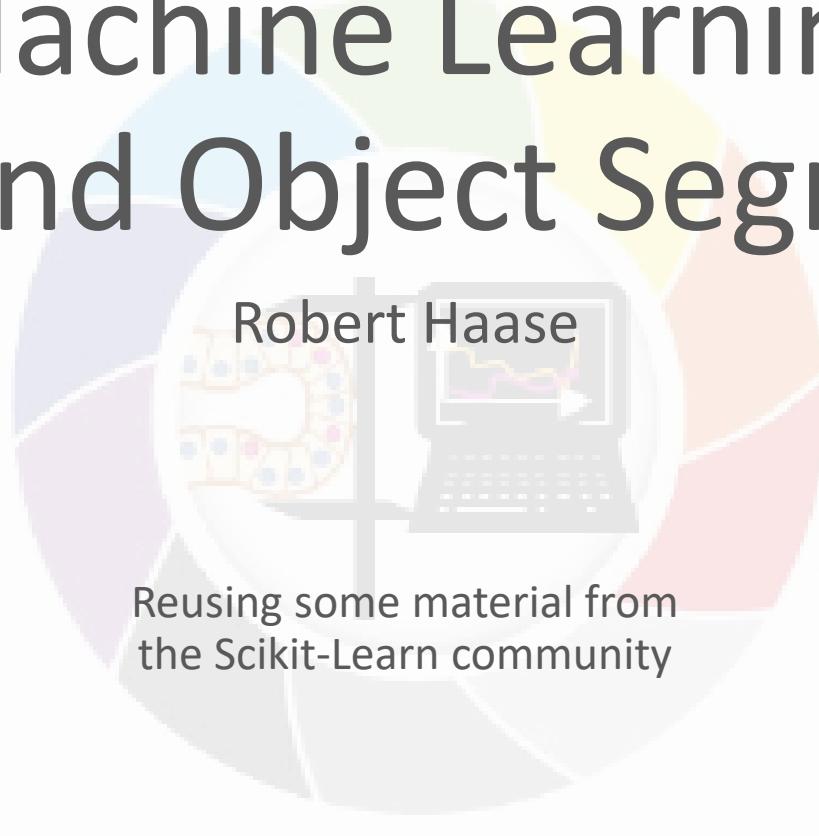


Machine Learning for Pixel and Object Segmentation



Overview

- Machine learning for Pixel and Object Classification
 - Random Forest Classifiers
- Python
 - scikit-learn / napari
 - Accelerated pixel and object classification (APOC)

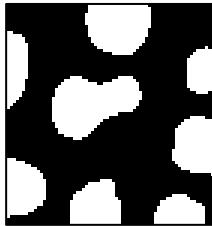
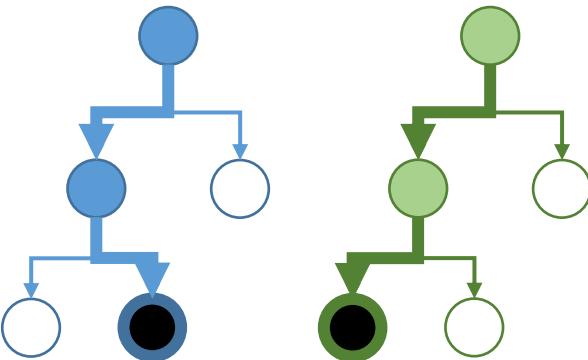
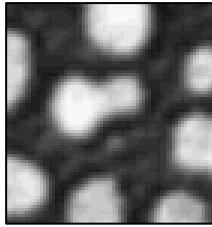


Image segmentation using thresholding

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

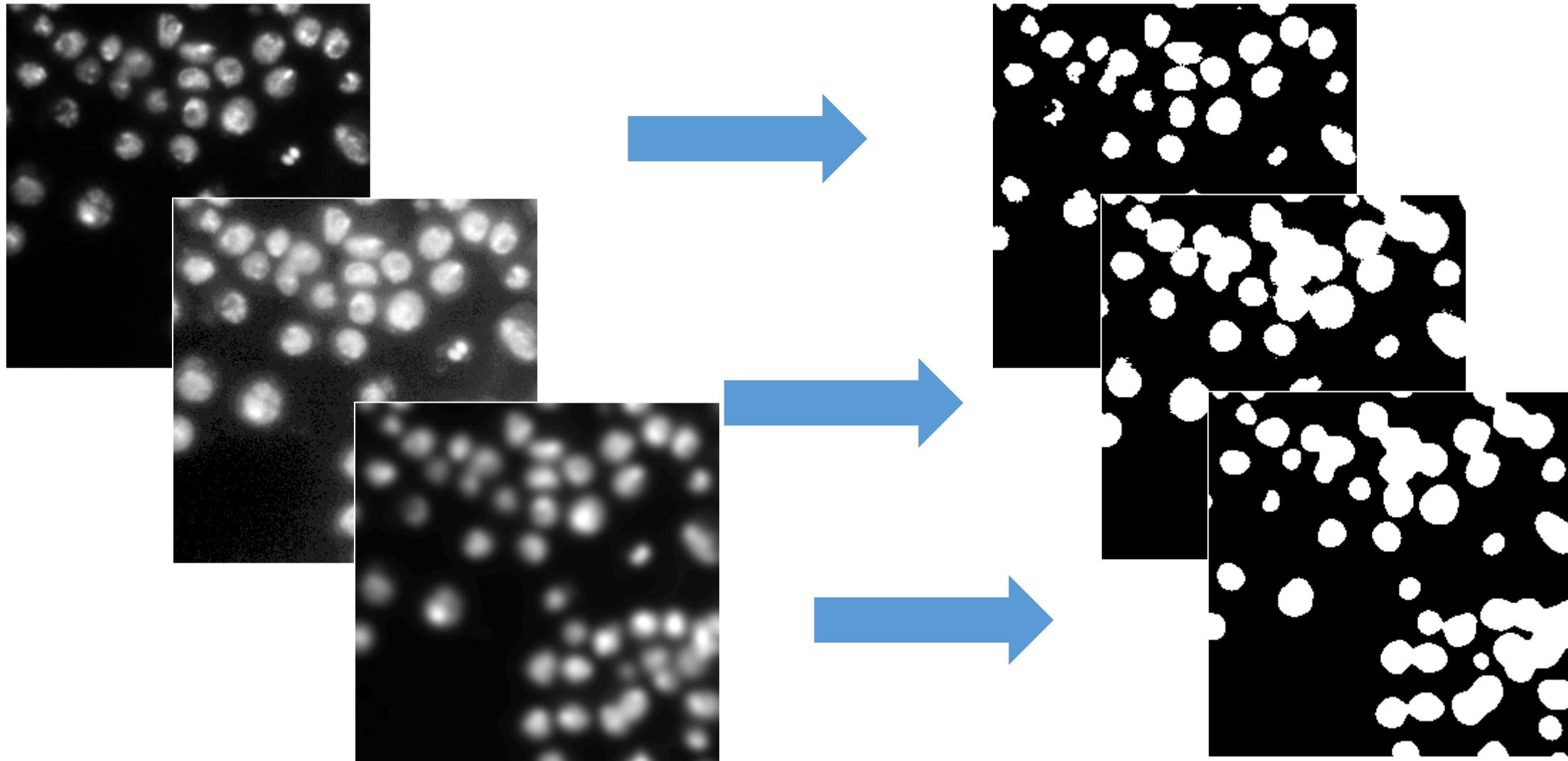


Image segmentation using thresholding

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

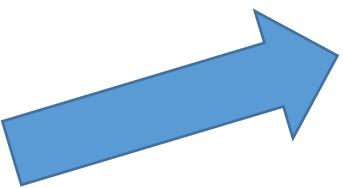
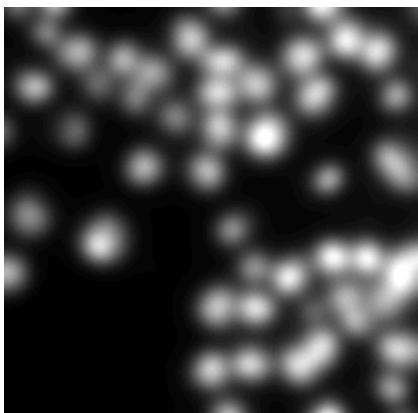
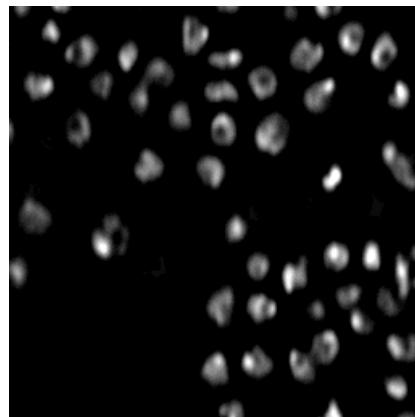
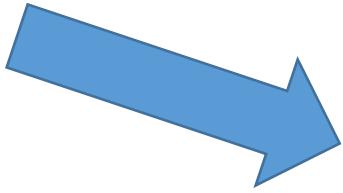
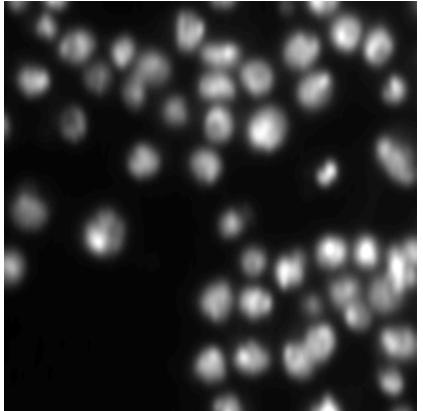
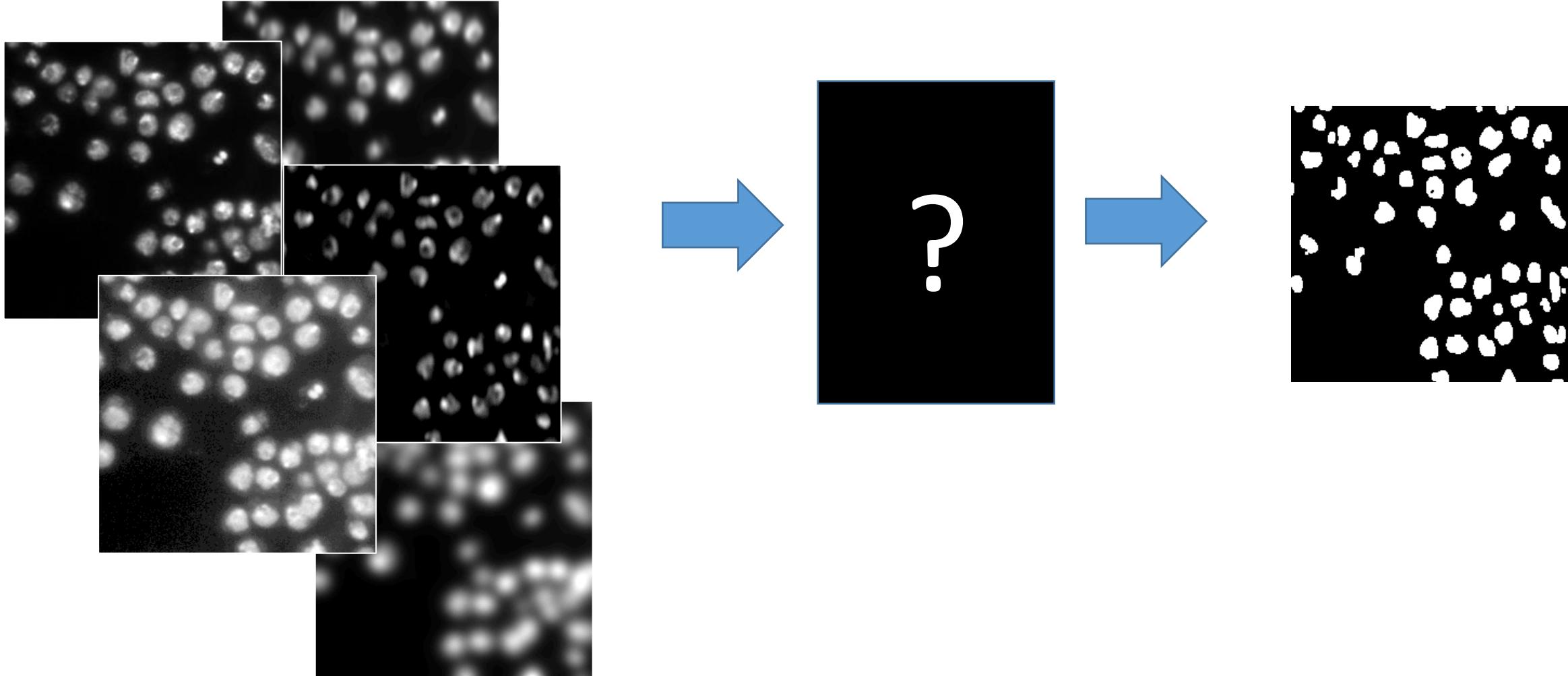


