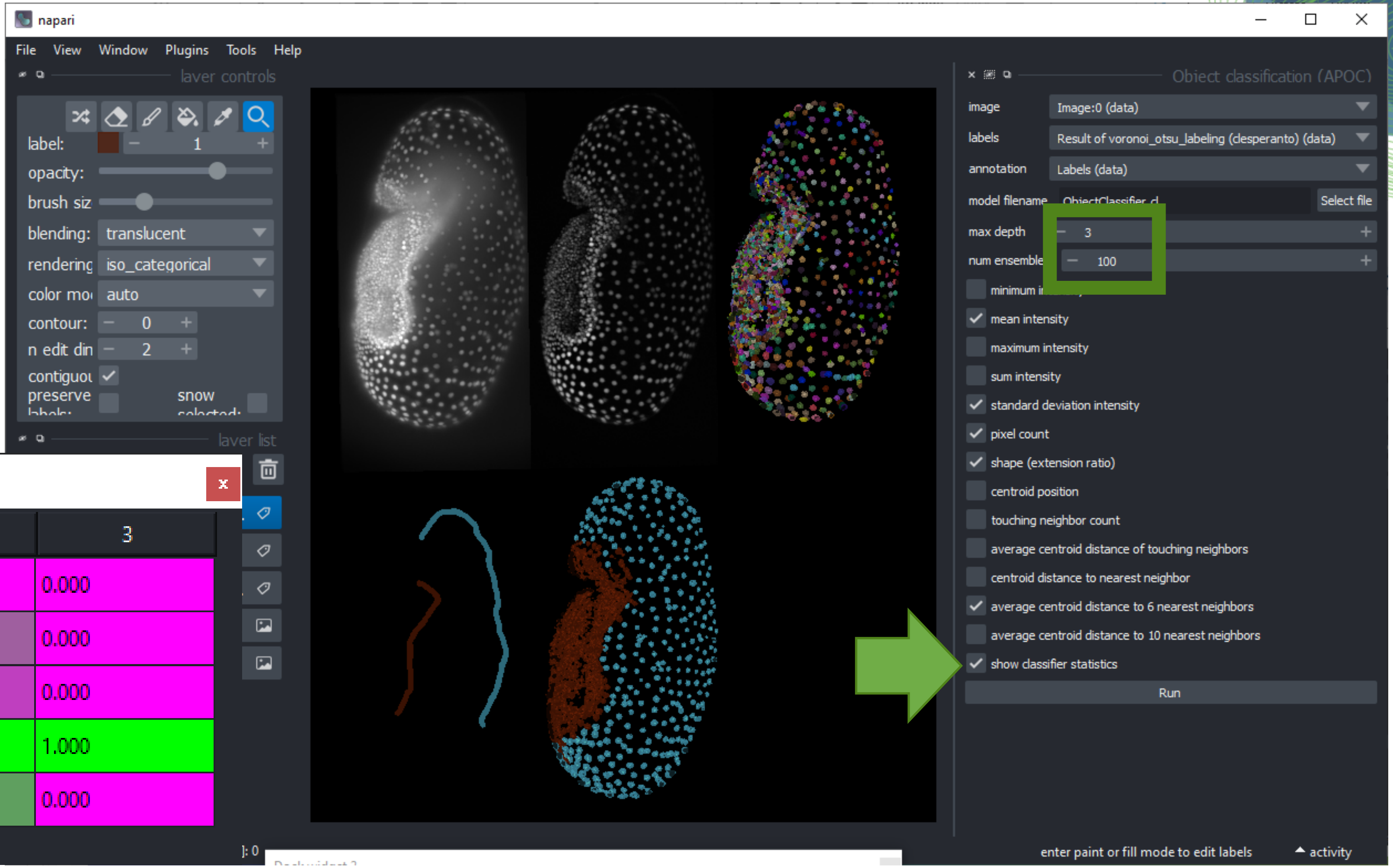
Dock widget 1

	1	2	3
area	0.010	0.056	0.000
mean_intensity	0.200	0.278	1.000
standard_deviation_intensity	0.030	0.000	0.000
mean_max_distance_to_centroid_ratio	0.270	0.222	0.000
average_distance_of_n_nearest_neighbors=1	0.120	0.111	0.000
average_distance_of_n_nearest_neighbors=6	0.170	0.111	0.000
average_distance_of_n_nearest_neighbors=10	0.200	0.222	0.000

Graphical user interface: Object classification

Inspect how the random forest classifier makes decisions

Note: Beware of correlated parameters!



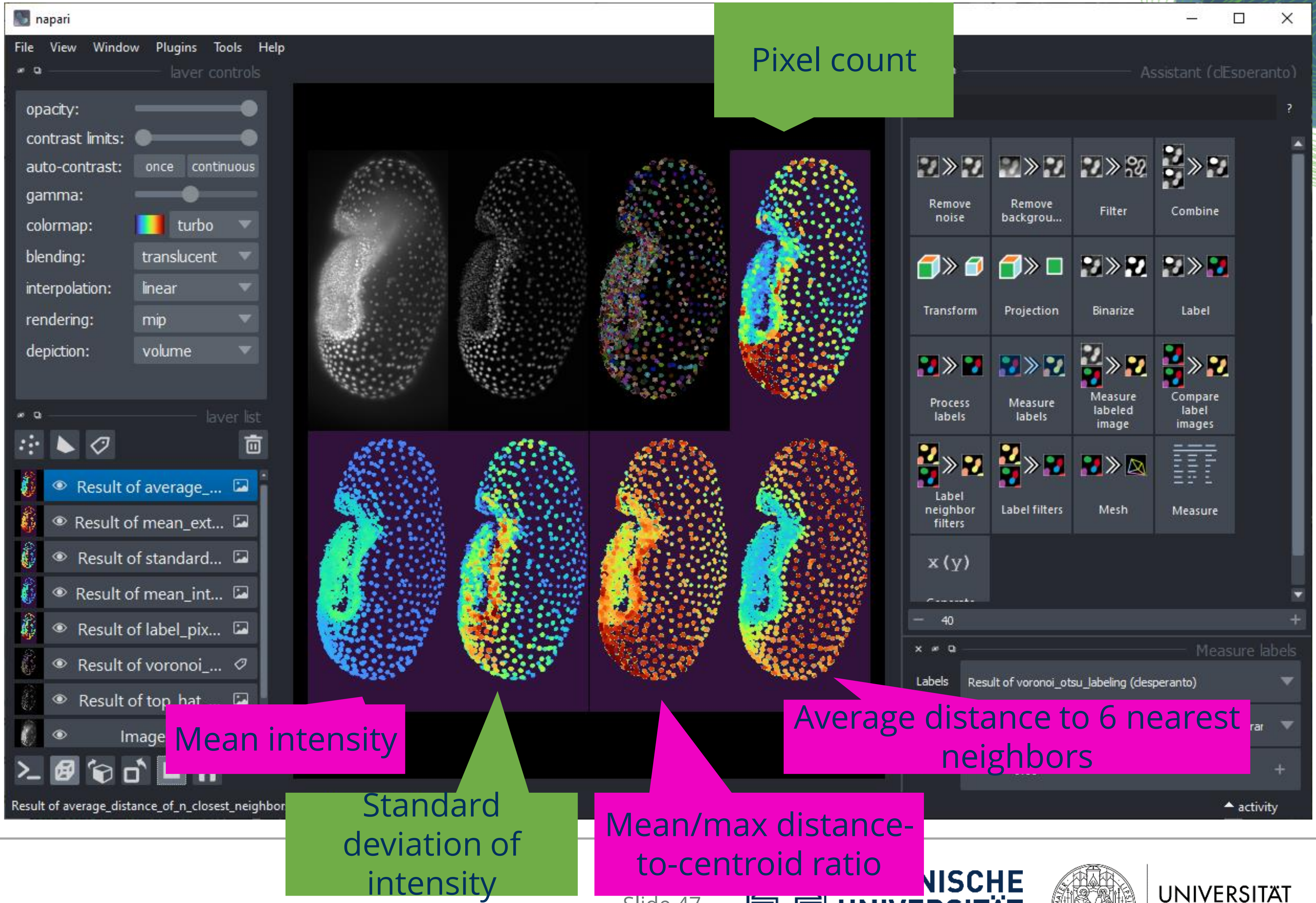
Dock widget 2

	1	2	3
area	0.060	0.000	0.000
mean_intensity	0.330	0.167	0.000
standard_deviation_intensity	0.040	0.111	0.000
mean_max_distance_to_centroid_ratio	0.260	0.444	1.000
average_distance_of_n_nearest_neighbors=6	0.310	0.278	0.000

Graphical user interface: Object classification

Inspect how the random forest classifier makes decisions

Note: Beware of correlated parameters!



Dock widget 2

	1	2	3
area	0.060	0.000	0.000
mean_intensity	0.330	0.167	0.000
standard_deviation_intensity	0.040	0.111	0.000
mean_max_distance_to_centroid_ratio	0.260	0.444	1.000
average_distance_of_n_nearest_neighbors=6	0.310	0.278	0.000

Image data source: Daniela Vorkel, Myers lab, MPI-CBG/CSBD

CENTER FOR SCALABLE DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE

Supervised and Unsupervised Machine Learning for Bio-image Analysis

Robert Haase

Funded by

Bundesministerium für Bildung und Forschung

SACHSEN

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.

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.

Sample preparation

Imaging

Shall we use a different microscope?

Should we use a different segmentation algorithm?

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

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

Imaging

Cell segmentation algorithm A, algorithm B, algorithm C

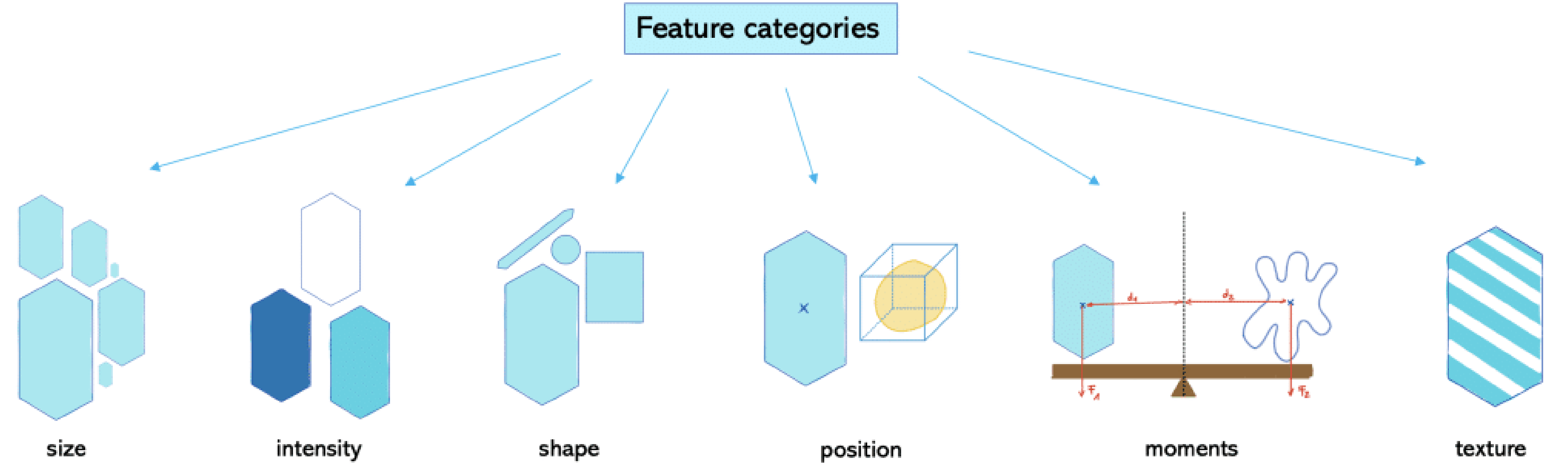
Why?

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

Statistics

Which parameter shows any relationship with X?

Feature selection



Which of these features reflect the phenotype we are perceiving?

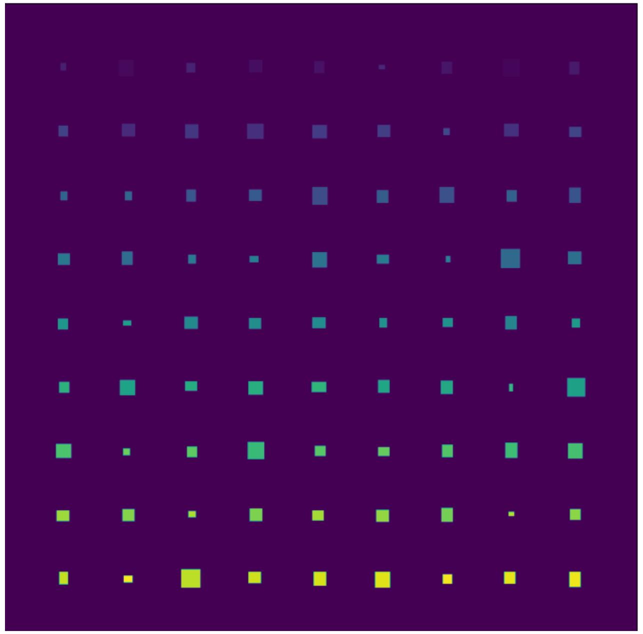
Feature selection: challenges

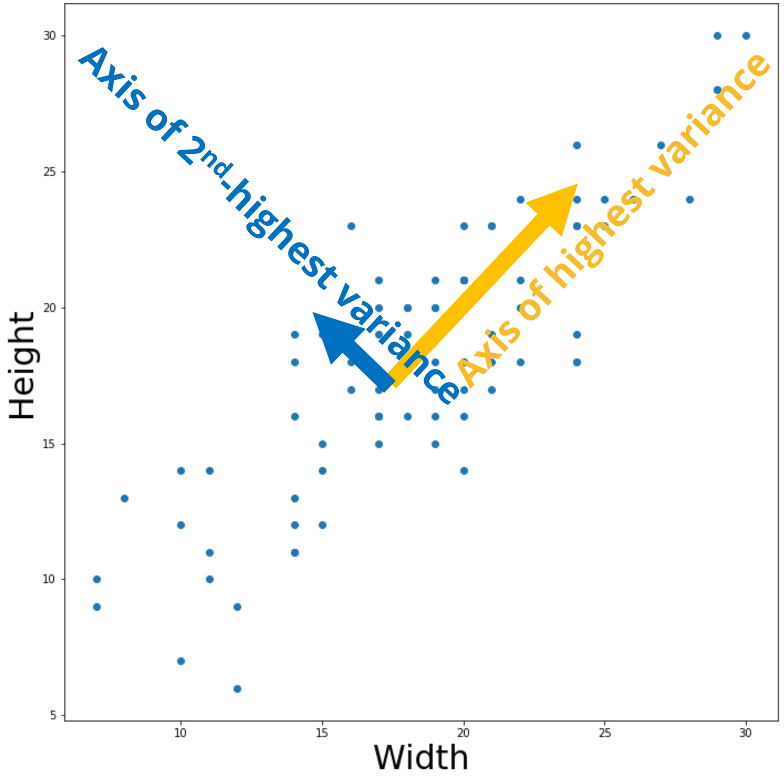
- Features are not independent
  - Area and diameter
  - Roundness, circularity, solidity, extent, aspect ratio, elongation, Feret's diameter, …
- Best classification most likely involves multiple features
- Vast amount of features can hardly be visualized
- Need for dimensionality reduction
  - Principal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

PCA: Principal Component Analysis

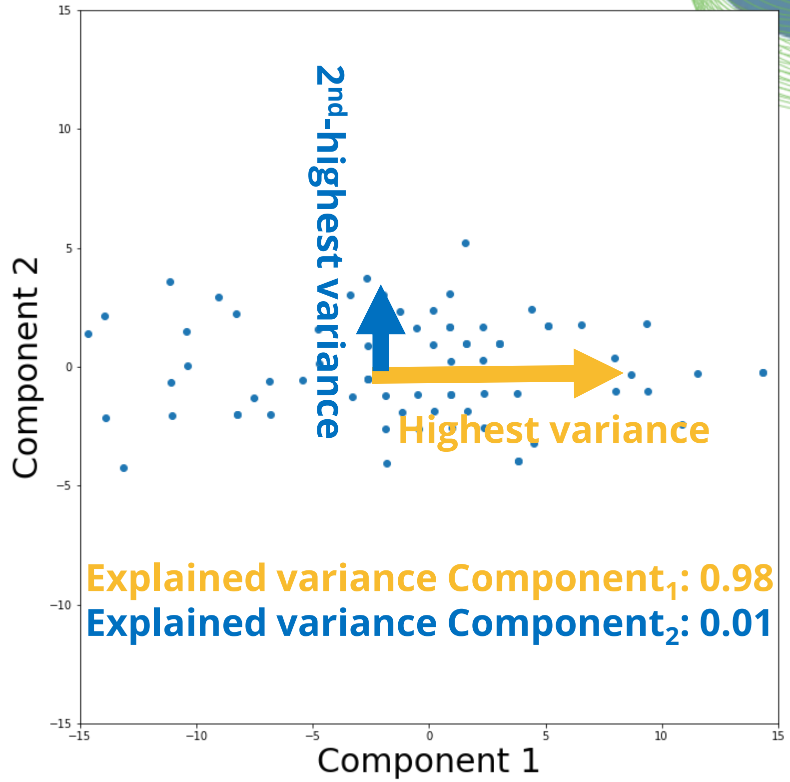
Decomposes data into linear combinations of features that explain the highest variance

Example: Squares of different size





PCA

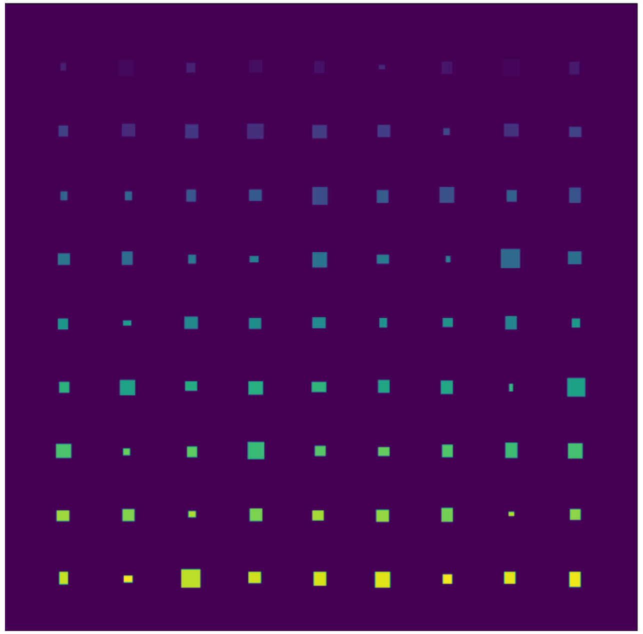


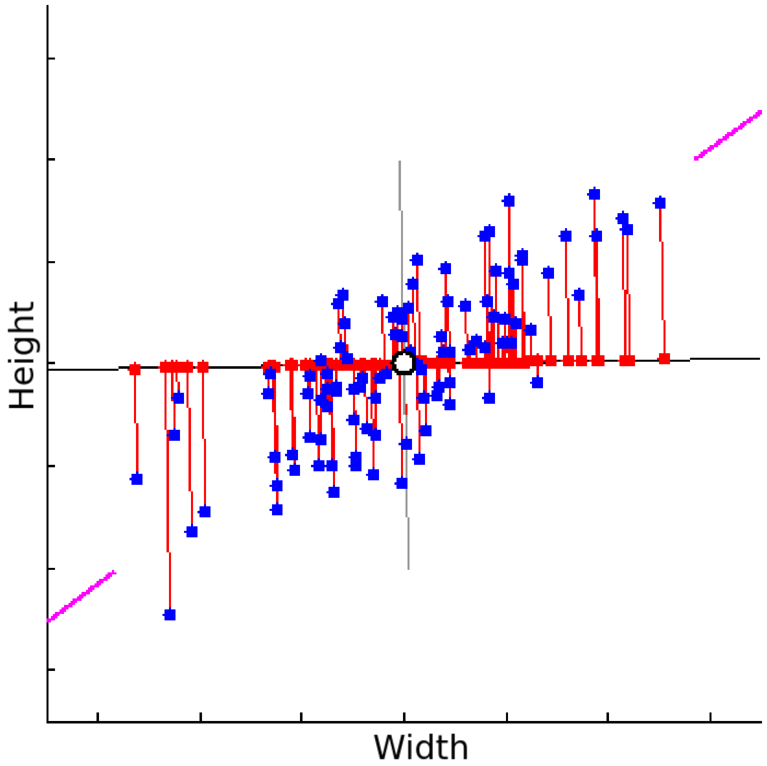
→ PCA transforms width/height measurements into a coordinate system that explains existing variance better

PCA: Principal Component Analysis

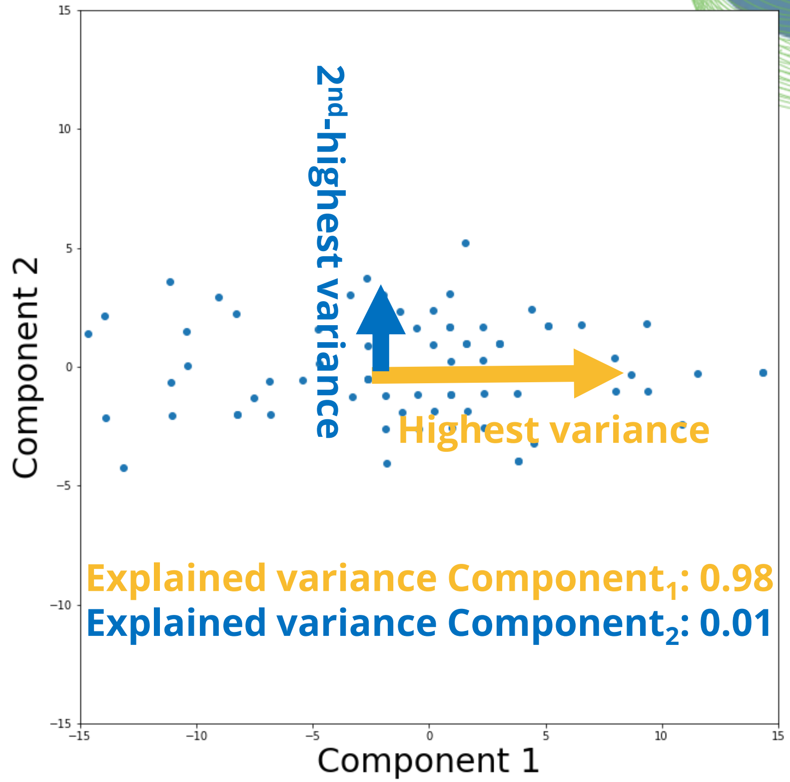
Decomposes data into linear combinations of features that explain the highest variance

Example: Squares of different size





PCA



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

PCA in Python: `sklearn.decomposition.PCA`

- Import package

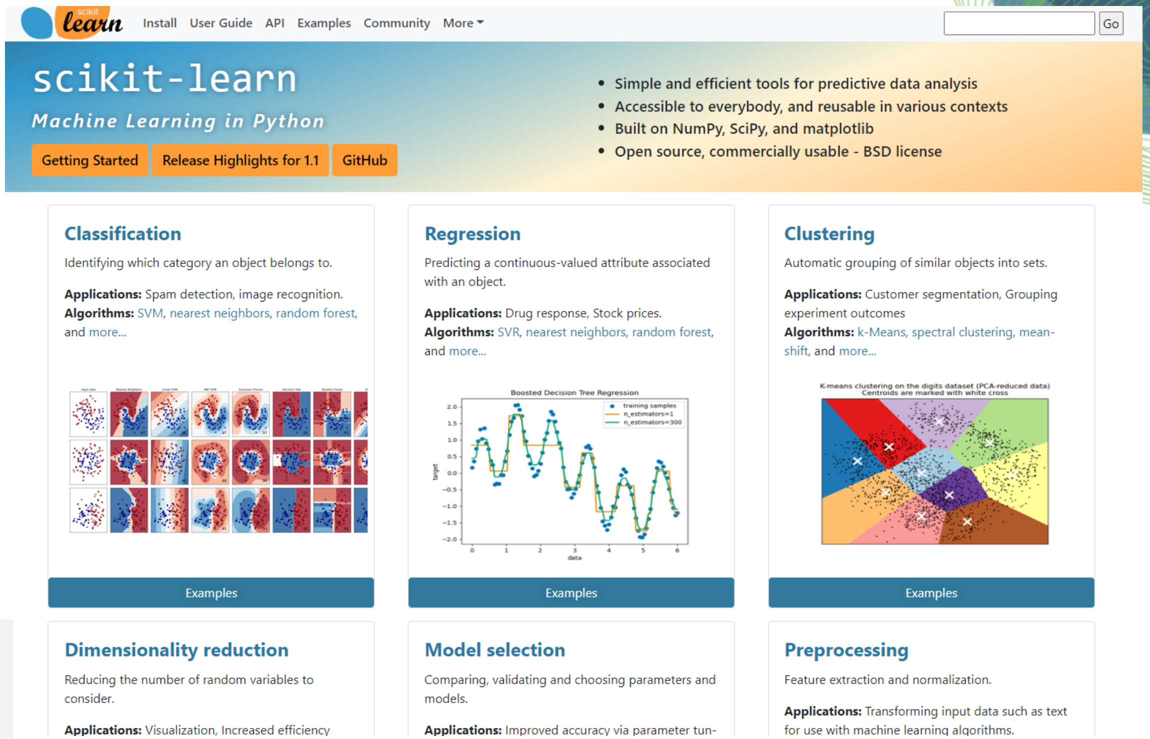
```
from sklearn.decomposition import PCA
```

- Apply PCA

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

- Transform data into new coordinate system

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



Important!