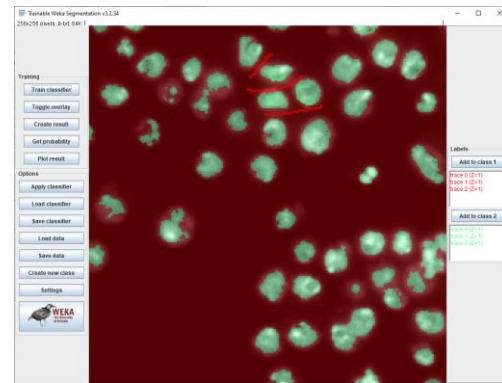
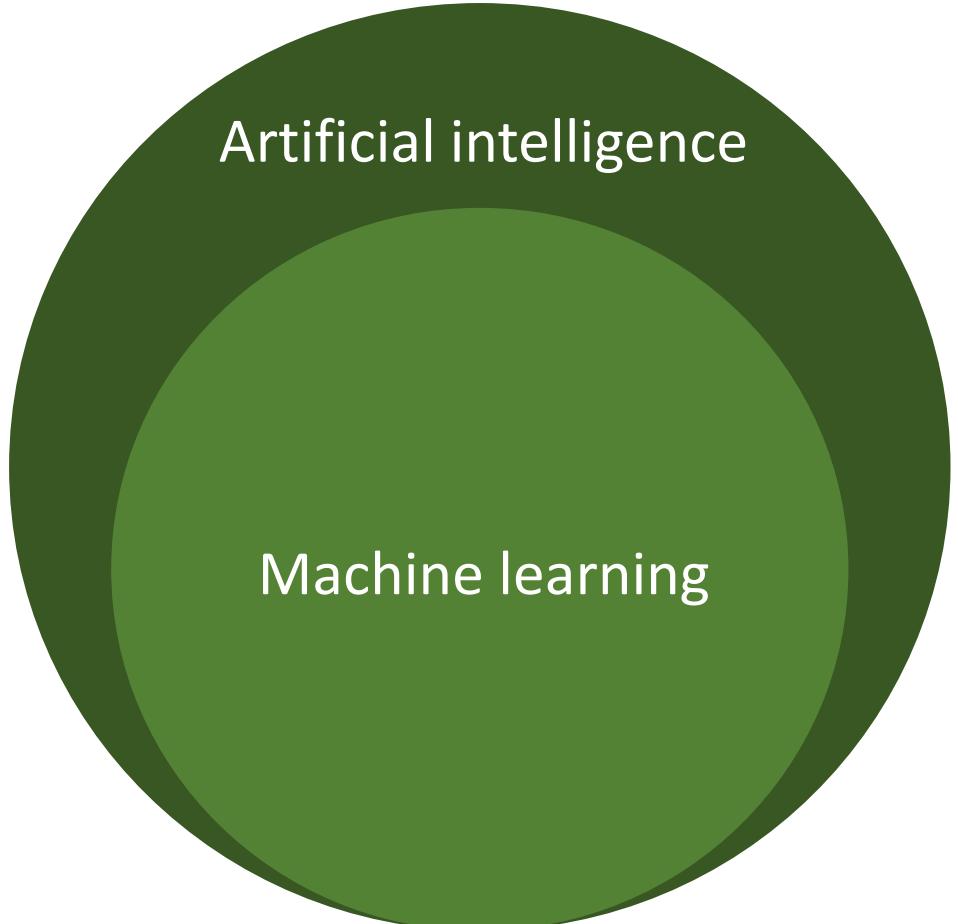
Image segmentation using thresholding

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

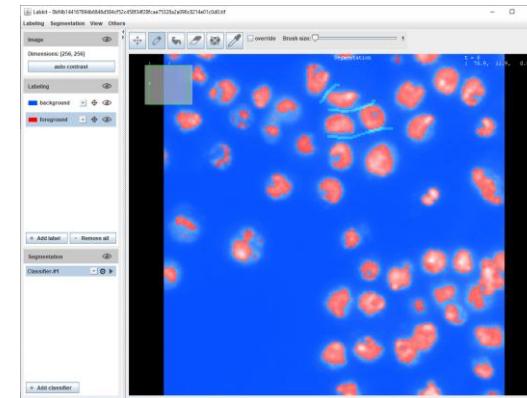


Machine learning

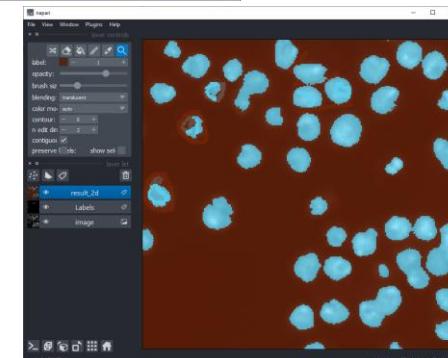
- A research field in computer science
- Finds more and more applications, also in life sciences.



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



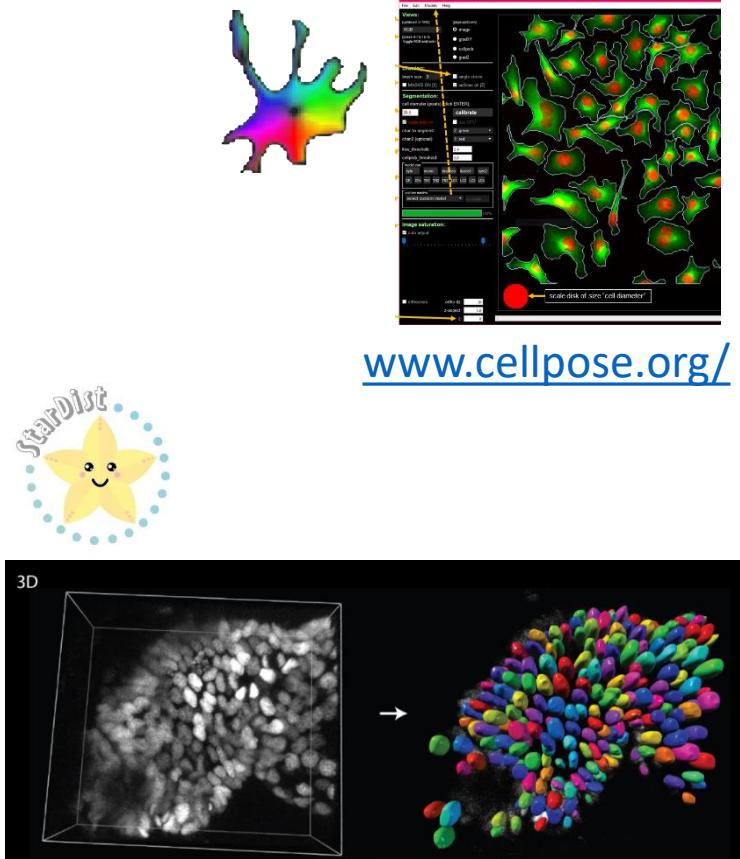
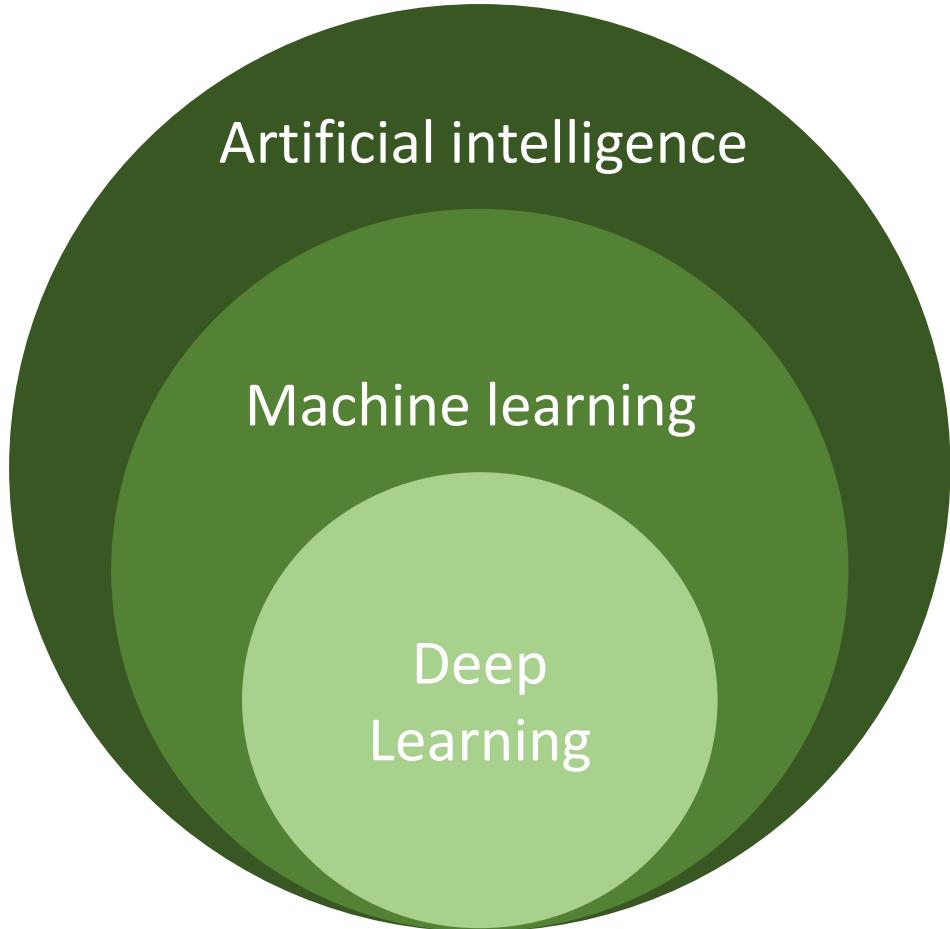
LabKit
<https://imagej.net/plugins/labkit/>