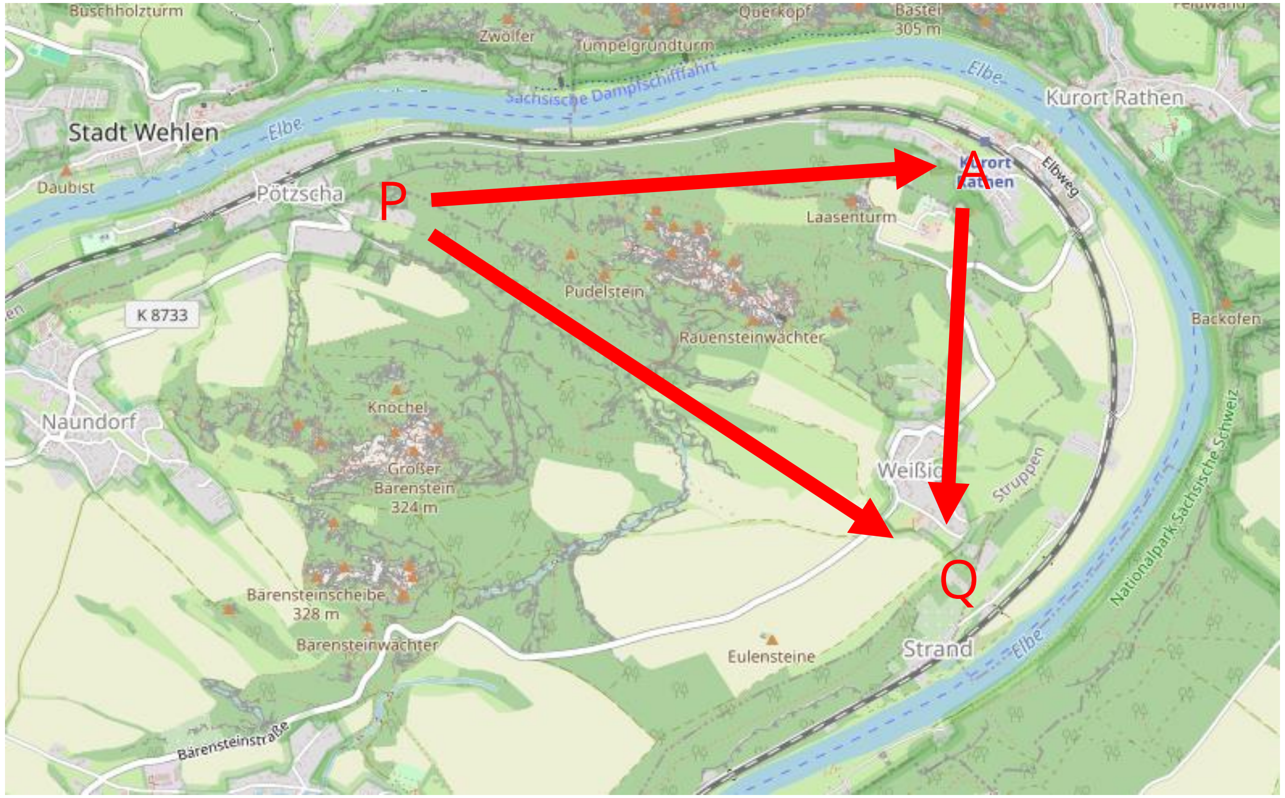
Always check the explained variance along the PCA component axes!

```
pca.explained_variance_ratio_
```

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

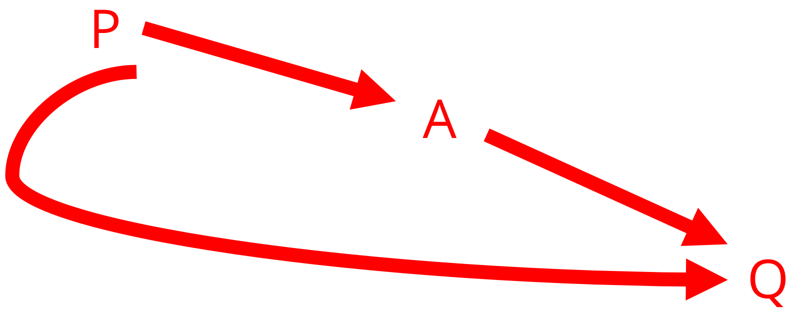
Non-Euclidian spaces

Not all dimensions (features) might be distances



Use travel time between P and Q as metric for distance

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



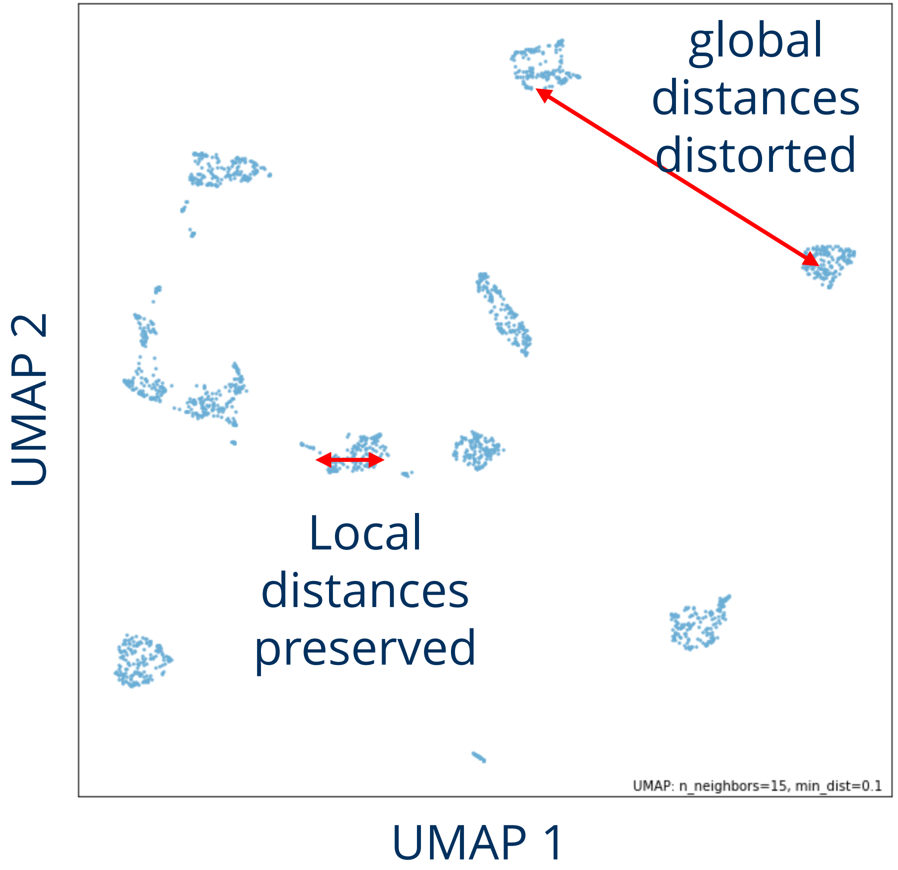
Dimensionality reduction: UMAP

Uniform Manifold Approximation Projection

Preserve local distances at the expense of global distortions

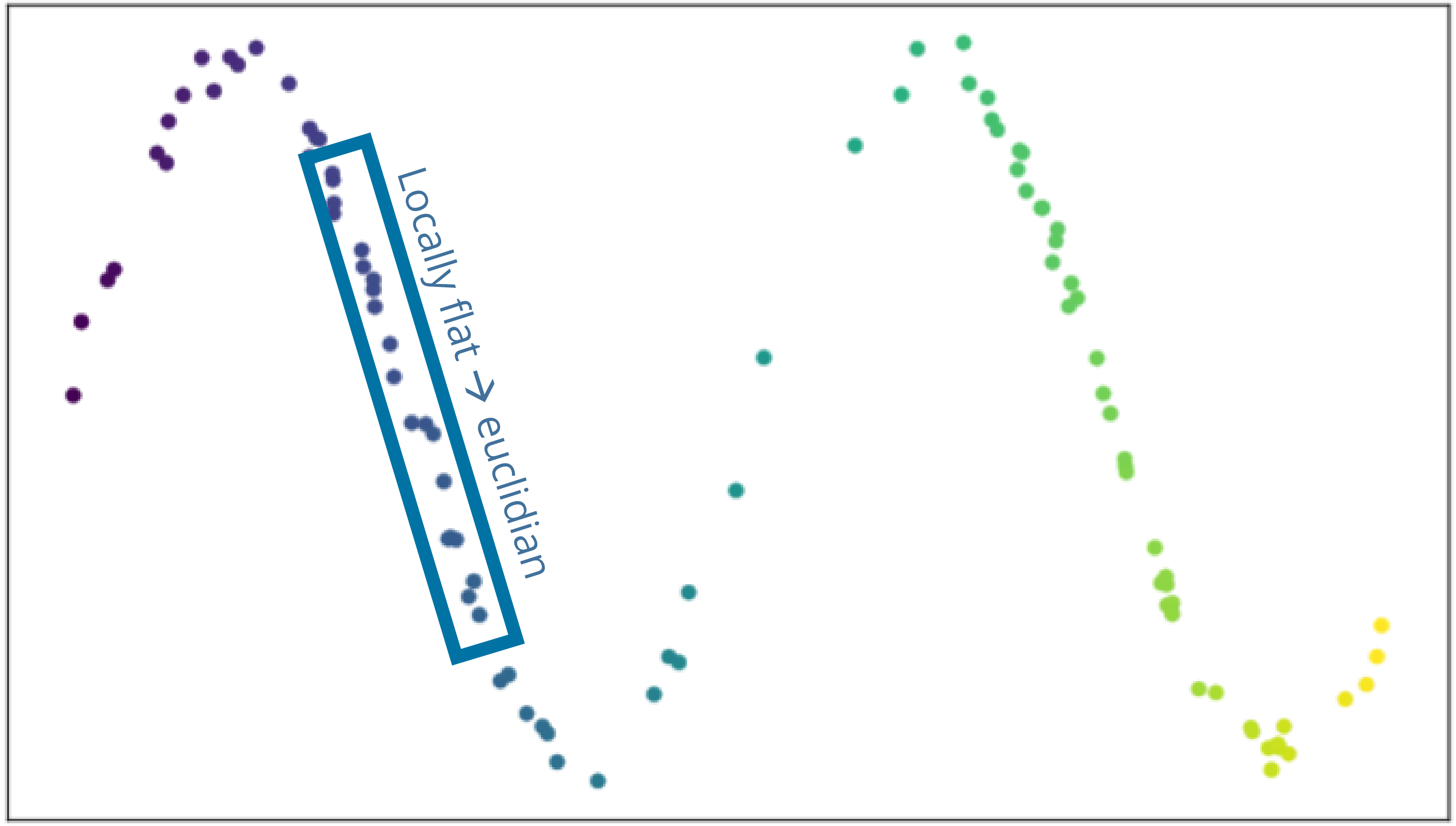
Many dimensions

	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



Dimensionality reduction: UMAP

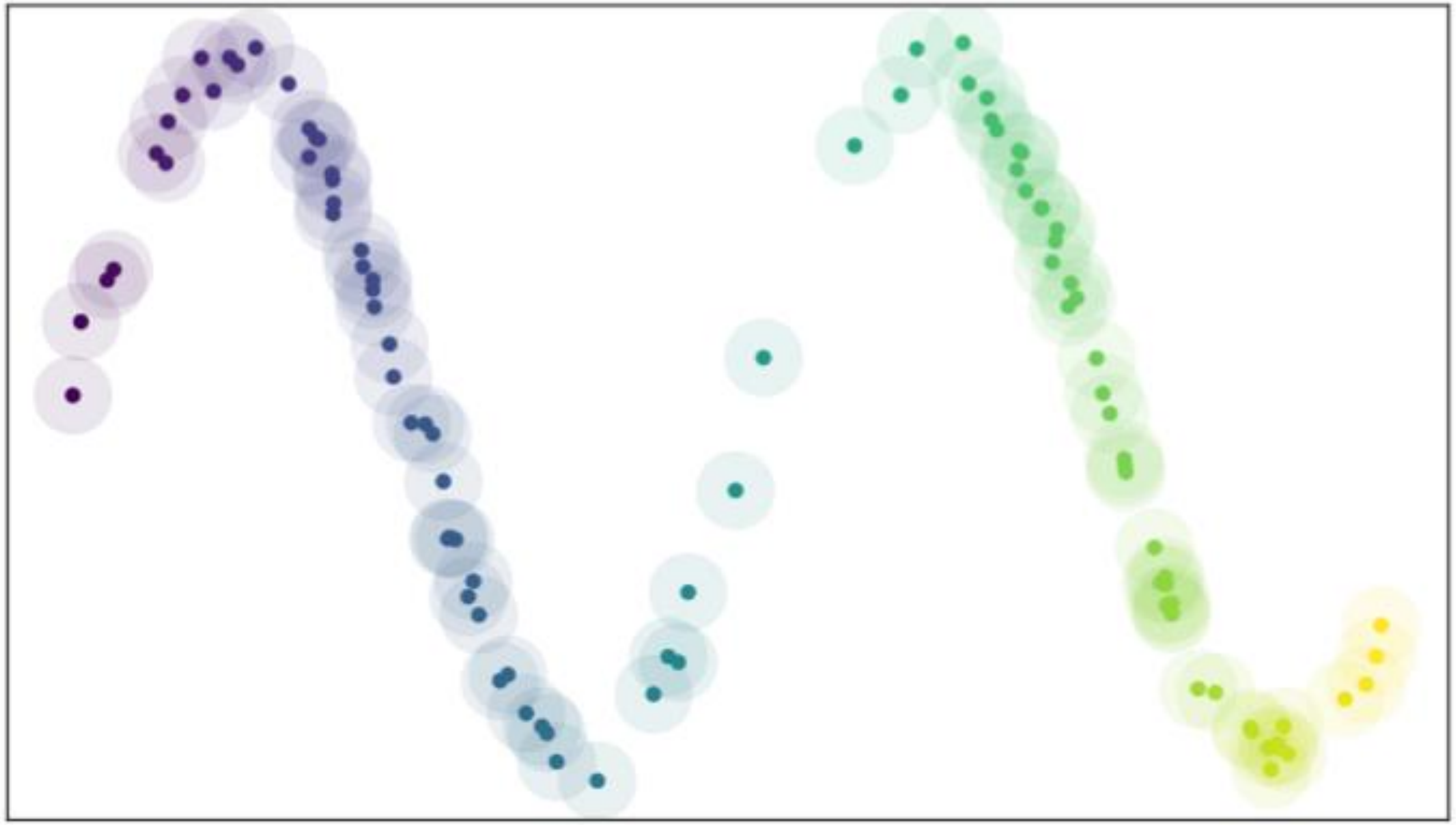
Initial situation: Our data suggests an underlying structure (“topology”)



Goal:
Reconstruct underlying topology to identify a space that best explains differences in our data