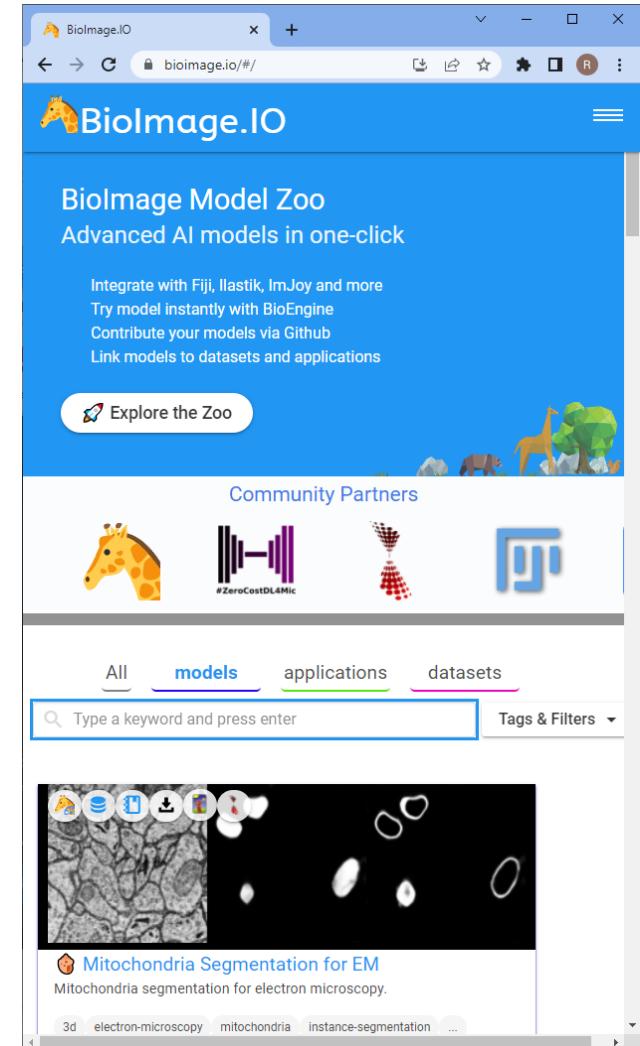
Python /
scikit-learn /
napari /
apoc

Machine learning

- A research field in computer science
- Finds more and more applications, also in life sciences.



<https://github.com/cellpose/cellpose>



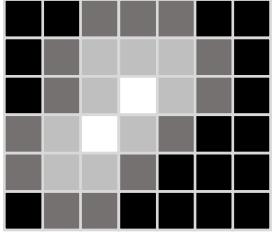
<https://bioimage.io/>

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

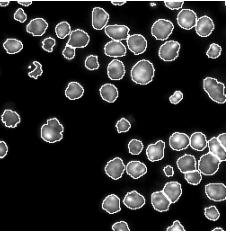
Machine learning

- Automatic construction of predictive models from given data

Pixels,



Objects,

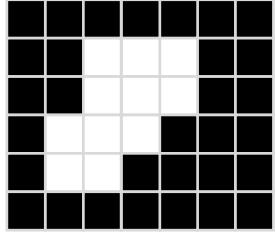


Images, Audio, Text, Measurements, ...



Annotated raw data, usually generated by humans

Dense Segmentation / Binarization



Object classification

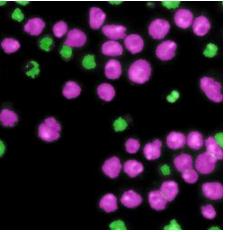
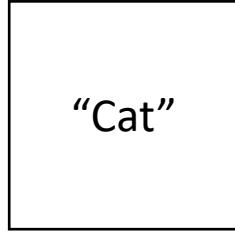
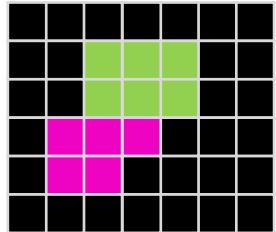


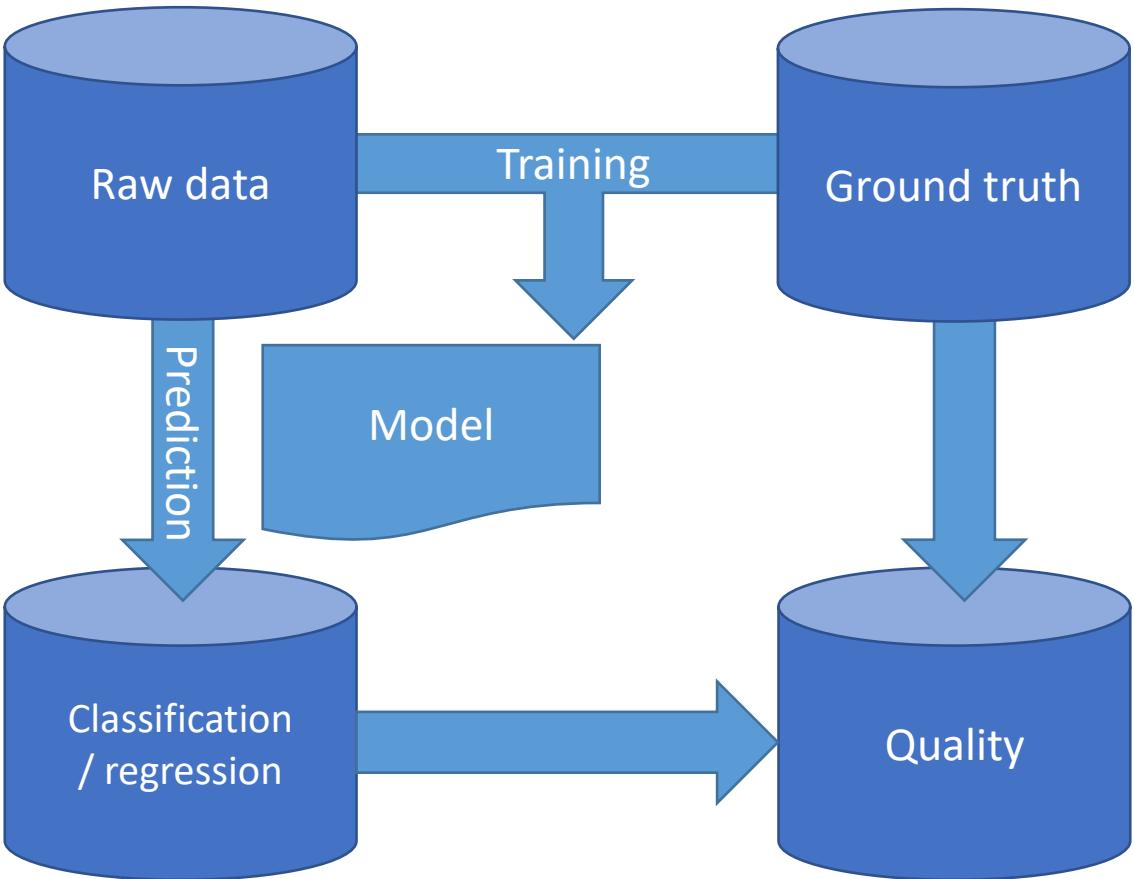
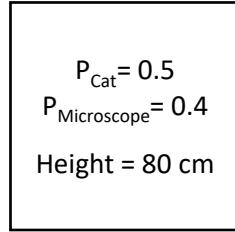
Image classification
“Cat”



Instance segmentation

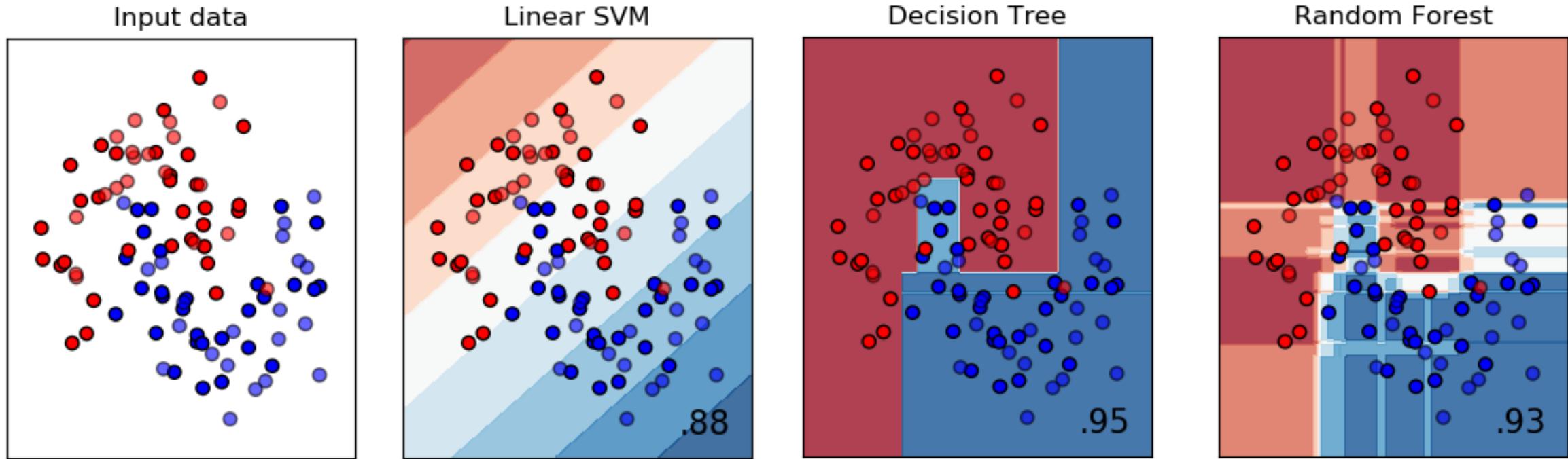


Cont. quantity



Precision,
Recall

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



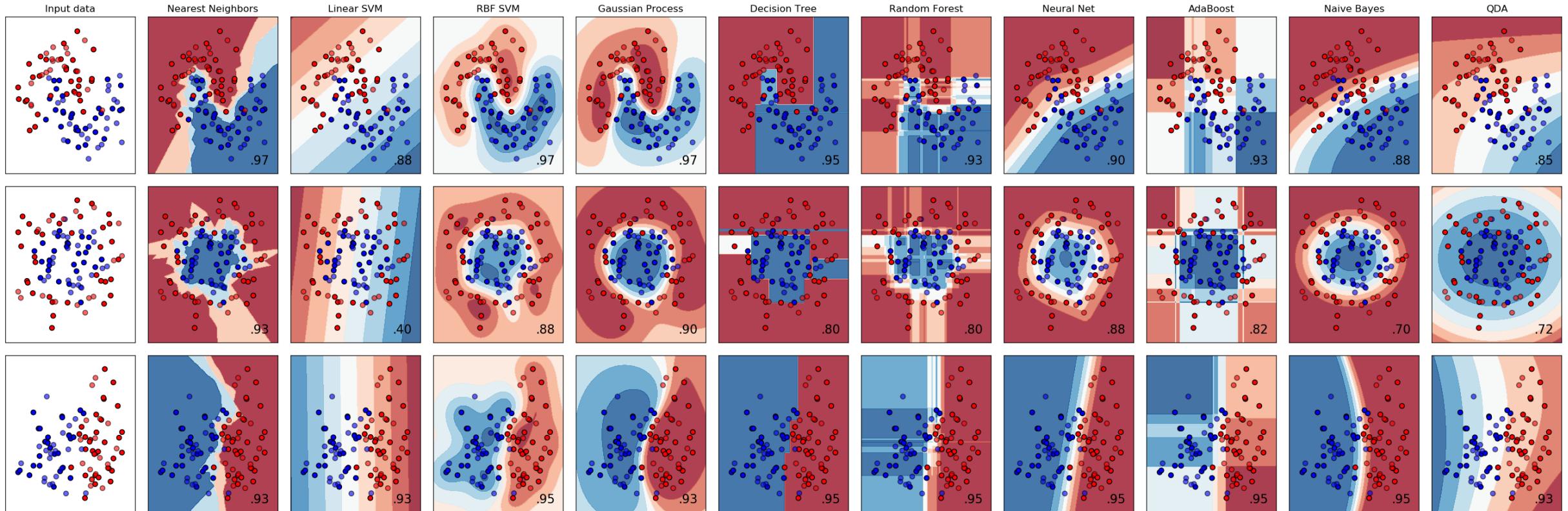
Adapted from https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

August 2023

© 2007 - 2019, scikit-learn developers (BSD License).

Approaches

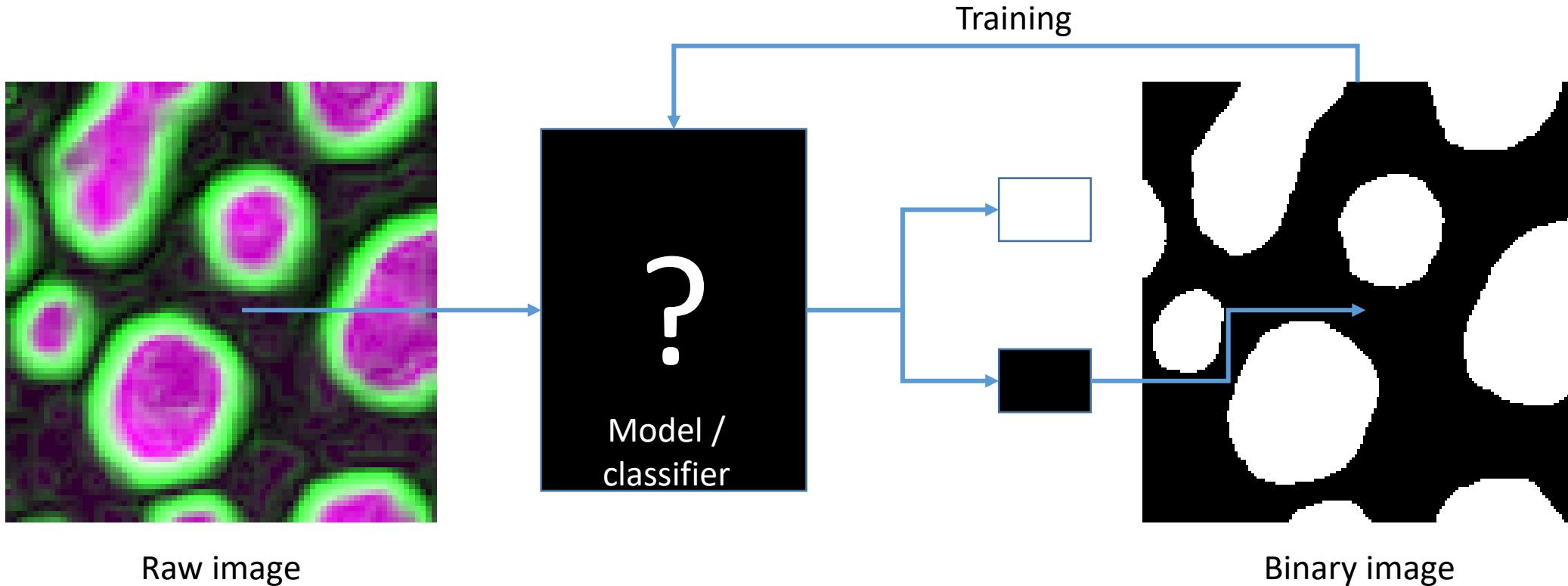
- The right approach depends on data, computational resources and desired quality



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

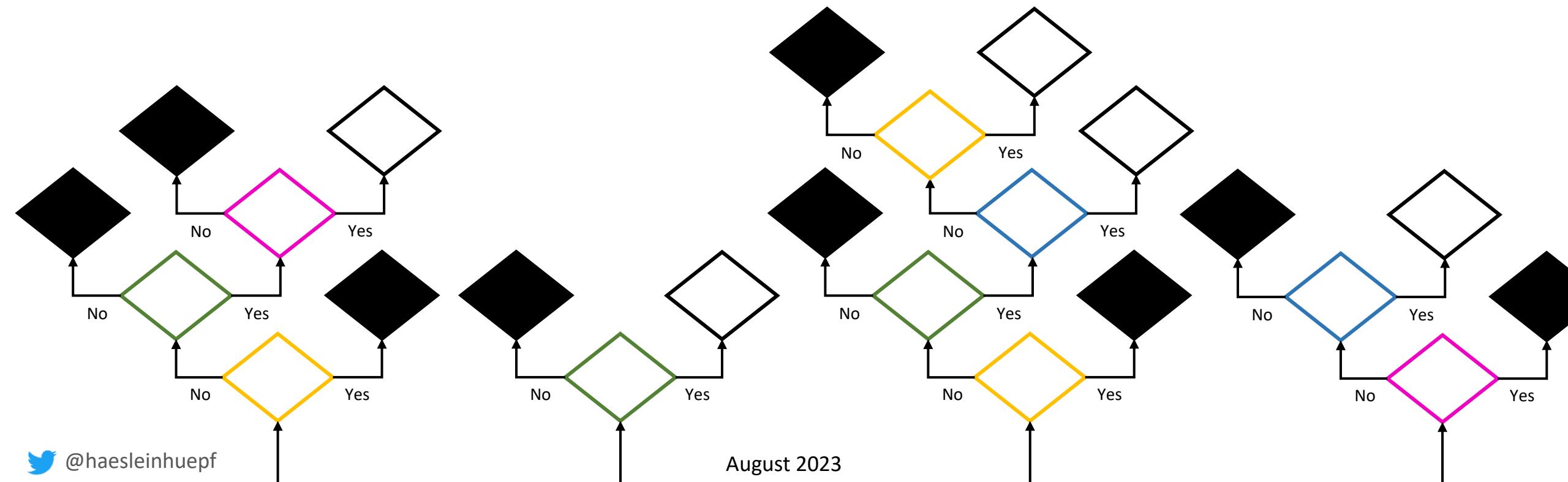
Machine learning for image segmentation

- *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



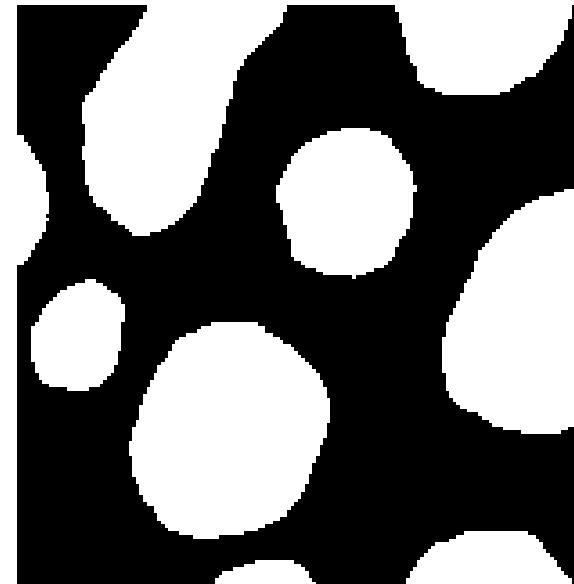
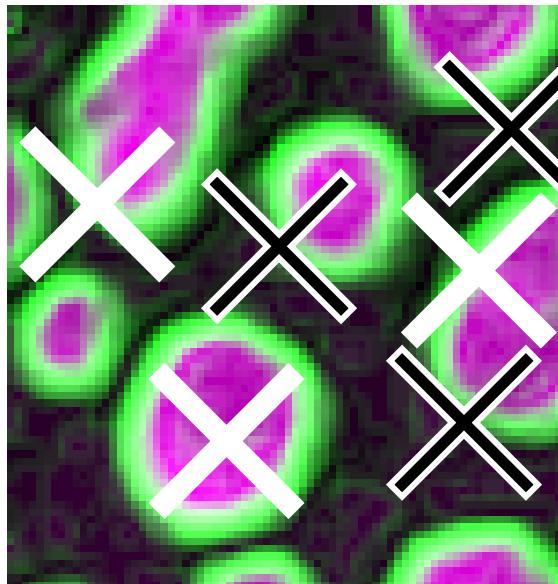
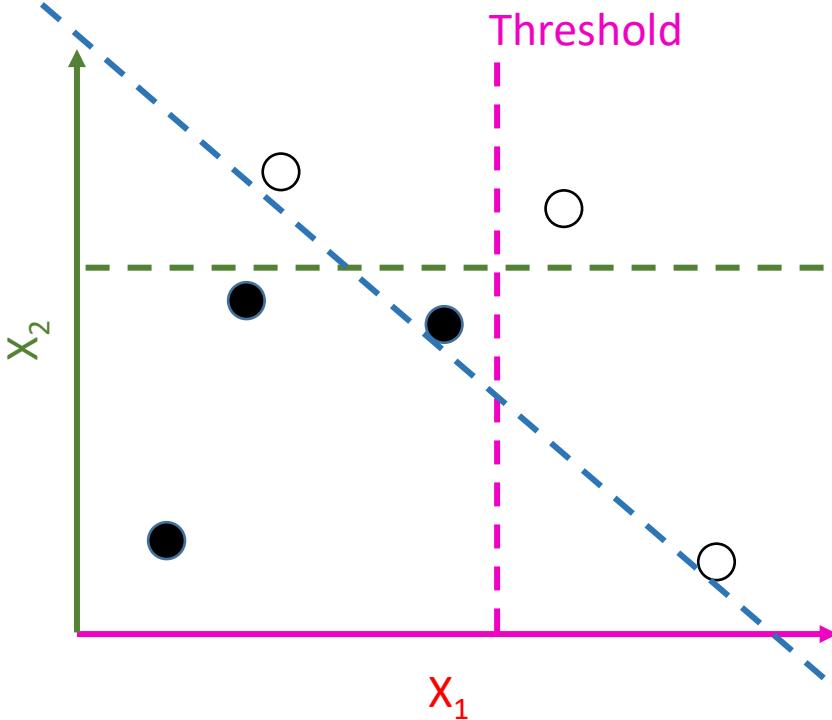
Random forest based image segmentation