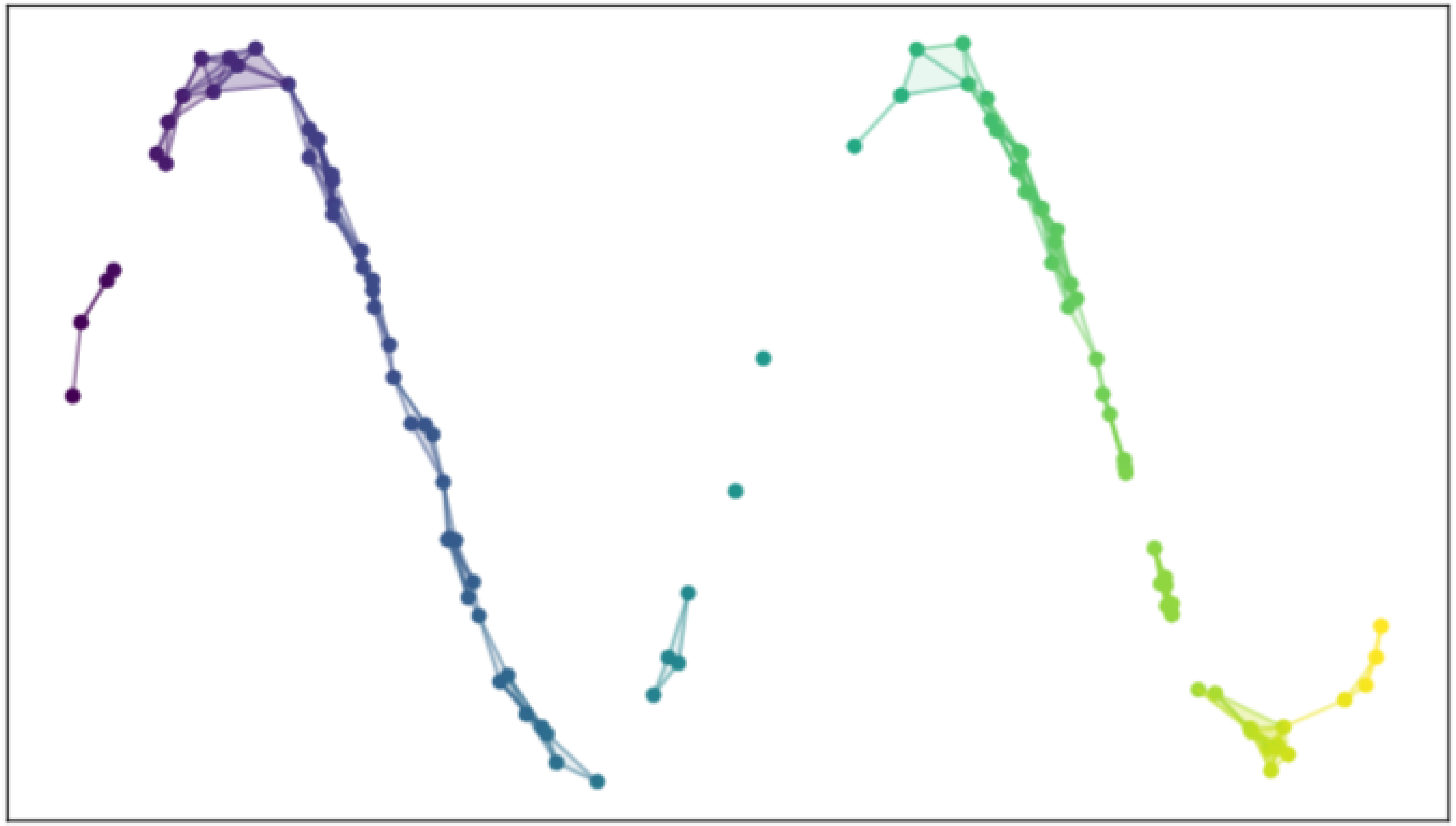
Source: https://umap-learn.readthedocs.io/en/latest/how_umap_works.html

Dimensionality reduction



Naïve approach:
Points within a defined radius are considered neighbors

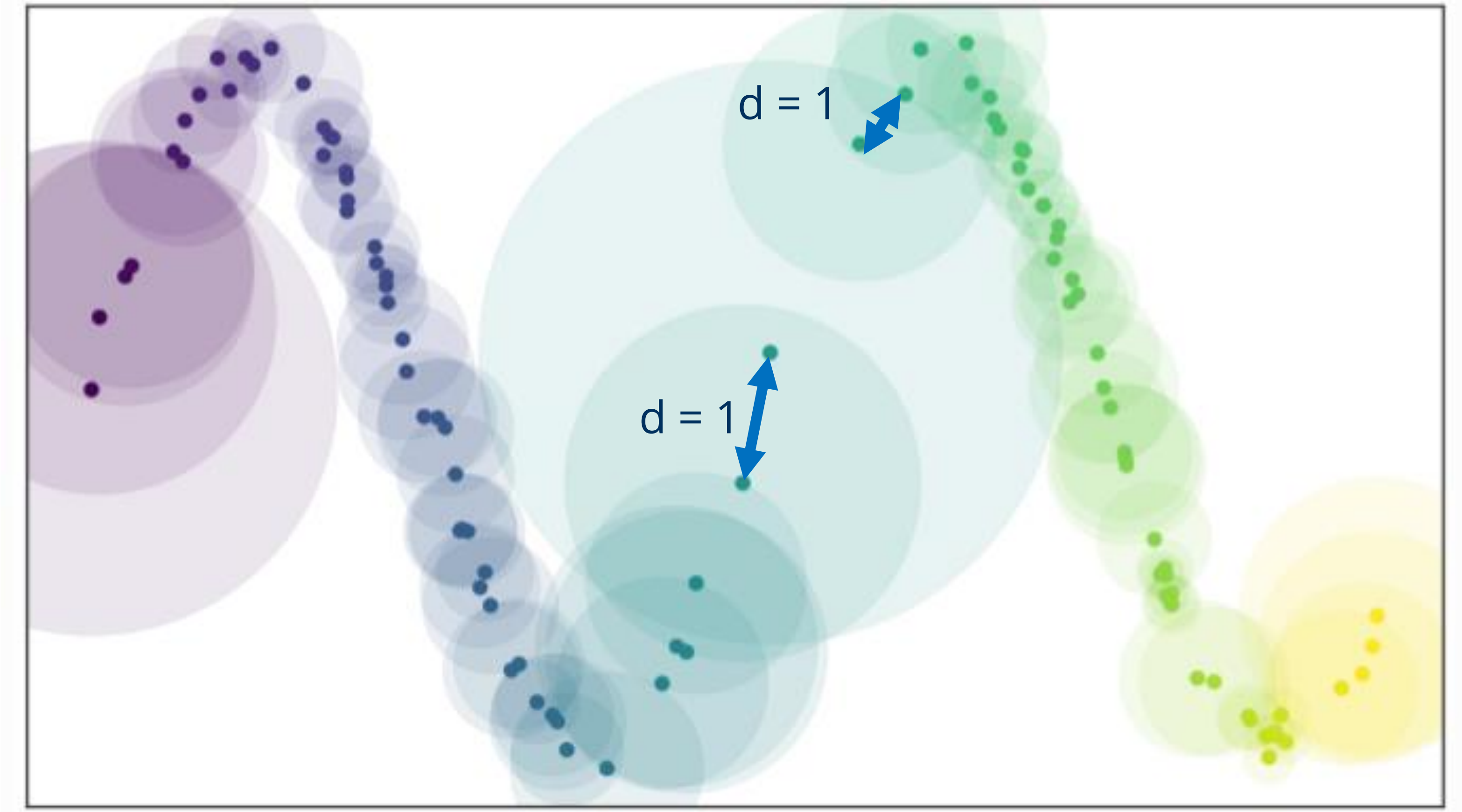
Dimensionality reduction



Naïve approach:
Points within a defined radius are considered neighbors

Result:
Neighborhood graph with interruptions

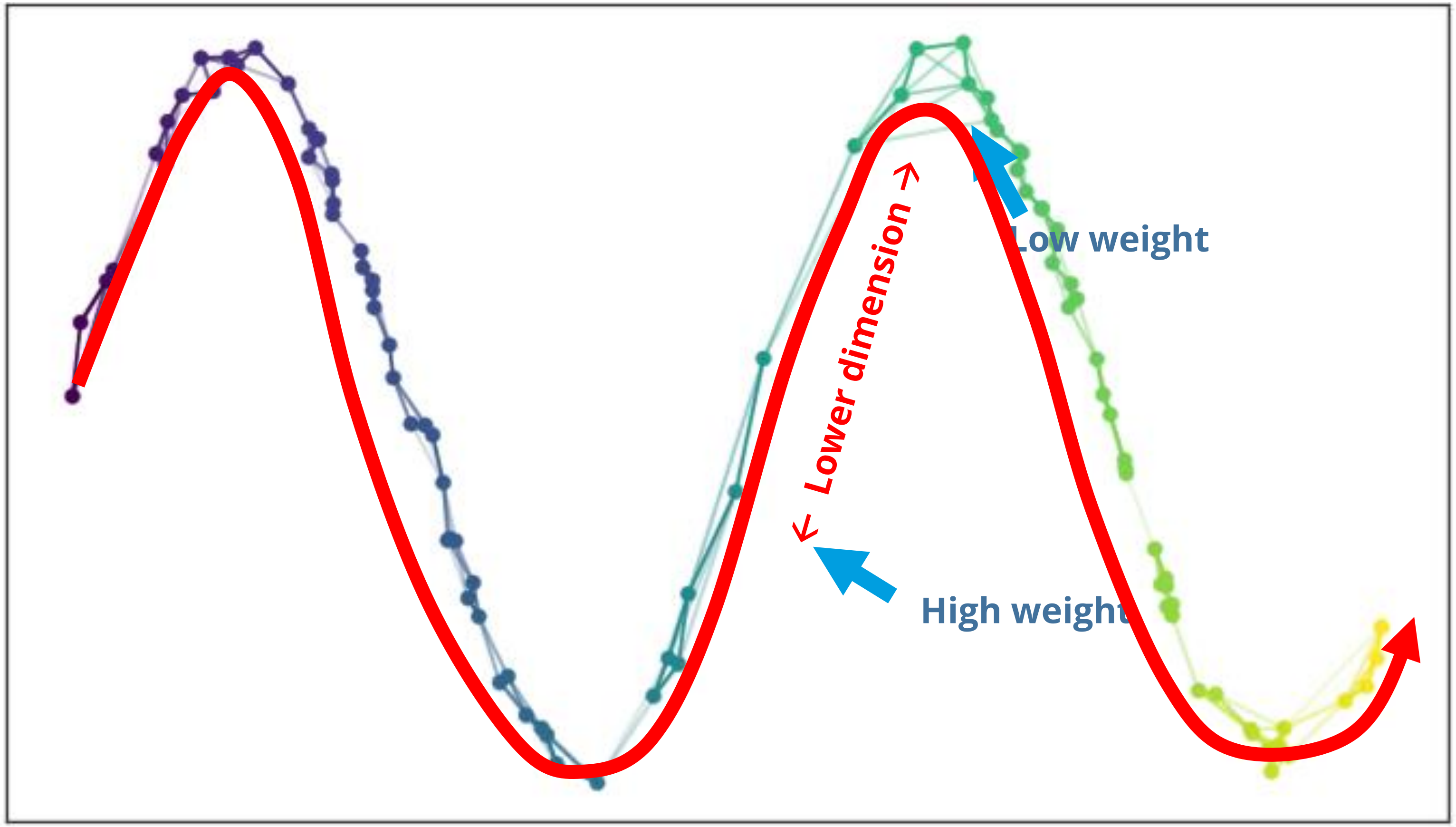
Dimensionality reduction: UMAP



Approach:
Normalize distances by dividing by the average distance to n nearest neighbors

(Example: n=1)

Reduce dimensionality preserving fuzzy topology



Approach:
Normalize distances by dividing by the average distance to n nearest neighbors

Build a graph considering normalized distances

Project data into lower dimensional space

Dimensionality reduction

Uniform manifold approximation and projection (UMAP)

t-distributed stochastic neighbor embedding (t-SNE)

Principal component analysis (PCA)

Laura Žigutytė @zigutyte

Ryan Savill @RyanSavill4

Marcelo Zoccoler @zoccolermarcelo

UMAP in Python

Selecting columns from a pandas DataFrame

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

[8]:

	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']]
selected_measurements.describe().T
```

Select *reasonable* features

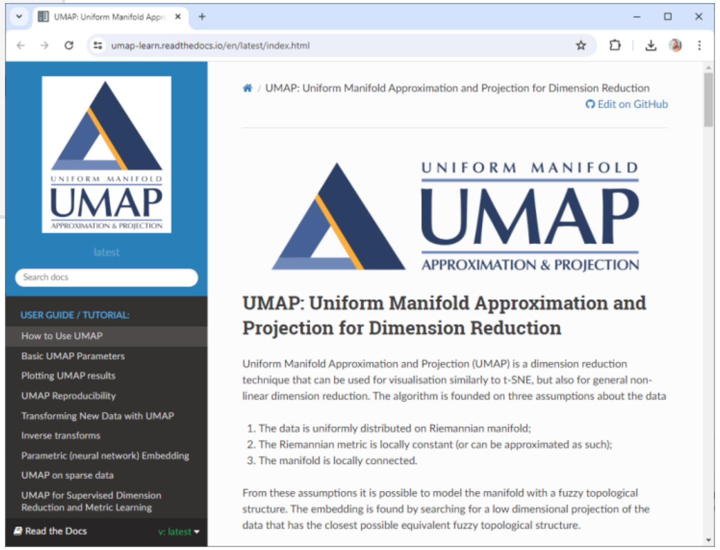
[9]:

	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

UMAP in Python

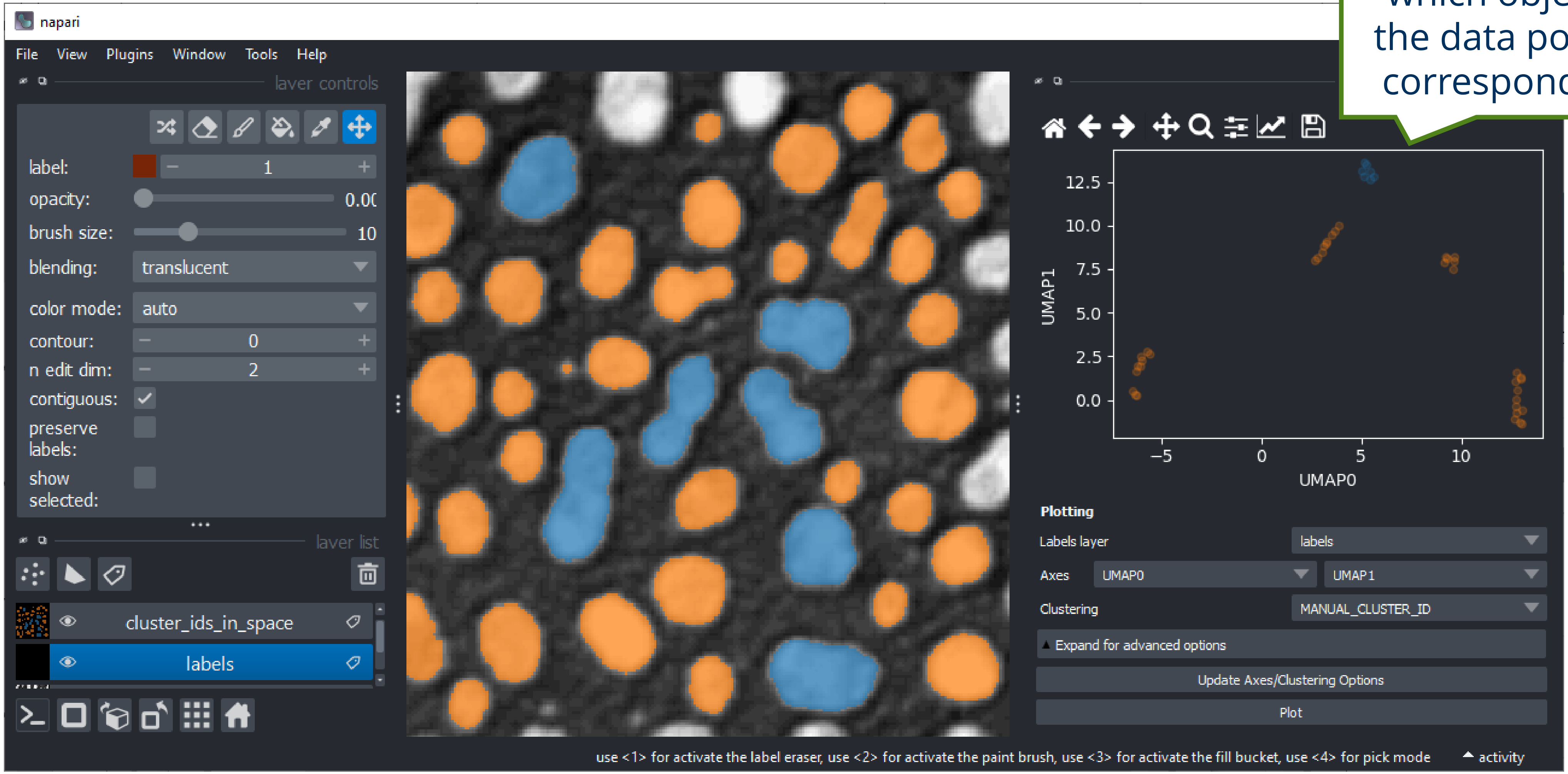
```
[10]: # configure UMAP algorithm
      umap = UMAP(n_neighbors=5, n_components=2)

      # 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

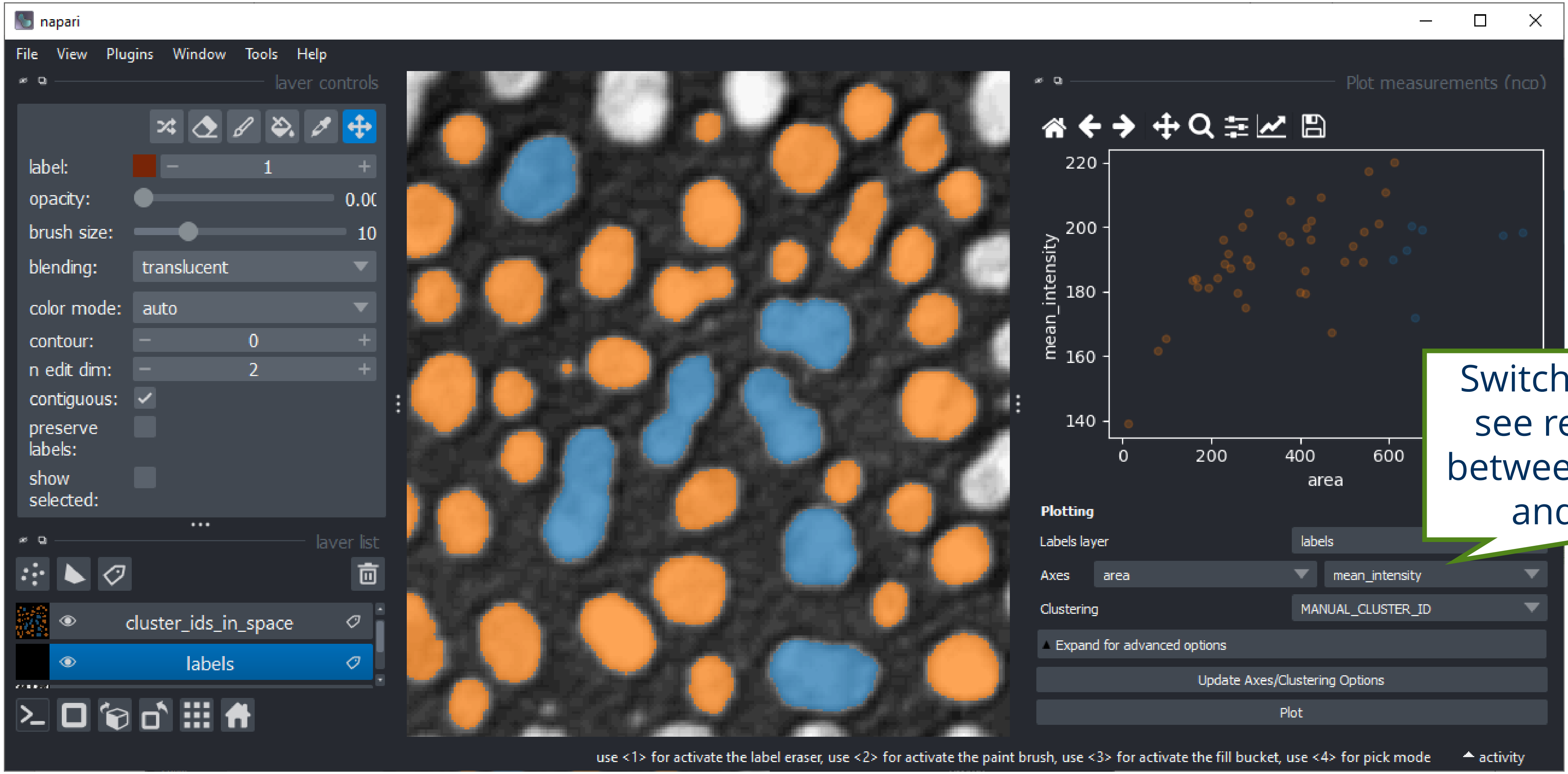


Annotating UMAPs in Napari

Draw a lasso here to visualize which objects the data points correspond to



Interpreting annotations in Napari



Switch plot axes to see relationships between annotation and features

Correlation statistics

[16]:
```python
def colorize(styler):
    styler.background_gradient(axis=None, cmap="PiYG")
    return styler

df = measurements.corr().T
df.style.pipe(colorize)
```

[16]:

	label	area	bbox_area	equivalent_diameter	convex_area	max_intensity	mean_intensity	min_intensity	extent	local_centroid-0	local_centroid-1	solidity	feret_diameter_max	major_axis_length	minor_axis_length	orientation	eccentricity
label	1.000000	0.261682	0.223070	0.249249	0.250594	0.110791	0.235692	nan	0.031673	0.177363	0.227746	0.090163	0.208067	0.198908	0.237521	0.319053	0.059804
area	0.261682	1.000000	0.973718	0.978723	0.997560	0.511730	0.530250	nan	-0.362472	0.847281	0.935689	-0.243908	0.930981	0.911069	0.859240	0.280673	0.348585
bbox_area	0.223070	0.973718	1.000000	0.948328	0.985584	0.481524	0.476951	nan	-0.546728	0.902854	0.904551	-0.416707	0.973189	0.967337	0.752580	0.213080	0.479196
equivalent_diameter	0.249249	0.978723	0.948328	1.000000	0.974614	0.633984	0.618553	nan	-0.395696	0.858779	0.947036	-0.266587	0.931696	0.904412	0.904698	0.197456	0.363799
convex_area	0.250594	0.997560	0.985584	0.974614	1.000000	0.506730	0.517356	nan	-0.413323	0.862417	0.934090	-0.305706	0.948048	0.932682	0.832264	0.263176	0.389269
max_intensity	0.110791	0.511730	0.481524	0.633984	0.506730	1.000000	0.825115	nan	-0.324093	0.504879	0.603305	-0.253635	0.536089	0.502524	0.645600	-0.139025	0.246172
mean_intensity	0.235692	0.530250	0.476951	0.618553	0.517356	0.825115	1.000000	nan	-0.160940	0.412859	0.609264	-0.077797	0.458515	0.422638	0.707711	0.132754	0.017030
min_intensity	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
extent	0.031673	-0.362472	-0.546728	-0.395696	-0.413323	-0.324093	-0.160940	nan	1.000000	-0.631158	-0.375580	0.853431	-0.631776	-0.664733	-0.062873	0.252915	-0.756019
local_centroid-0	0.177363	0.847281	0.902854	0.858779	0.862417	0.504879	0.412859	nan	-0.631158	1.000000	0.706437	-0.439244	0.937673	0.932889	0.623186	0.003490	0.560853
local_centroid-1	0.227746	0.935689	0.904551	0.947036	0.934090	0.603305	0.609264	nan	-0.375580	0.706437	1.000000	-0.290177	0.863585	0.840724	0.875044	0.271191	0.318154
solidity	0.090163	-0.243908	-0.416707	-0.266587	-0.305706	-0.253635	-0.077797	nan	0.853431	-0.439244	-0.290177	1.000000	-0.512903	-0.556555	0.049965	0.279509	-0.723572
feret_diameter_max	0.208067	0.930981	0.973189	0.931696	0.948048	0.536089	0.458515	nan	-0.631776	0.937673	0.863585	-0.512903	1.000000	0.996744	0.690639	0.077145	0.614849
major_axis_length	0.198908	0.911069	0.967337	0.904412	0.932682	0.502524	0.422638	nan	-0.664733	0.932889	0.840724	-0.556555	0.996744	1.000000	0.639308	0.076773	0.647021
minor_axis_length	0.237521	0.859240	0.752580	0.904698	0.832264	0.645600	0.707711	nan	-0.062873	0.623186	0.875044	0.049965	0.690639	0.639308	1.000000	0.278107	-0.012148
orientation	0.319053	0.280673	0.213080	0.197456	0.263176	-0.139025	0.132754	nan	0.252915	0.003490	0.271191	0.279509	0.077145	0.076773	0.278107	1.000000	-0.305652
eccentricity	0.059804	0.348585	0.479196	0.363799	0.389269	0.246172	0.017030	nan	-0.756019	0.560853	0.318154	-0.723572	0.614849	0.647021	-0.012148	-0.305652	1.000000
standard_deviation_intensity	0.189165	0.288670	0.267528	0.402328	0.285105	0.867057	0.902001	nan	-0.216260	0.284331	0.379400	-0.169801	0.306228	0.280378	0.455324	-0.089349	0.107307
aspect_ratio	0.036433	0.411794	0.581132	0.386884	0.462720	0.121313	-0.044872	nan	-0.848271	0.678234	0.321805	-0.787587	0.690082	0.736200	-0.030443	-0.181927	0.853302
roundness	-0.055815	-0.415592	-0.569335	-0.406856	-0.464090	-0.191680	0.009002	nan	0.834550	-0.638667	-0.359961	0.801971	-0.690444	-0.732103	0.003699	0.224205	-0.955978
circularity	-0.054152	-0.626241	-0.718764	-0.701230	-0.659125	-0.636372	-0.411166	nan	0.808533	-0.785693	-0.644979	0.773934	-0.832660	-0.839196	-0.435236	0.242901	-0.779895
UMAP0	-0.065835	-0.442711	-0.413779	-0.509190	-0.435101	-0.324496	-0.387465	nan	0.168523	-0.391875	-0.488311	0.068021	-0.457079	-0.437340	-0.479807	0.025473	-0.204662
UMAP1	0.139702	0.819263	0.813951	0.793707	0.821940	0.391350	0.365621	nan	-0.375632	0.720004	0.753502	-0.260000	0.753713	0.736954	0.702828	0.277117	0.251959
MANUAL_CLUSTER_ID	0.080739	0.677335	0.719434	0.590973	0.700457	0.156570	0.074372	nan	-0.371454	0.582543	0.616873	-0.418390	0.671673	0.686248	0.387847	0.163152	0.424045

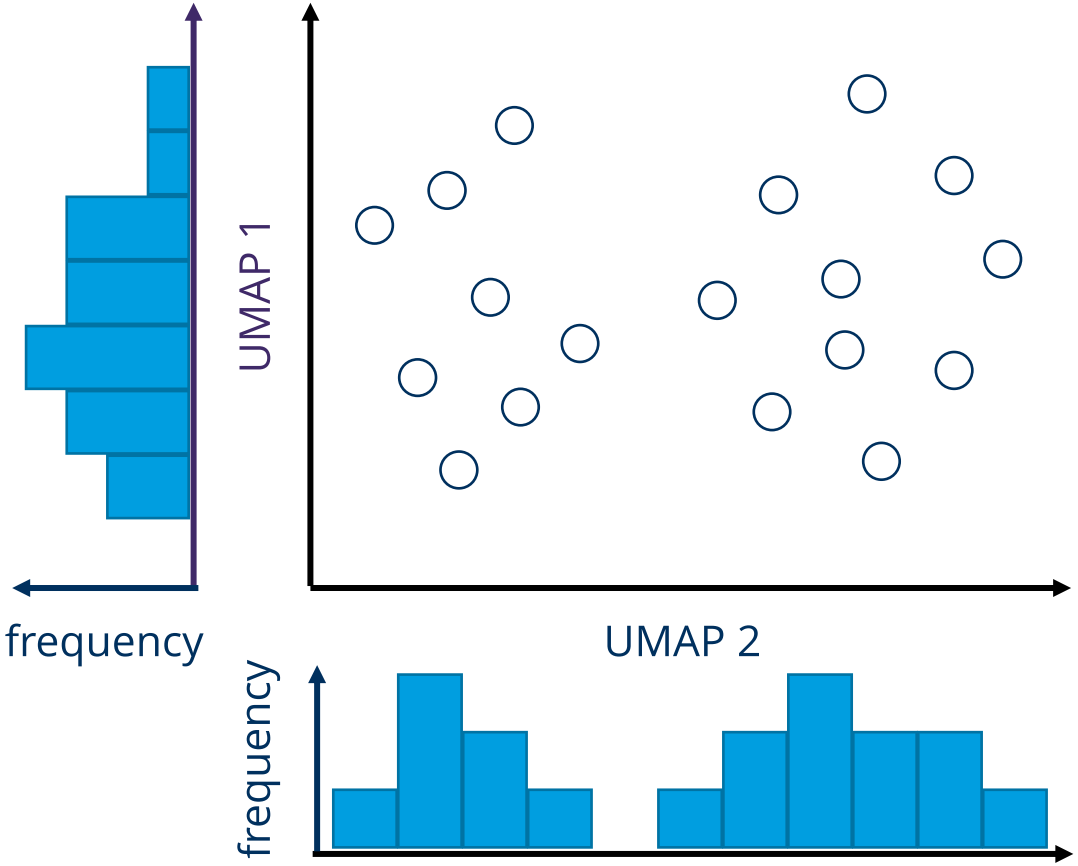
My annotation *seems related* to area

My annotation *seems not related* to intensity

Robert Haase
@haesleinhuepf

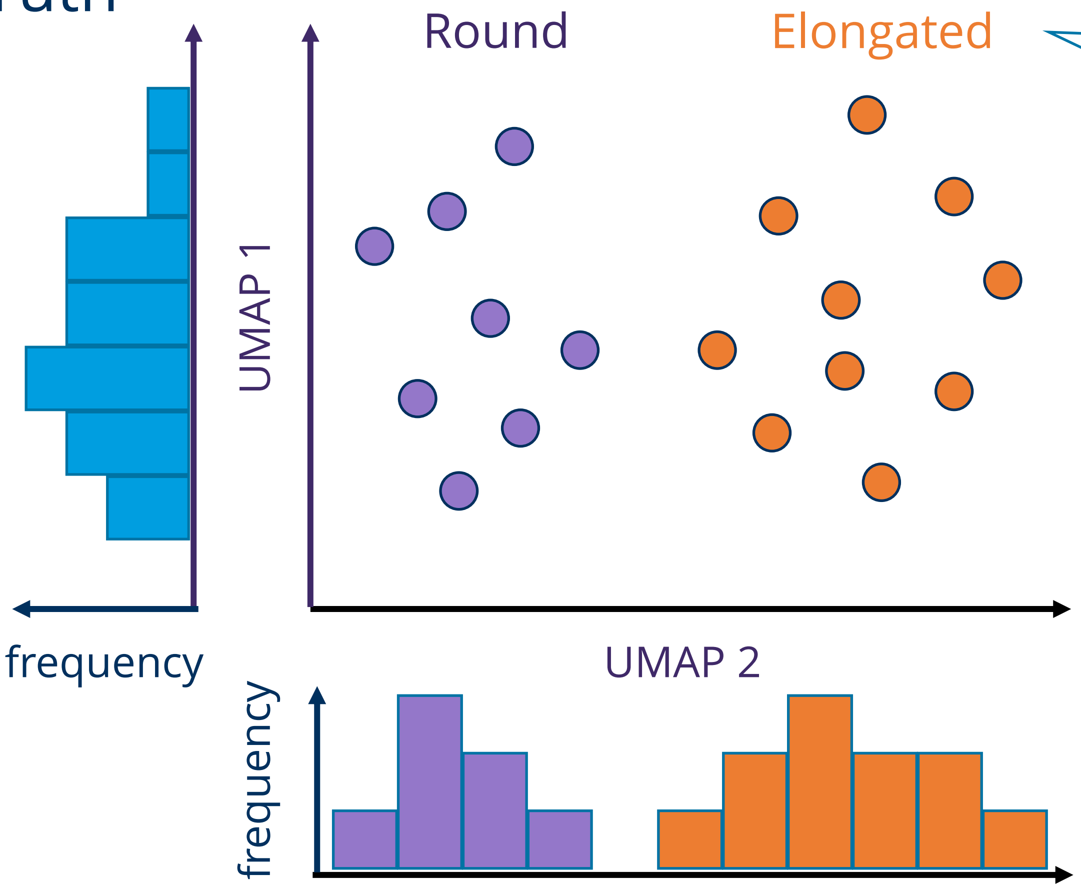
Clustering

Unsupervised machine learning may include grouping objects without given ground truth



Clustering

Unsupervised machine learning may include grouping objects without given ground truth



Names given by human observer after grouping / clustering

K-Means Clustering

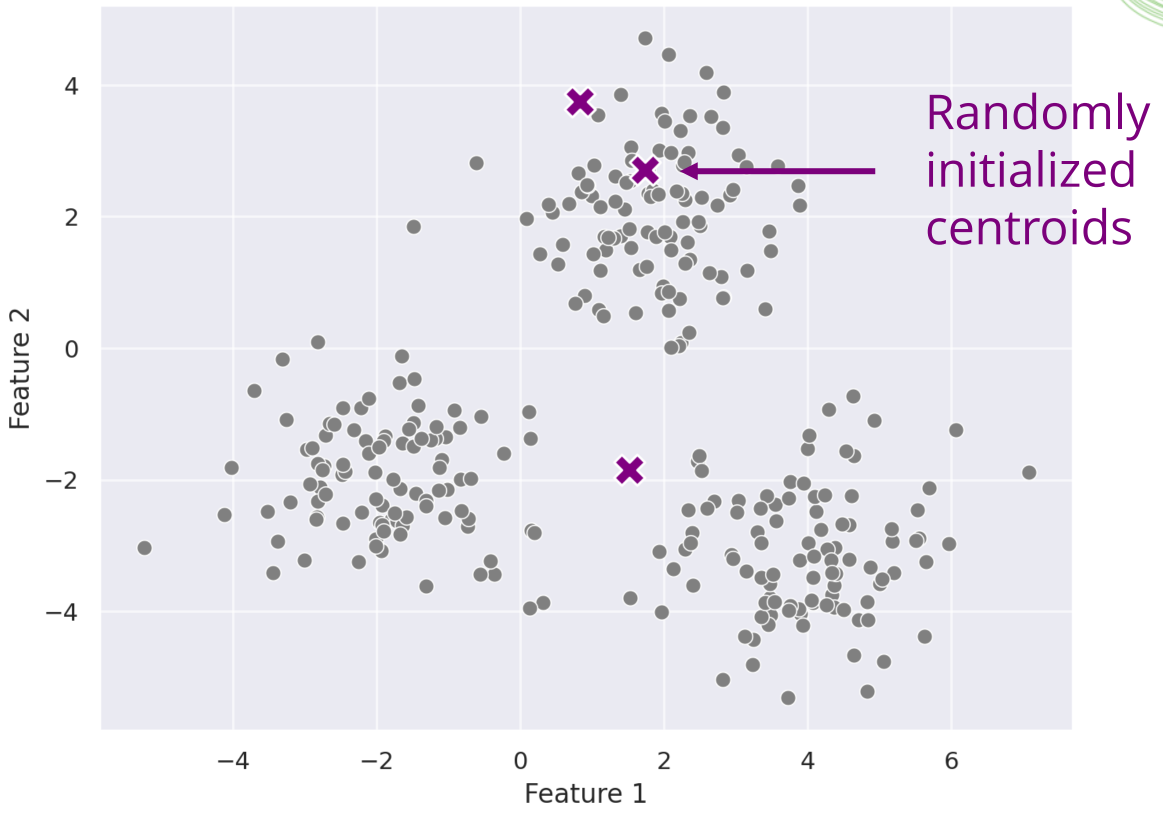
Goal: group data points into k groups so that variance within group is minimal.

STEP 1: Seed k initial cluster centroids randomly

STEP 2: Assign all points to nearest centroid

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}$$

n – dimensionality, in this example = 2



K-Means Clustering

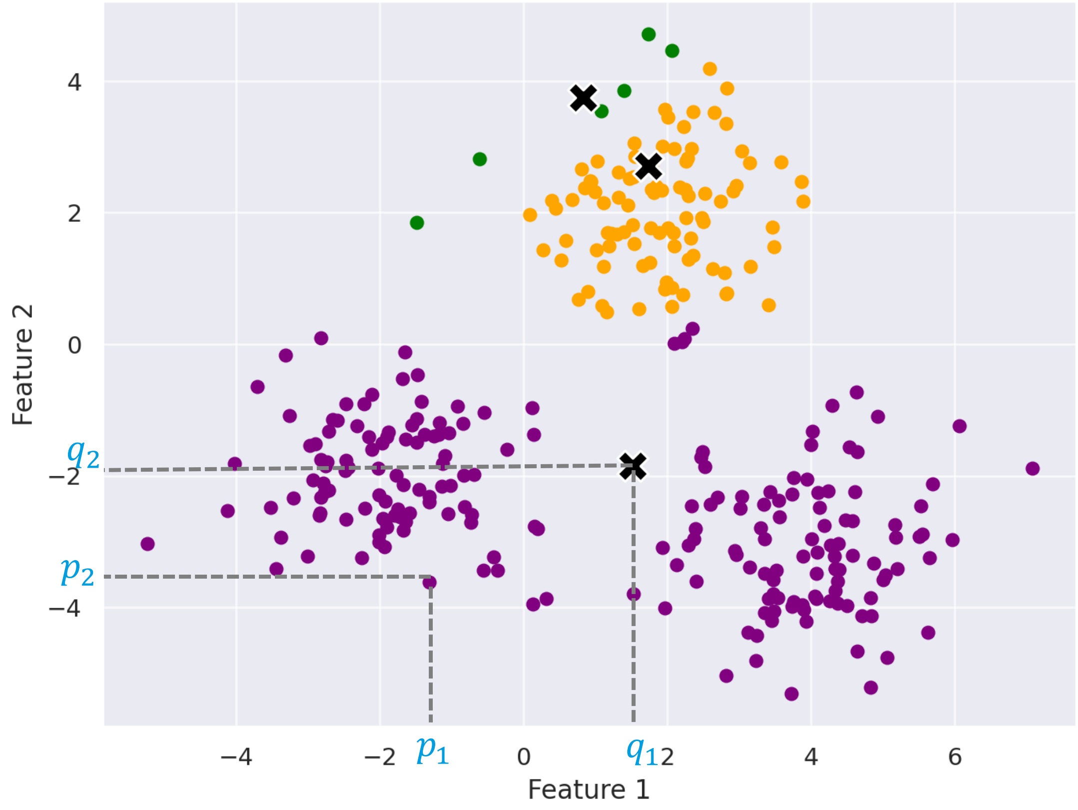
Goal: group data points into k groups so that variance within group is minimal.

STEP 1: Seed k initial cluster centroids randomly

STEP 2: Assign all points to nearest centroid

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}$$

n – dimensionality, in this example = 2



K-Means Clustering

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

STEP 3: Determine new centroid positions as mean position of all assigned points.

$$\text{New centroid}_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$$

C_i - the number of data points in cluster i

Repeat steps 2-3: the assignment and update steps are repeated iteratively until:

- Centroids not changing anymore,
- Point assignments not chainging anymore or
- Maximum number of iterations reached

K-Means Clustering

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

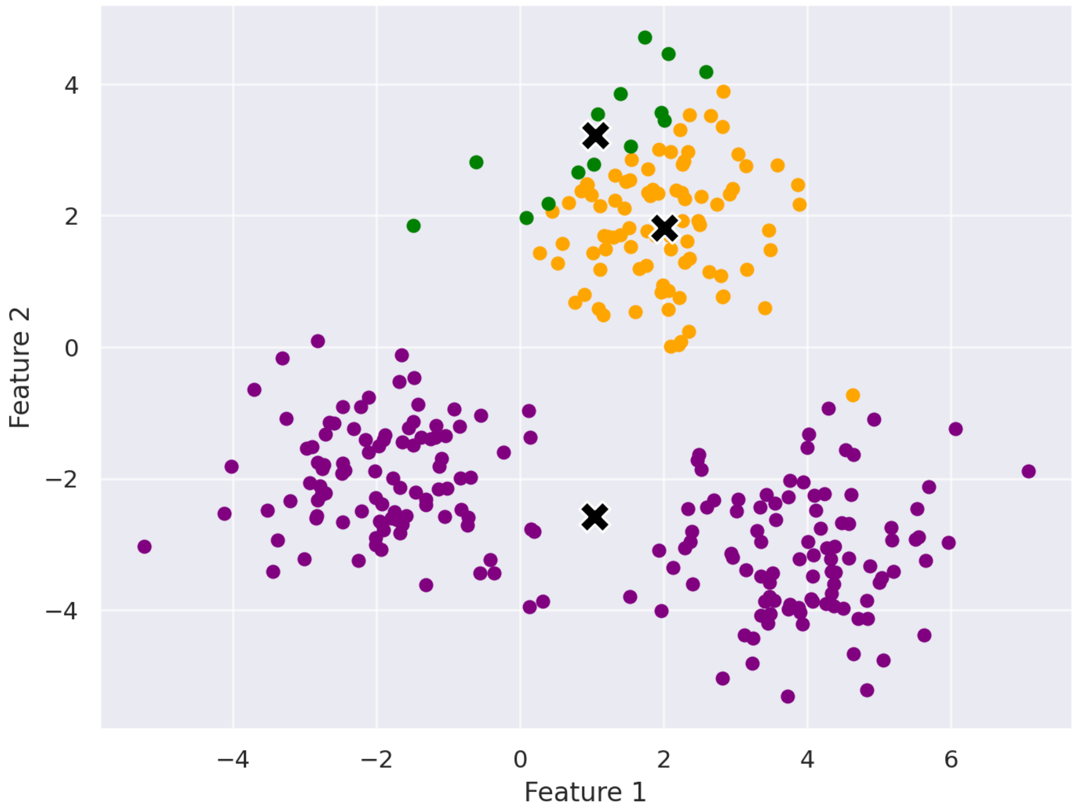
STEP 3: Determine new centroid positions as mean position of all assigned points.