- 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

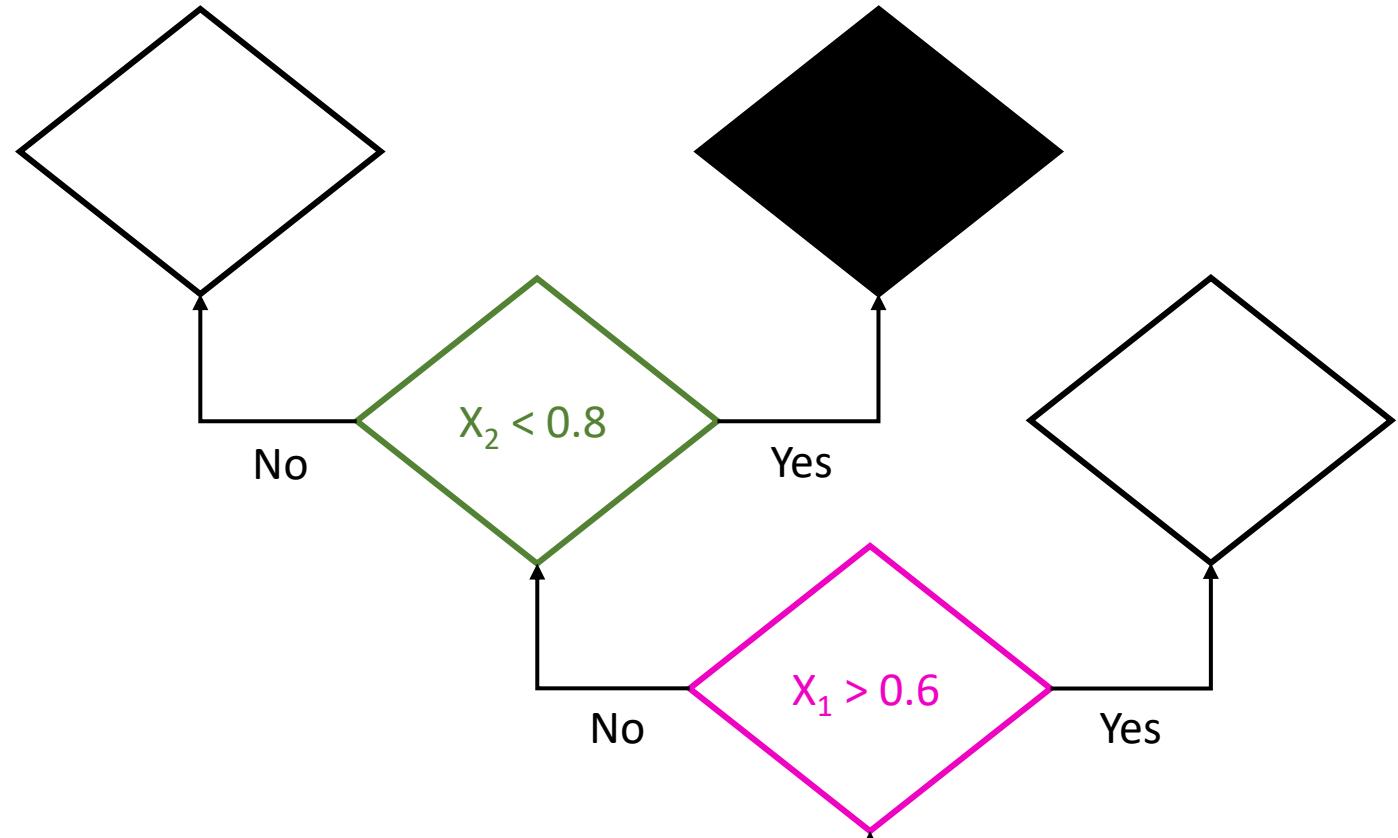
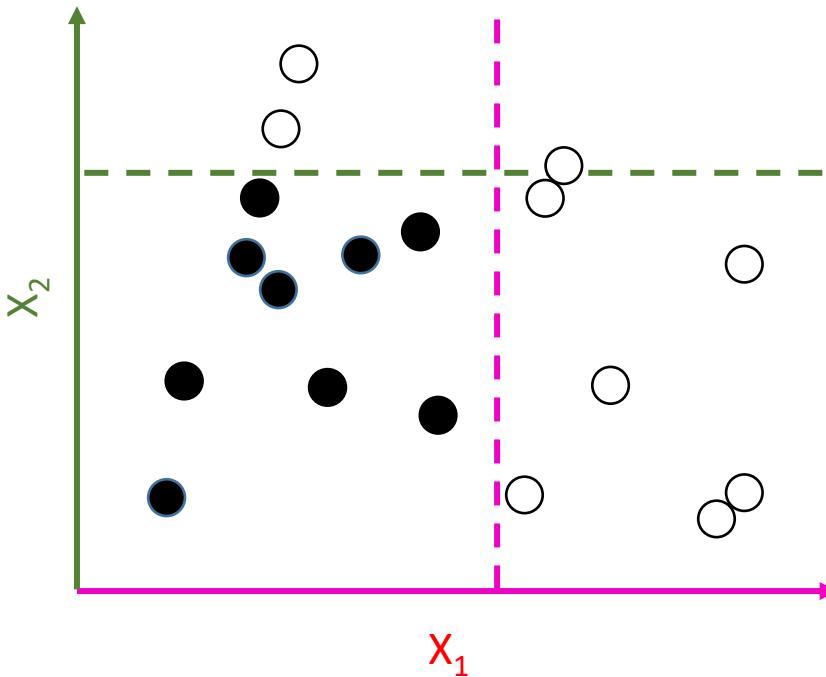
- For efficient processing, we randomly *sample* our data set
 - Individual pixels, their intensity and their classification



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

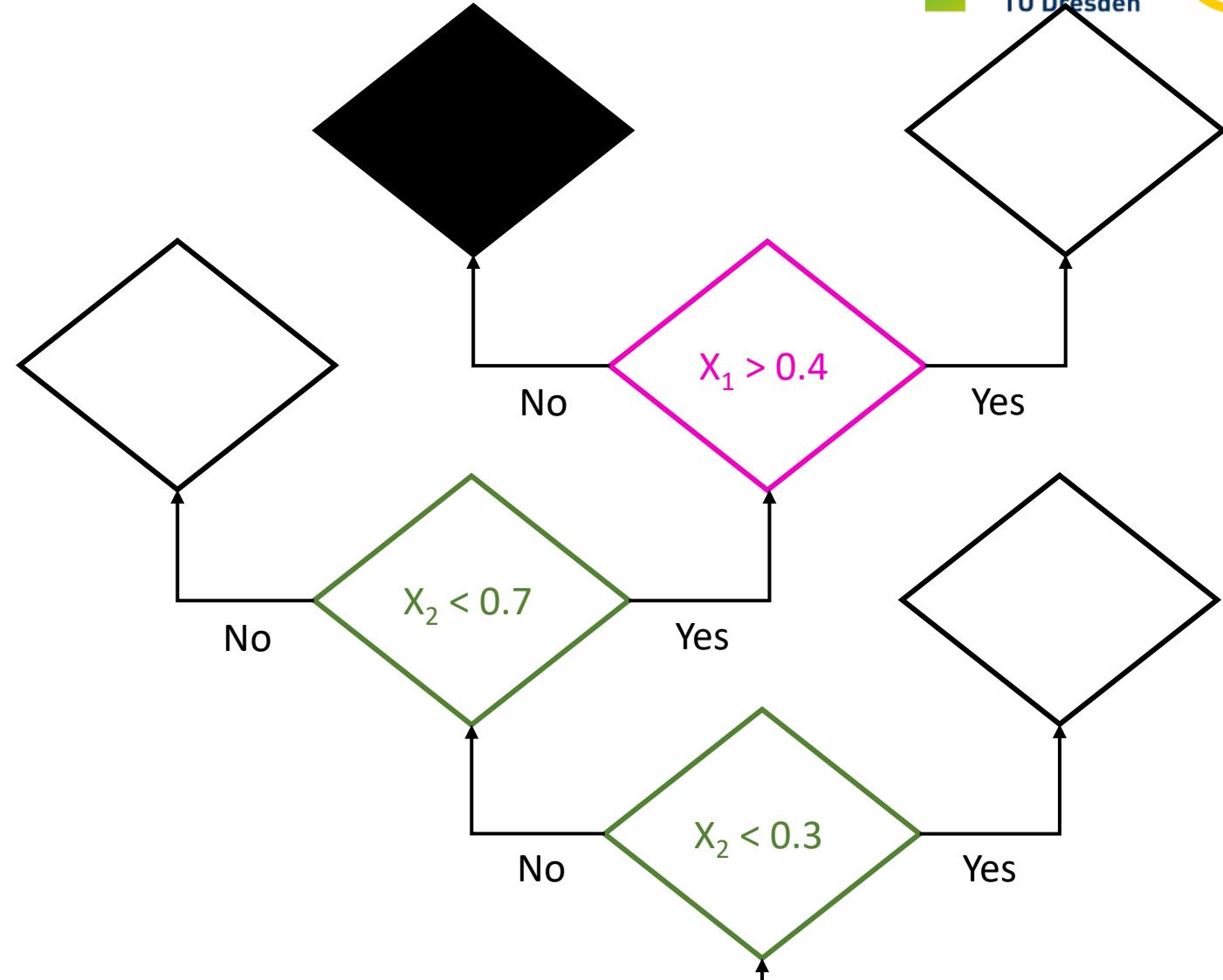
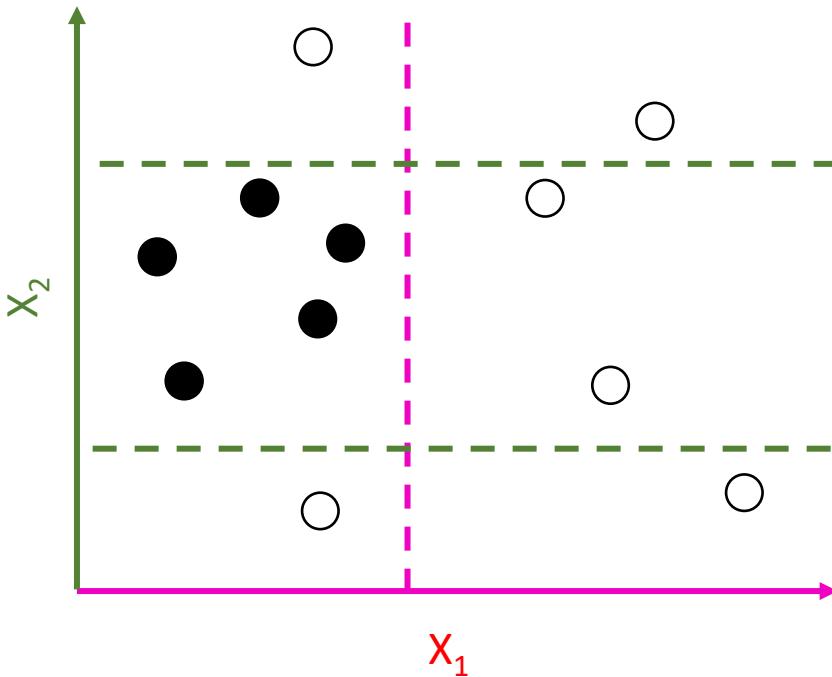
Deriving random decision trees

- Decision trees combine several thresholds on several parameters



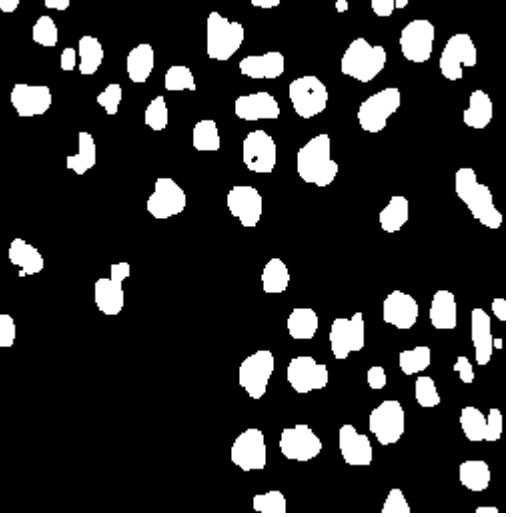
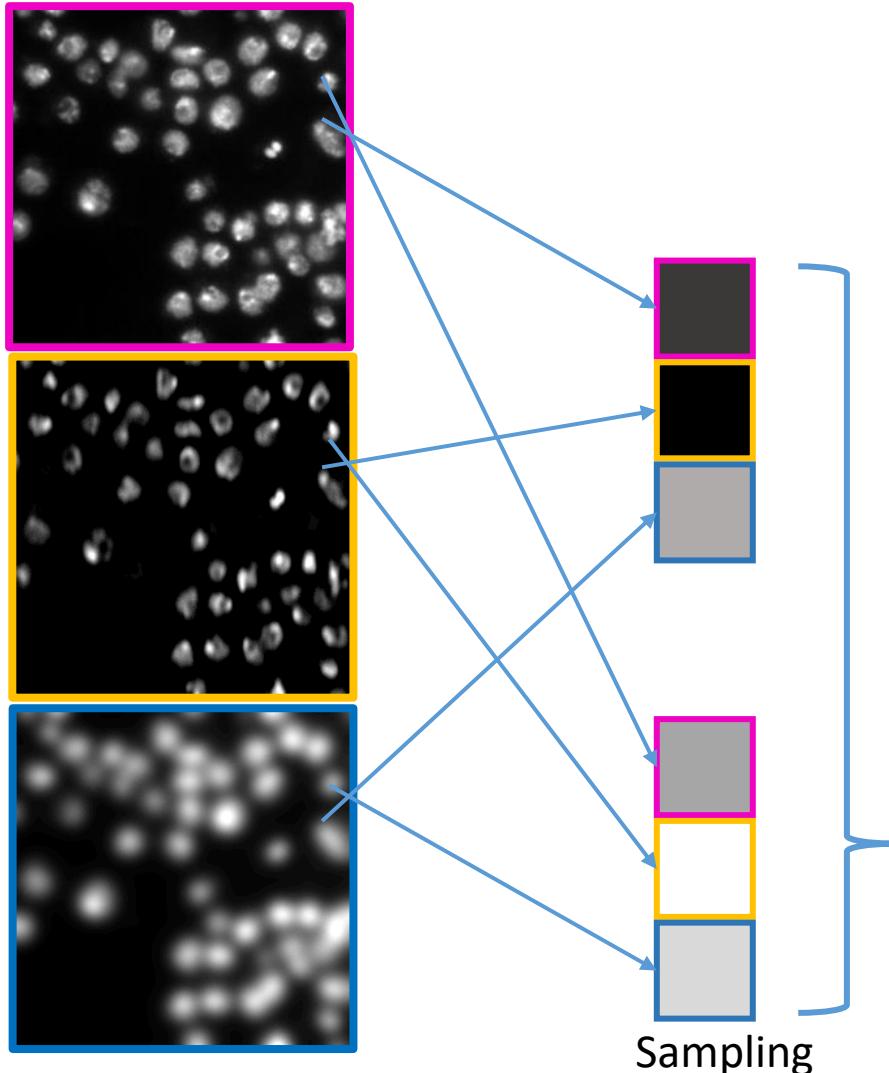
Deriving random decision trees

- Depending on sampling, the decision trees are different

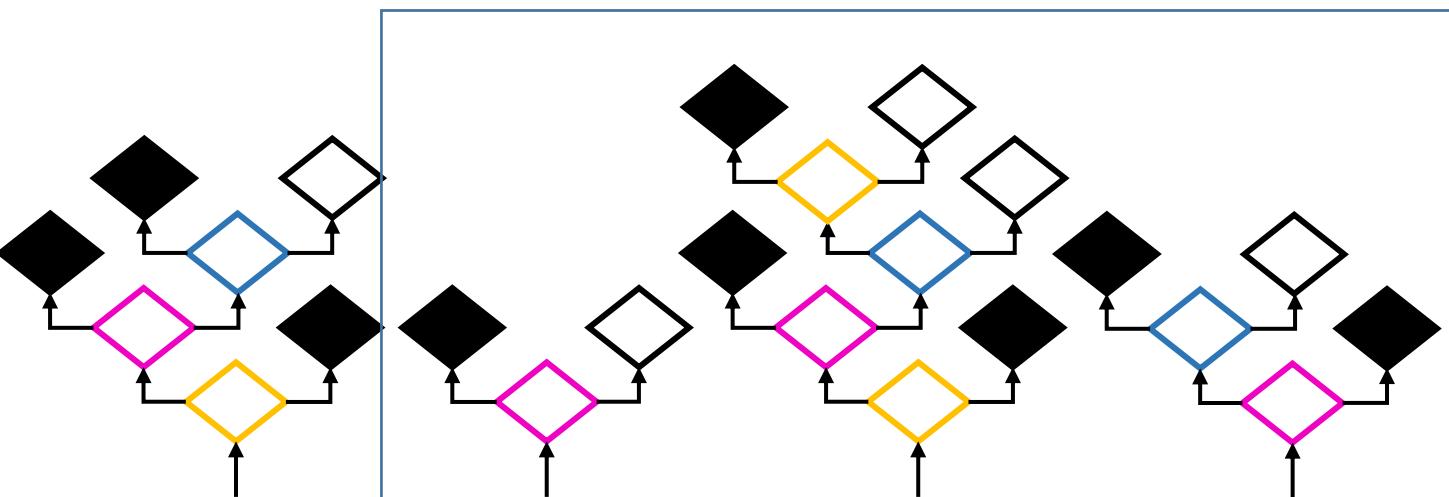


Random Forest Pixel Classifiers

- By comparing performance of individual decision trees, good ones can be selected, bad ones excluded.

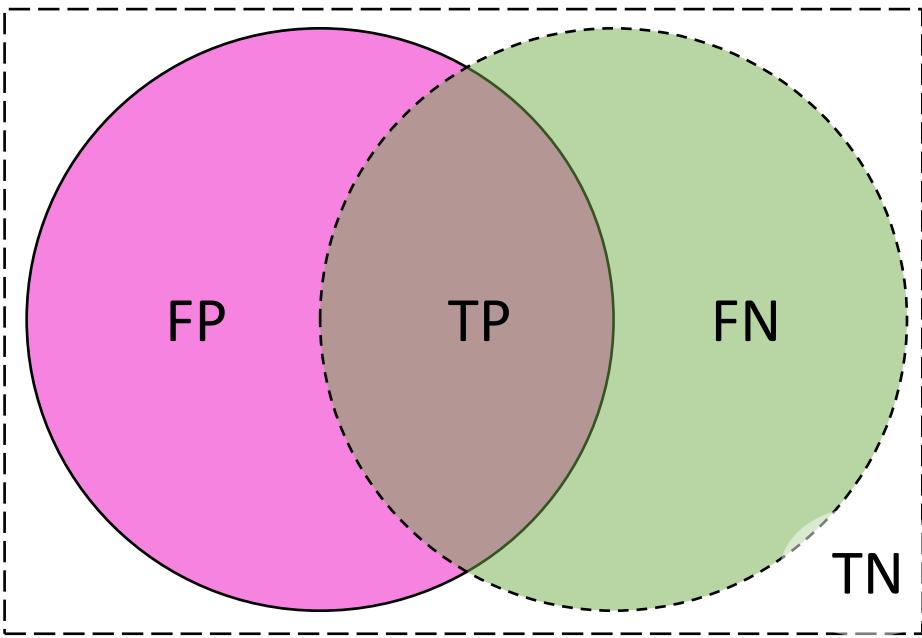


Selection



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



A Prediction A
B Reference B (ground truth)
ROI Region of interest
TP True-positive
FN False-negative
FP False-positive
TN True-negative

Overlap
(a.k.a. Jaccard index) $\frac{TP}{TP + FN + FP}$

How much do A and B overlap?