$$\text{New centroid}_i = \frac{1}{|C_i|}\sum_{x \in C_i} x$$

C_i - the number of data points in cluster i

Repeat steps 2-3: the assignment and update steps are repeated iteratively until:

- Centroids not changing anymore,
- Point assignments not chainging anymore or
- Maximum number of iterations reached



K-Means Clustering

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

In Python:

```python
from sklearn import cluster
```

Create

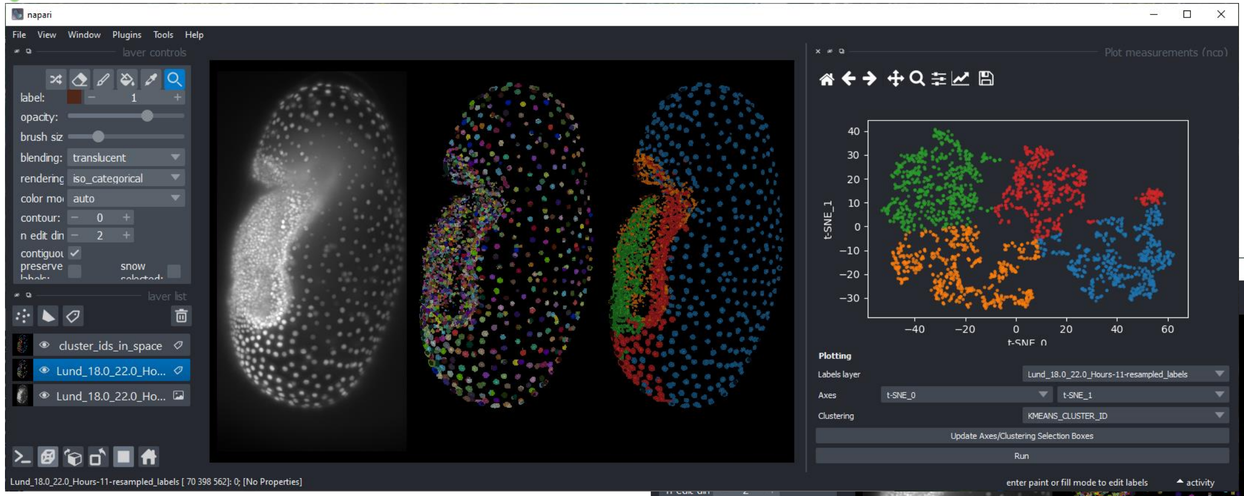
```python
clusterer = cluster.KMeans(n_clusters=3)
clusterer.fit(X)
```

Predict

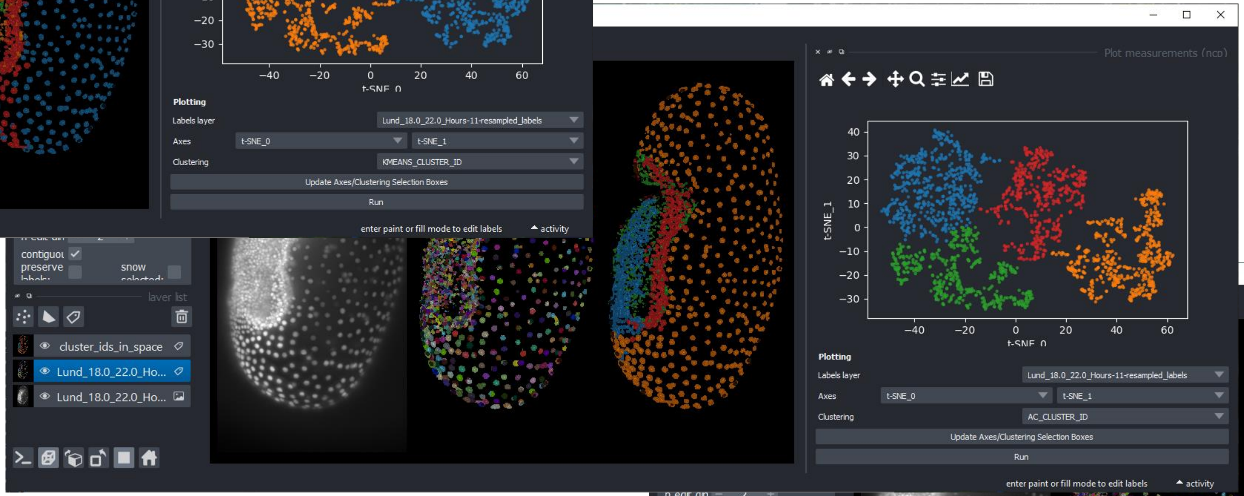
```python
predicted_class = clusterer.predict(X)
```

Clustering

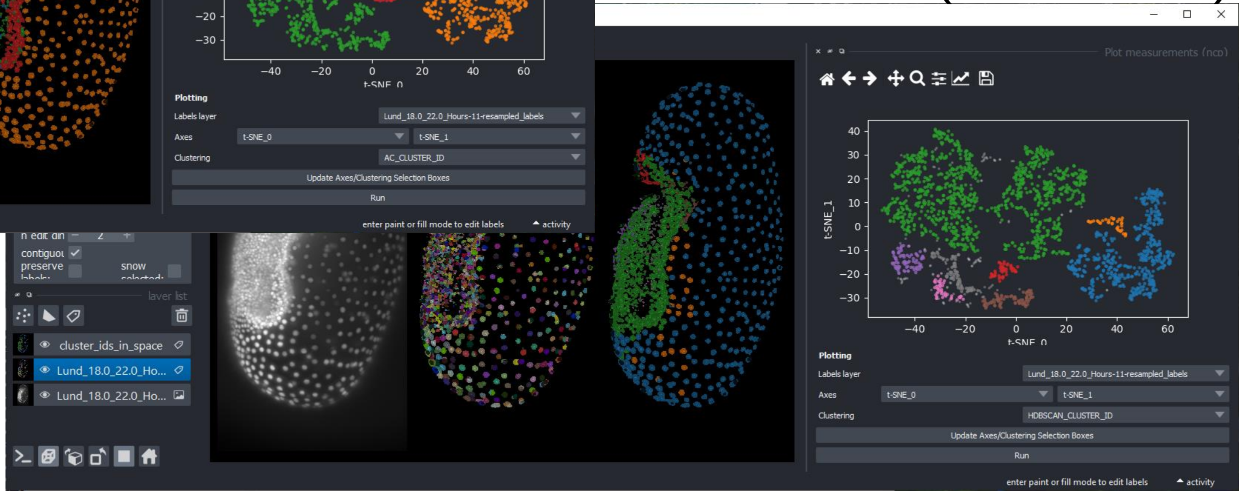
K-means clustering



Agglomerative clustering



Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN)



Laura Žigutytė
@zigutyte

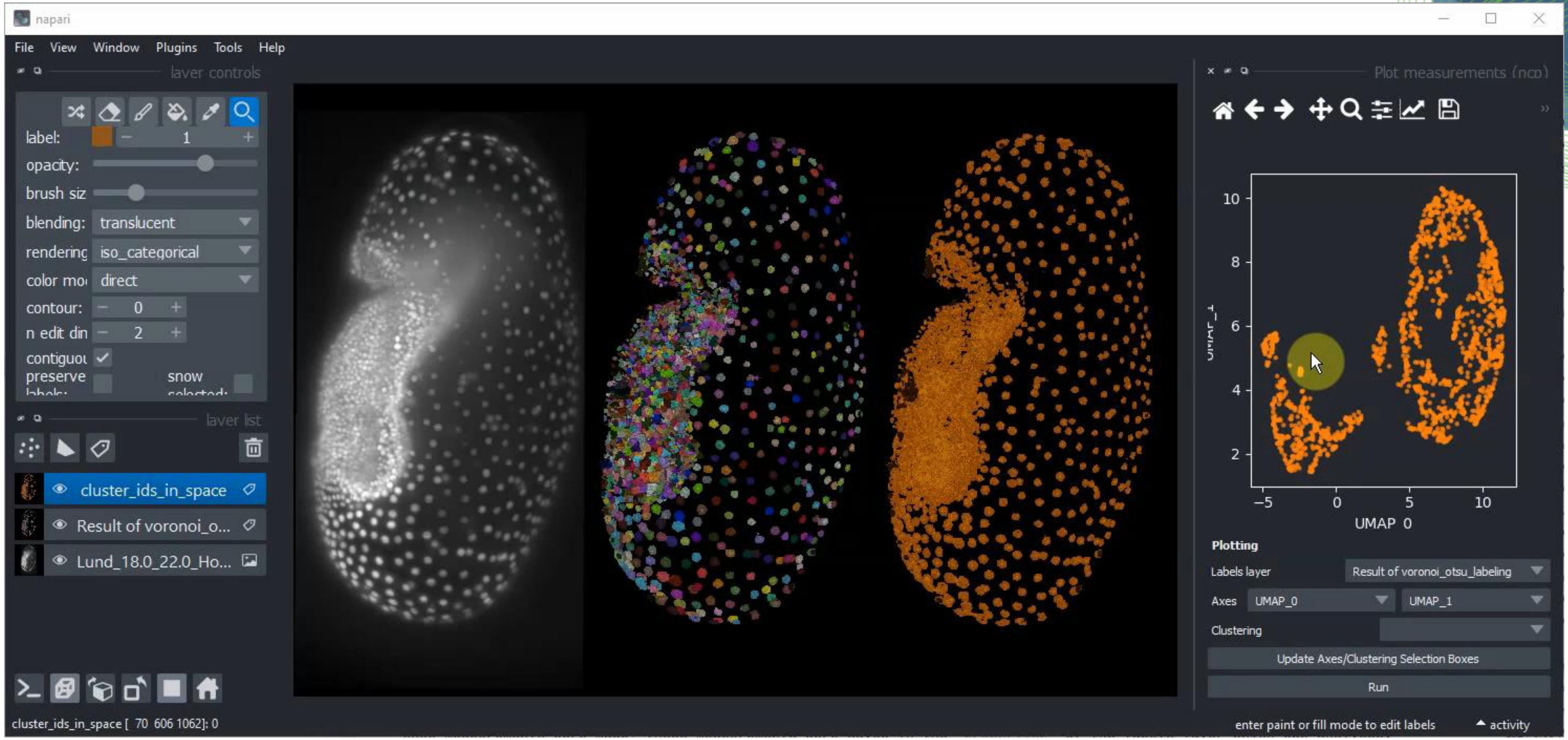
Ryan Savill
@RyanSavill4

Marcelo Zoccoler
@zoccolermarcelo

https://github.com/BiAPoL/napari-clusters-plotter

Manual clustering

To better understand relationships between data



Laura Žigutytė
@zigutyte

Ryan Savill
@RyanSavill4

Marcelo Zoccoler
@zoccolermarcelo

ScaDS.AI DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE

Exercises

Robert Haase

Funded by

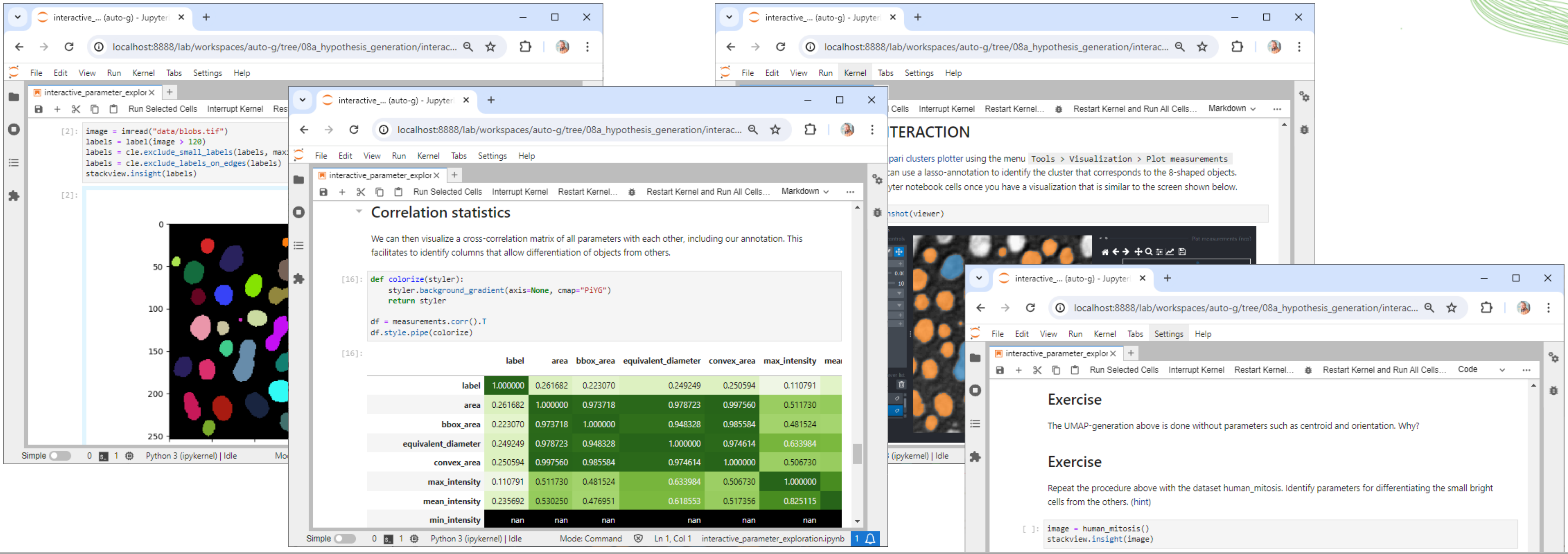
Bundesministerium für Bildung und Forschung