Precision $\frac{TP}{TP + FP}$

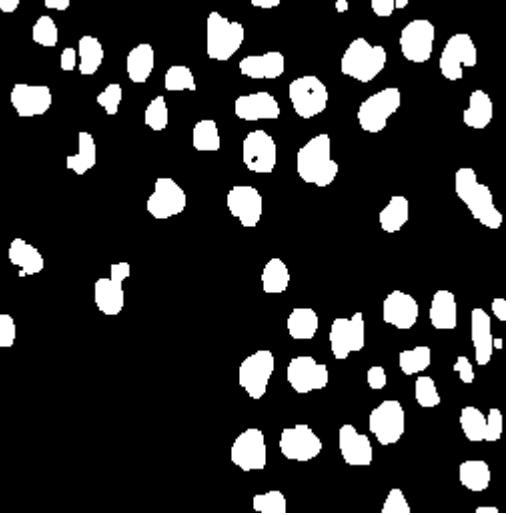
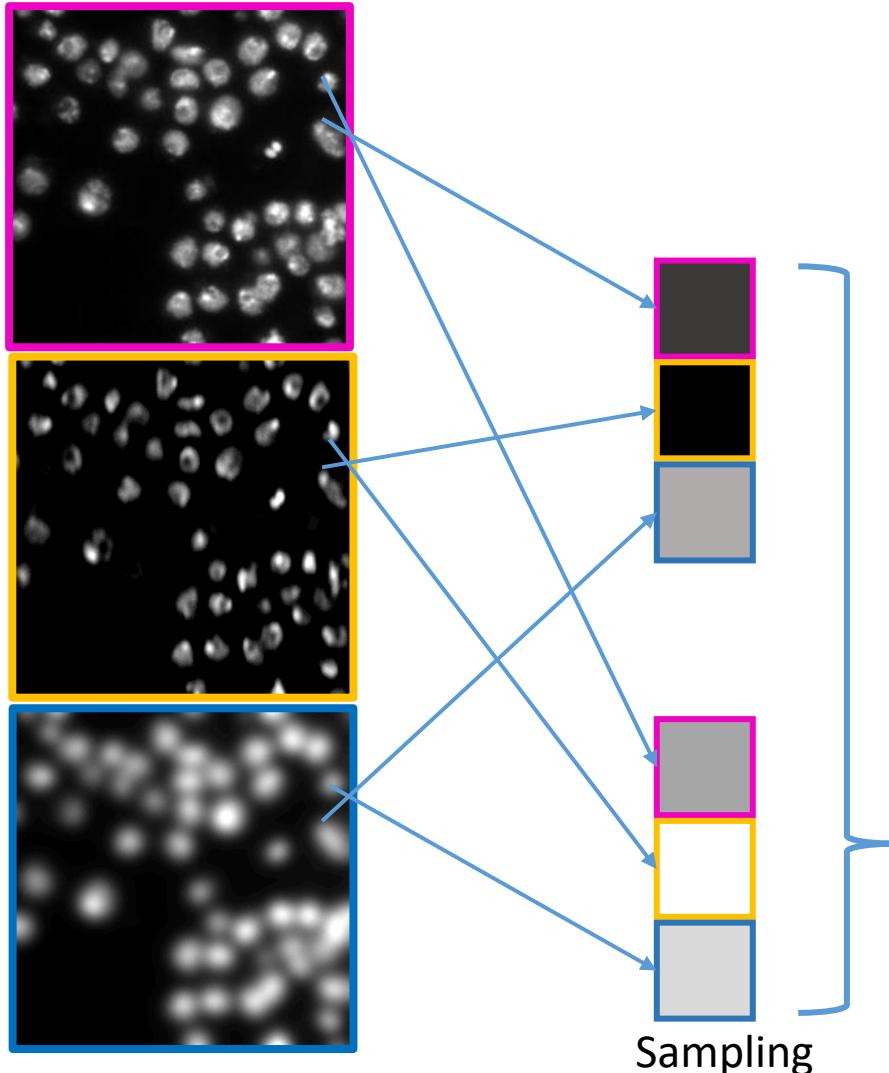
What fraction of points that were predicted as positives were really positive?

Recall
(a.k.a. sensitivity) $\frac{TP}{TP + FN}$

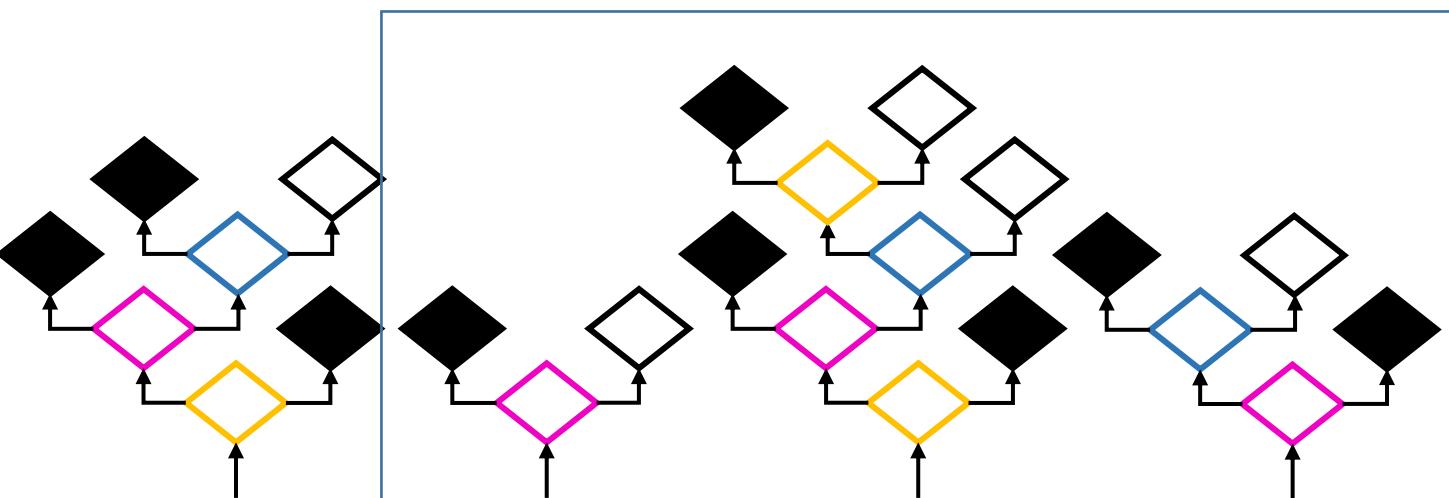
What fraction of positives points were predicted as positives?

Random Forest Pixel Classifiers

- By comparing performance of individual decision trees, good ones can be selected, bad ones excluded.

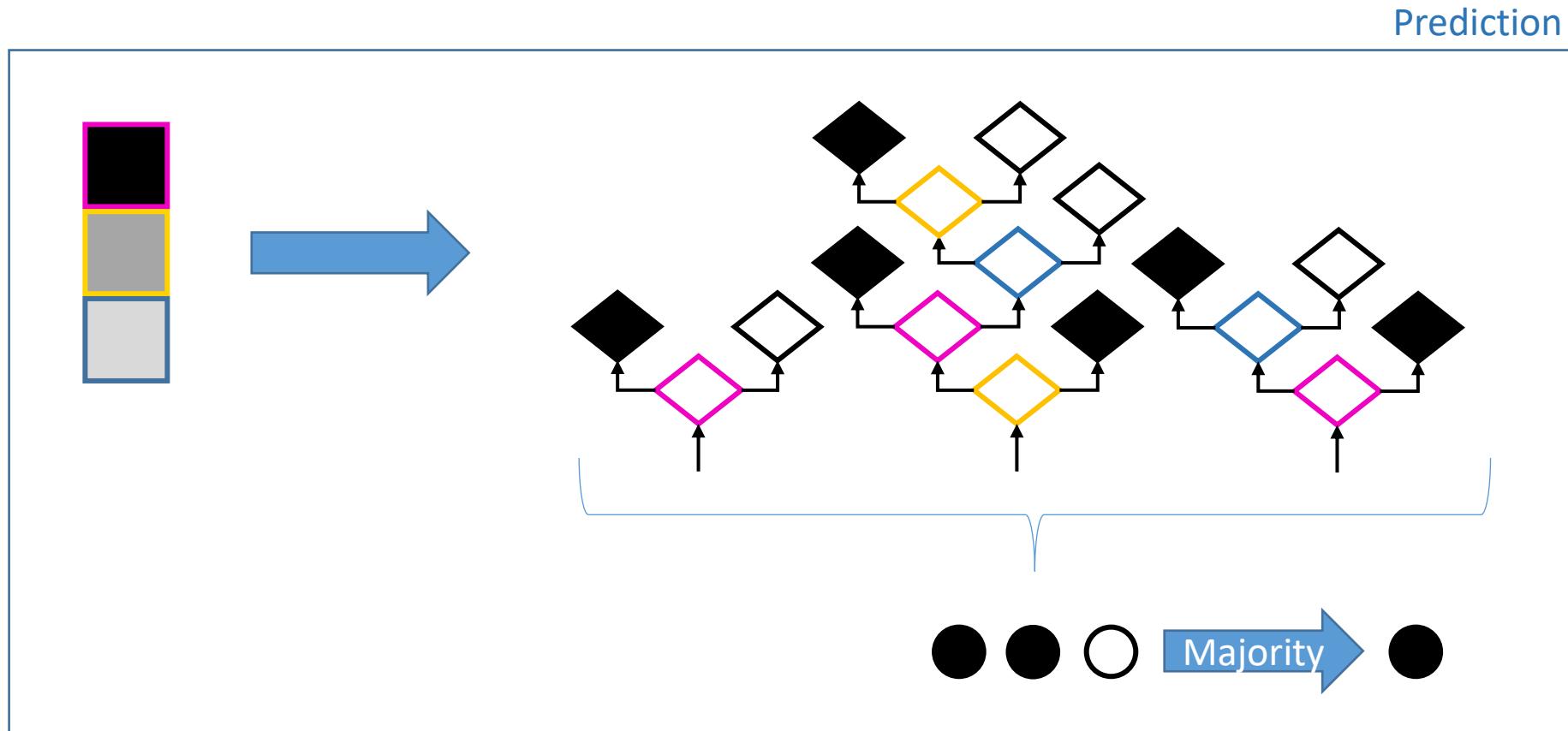


Selection



Random Forest Pixel Classifiers

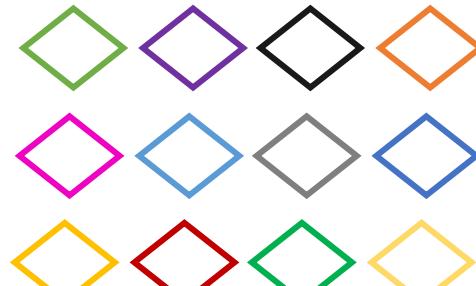
- Combination of individual tree decisions by voting or max / mean