SACHSEN

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



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

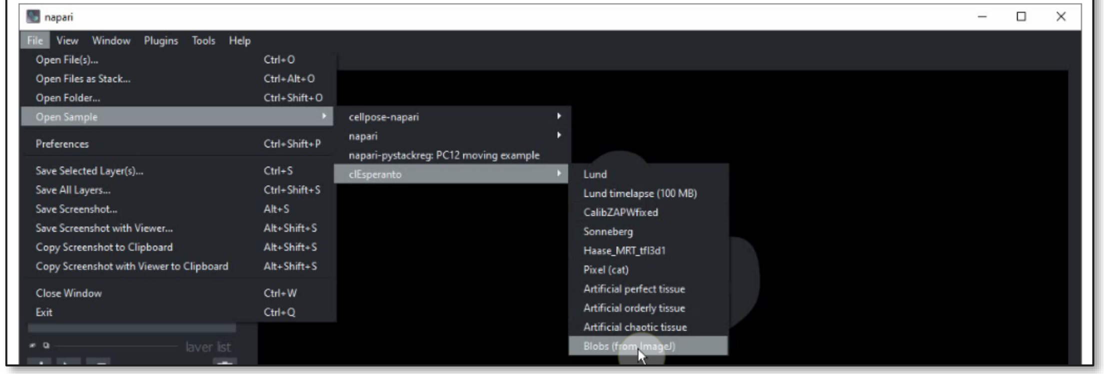
Open a terminal window and activate your conda environment:

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

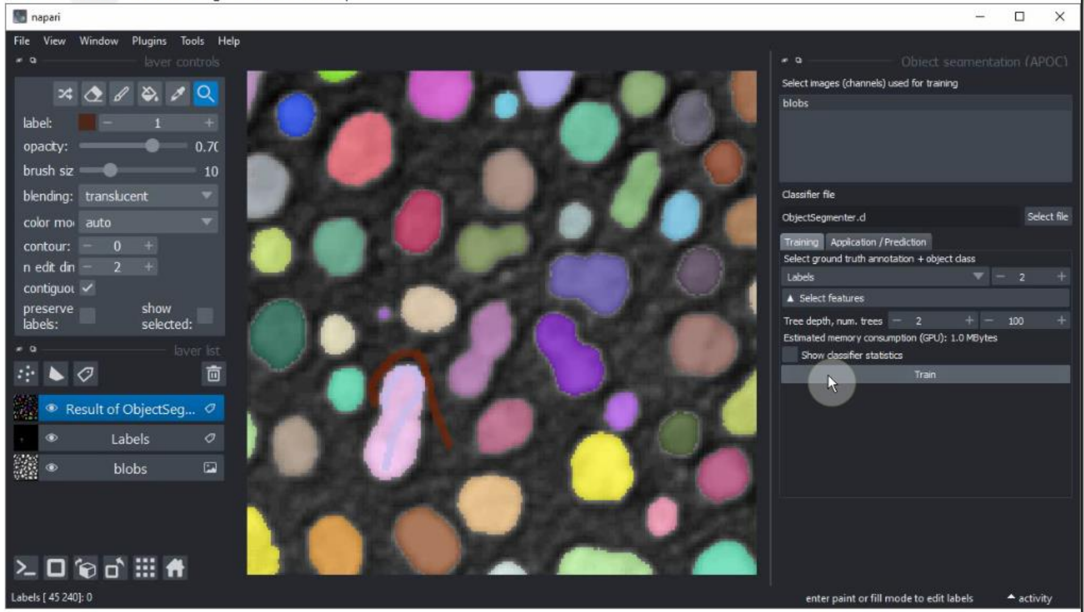
Afterwards, start up Napari:

```
napari
```

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



Click on `Train`. A label image should show up.



If the segmentation works well, consider backing up the `ObjectSegmenter.cl` file that has been saved. If you didn't change the file location before training, it will be located in the folder from where you started napari on the command line.

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:

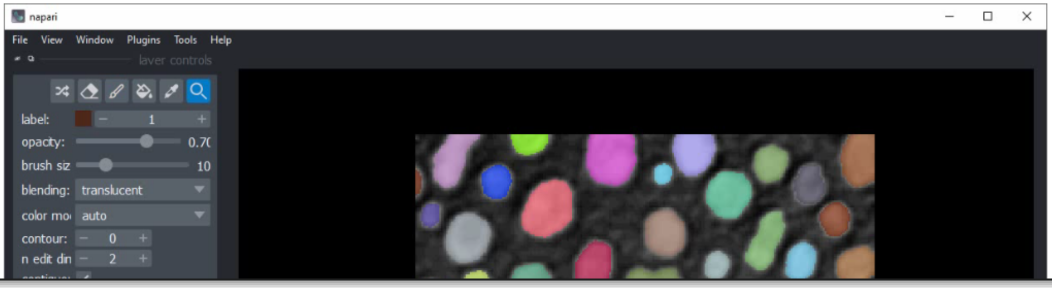
```
napari
```

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

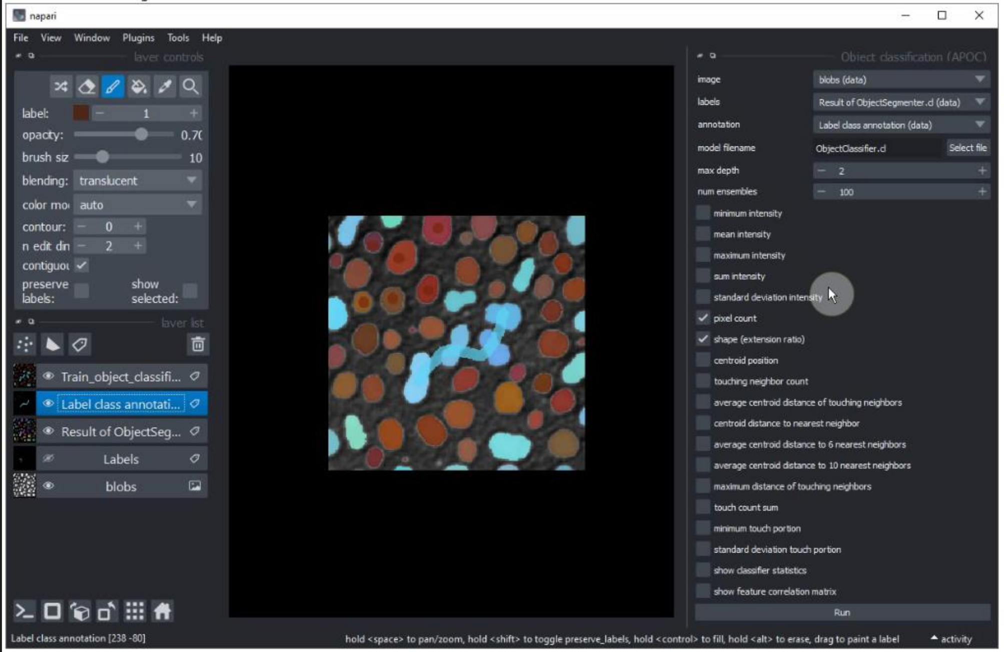
We furthermore need a label image. You can create it using the pixel classifier trained earlier or using the menu `Tools > Segmentation / labeling > Gauss-Otsu Labeling (clesperanto)`.

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.

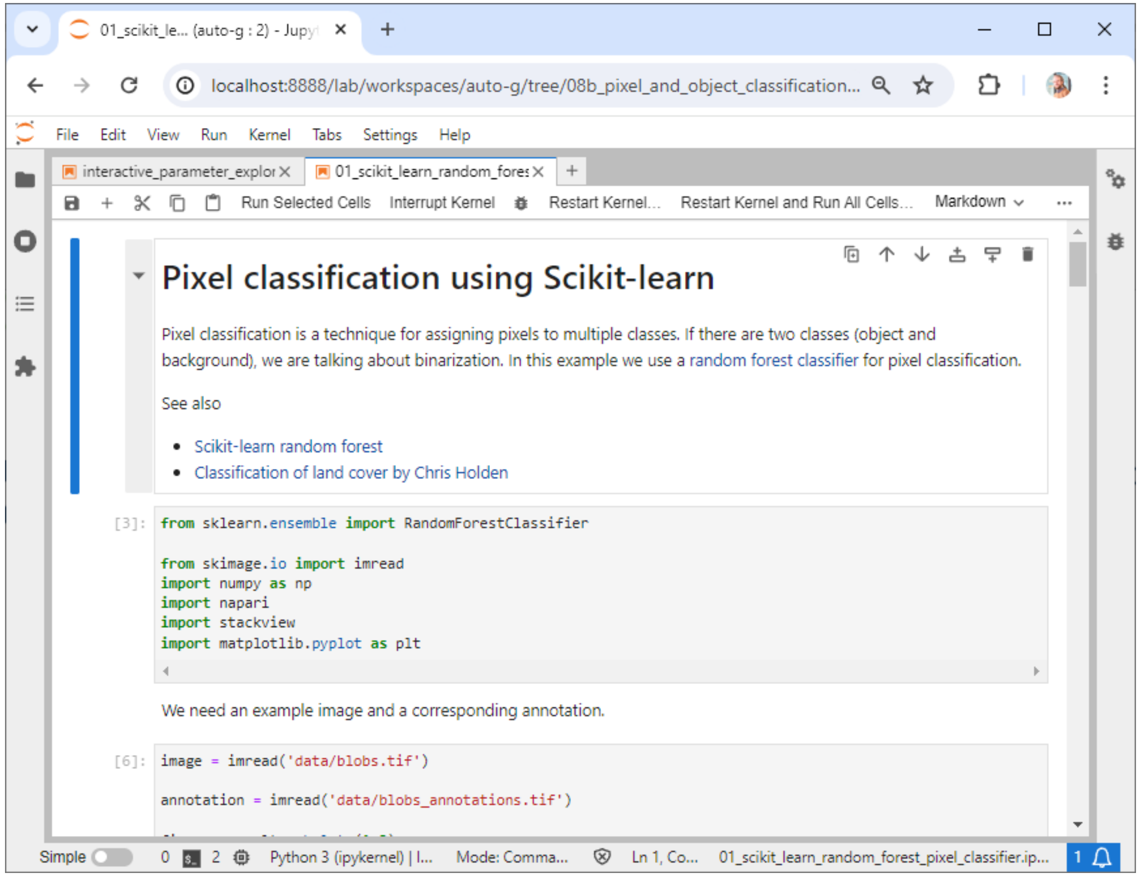
Extra exercise

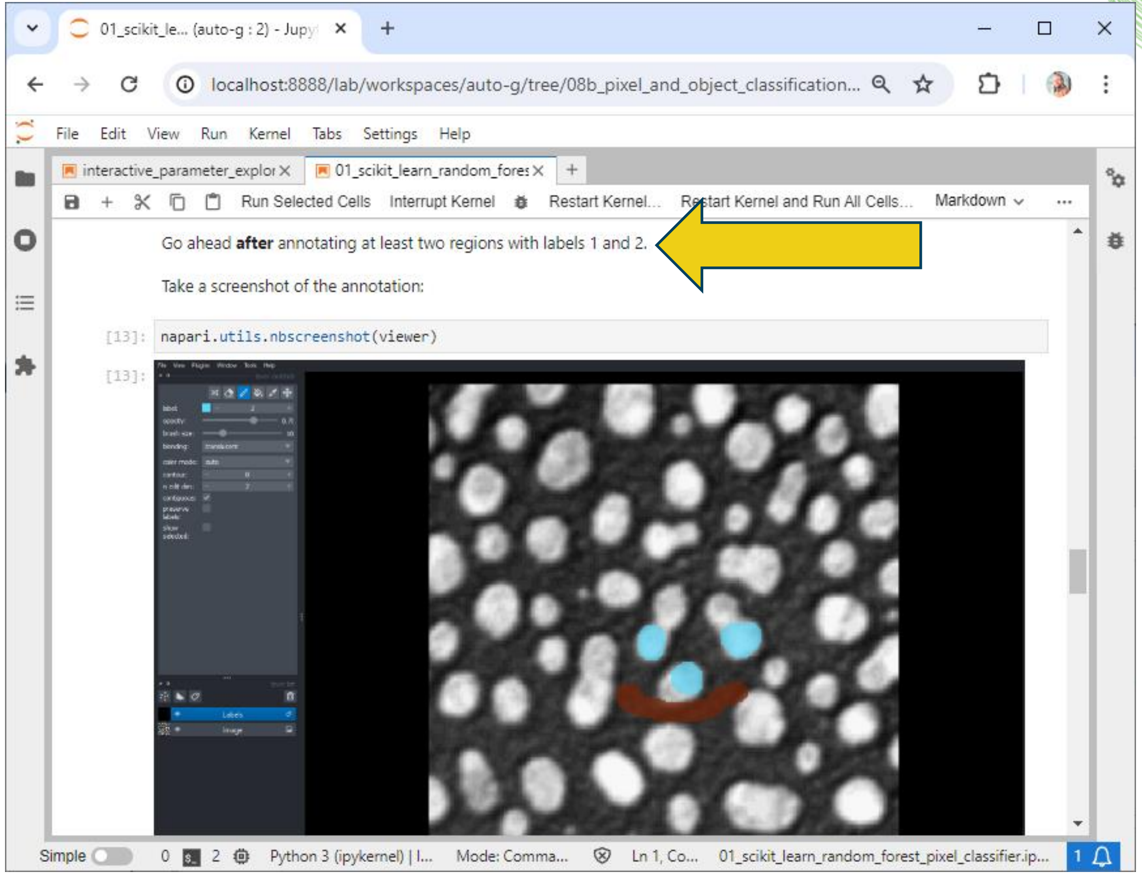
Retrain the classifier so that it can differentiate three different classes:

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

Supervised machine learning using Python

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





Configuring Random Forest Classifiers