Random Forest Pixel Classifiers

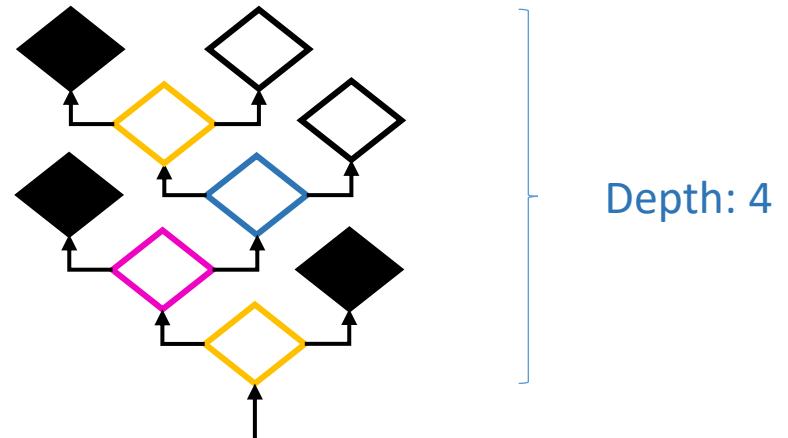
- Typical numbers for pixel classifiers in microscopy

Available features: > 20

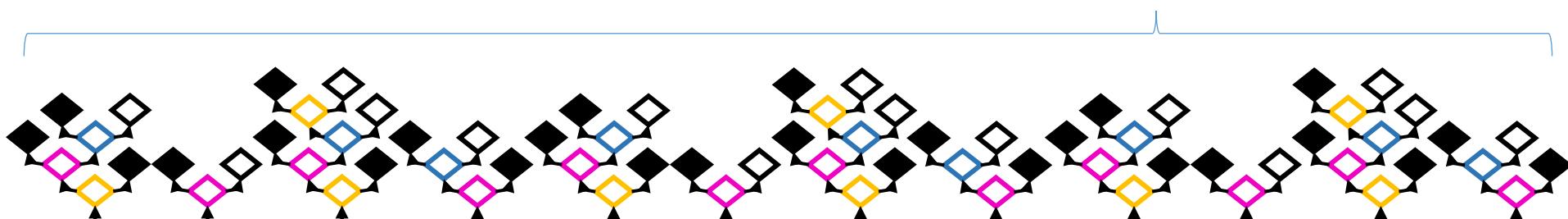


- Gaussian blur image
- DoG image
- LoG image
- Hessian
-

Selected features: <= depth

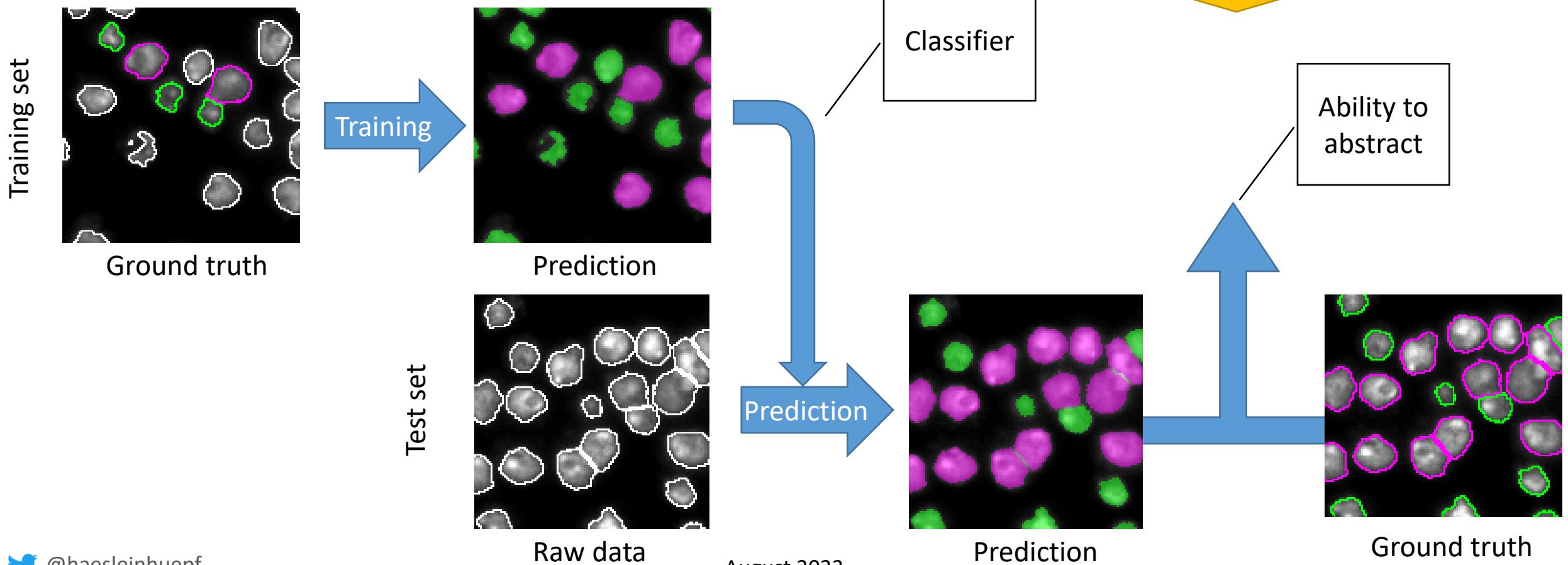


Number of trees: > 100



Model validation

- A good classifier is trained on a hand full of datasets and works on thousands similarly well.
- In order to assess that, we split the ground truth into two set
 - Training set (50%-90% of the available data)
 - Test set (10%-50% of the available data)



Train dataset (e.g. 80% of the data)

- Used for training directly

Validation dataset (10% of the data)

- After every iteration see if the model overfits

Test dataset (10% of the data)

- Final evaluation after training is finished (once)

Underfitting

- A trained model that is not even able to properly process the data it was trained on

Overfitting

- A model that is able to process data it was trained on well
- It processes other data poorly

Underfitting

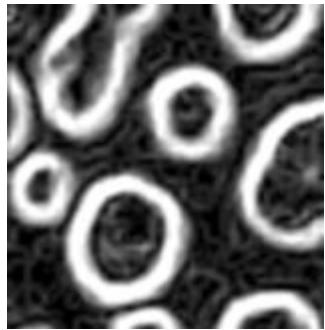
Loss (lower is better)

Overfitting

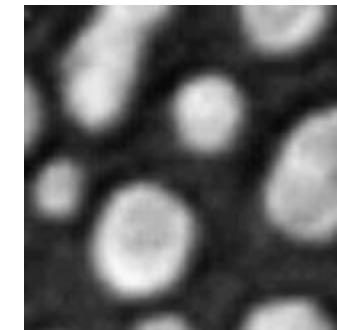
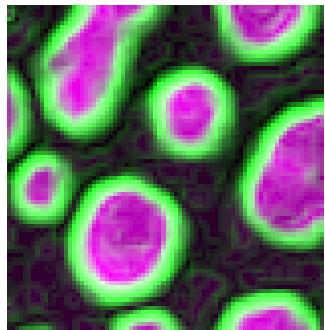
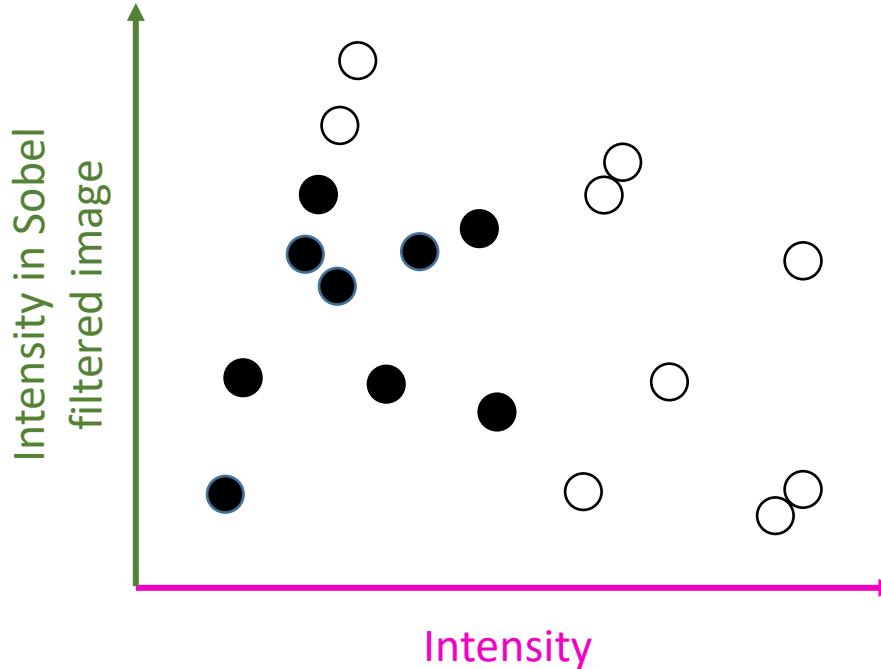
Training duration

<https://towardsdatascience.com/how-to-split-data-into-three-sets-train-validation-and-test-and-why-e50d22d3e54c>

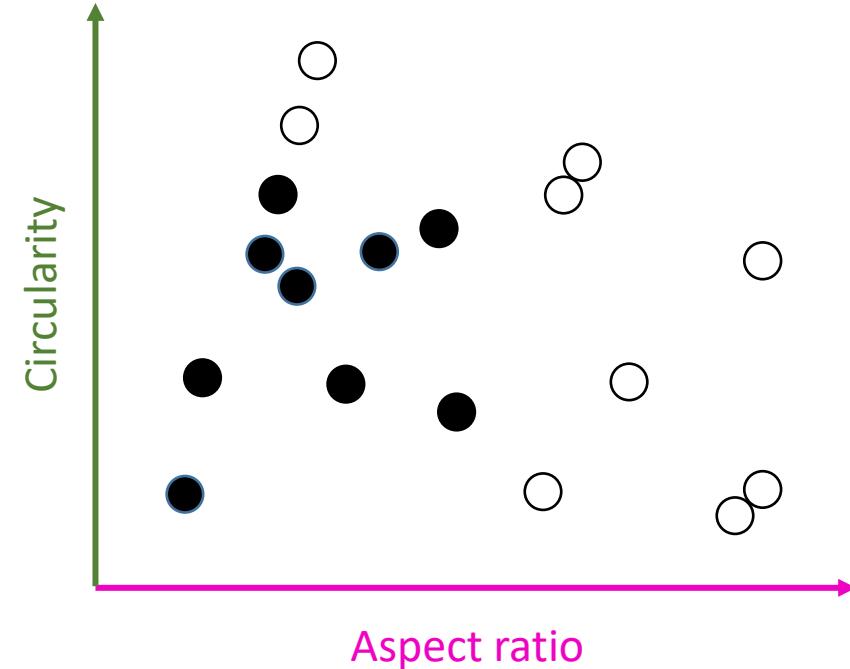
- What if we exchange pixel features with object features?



Pixel classification



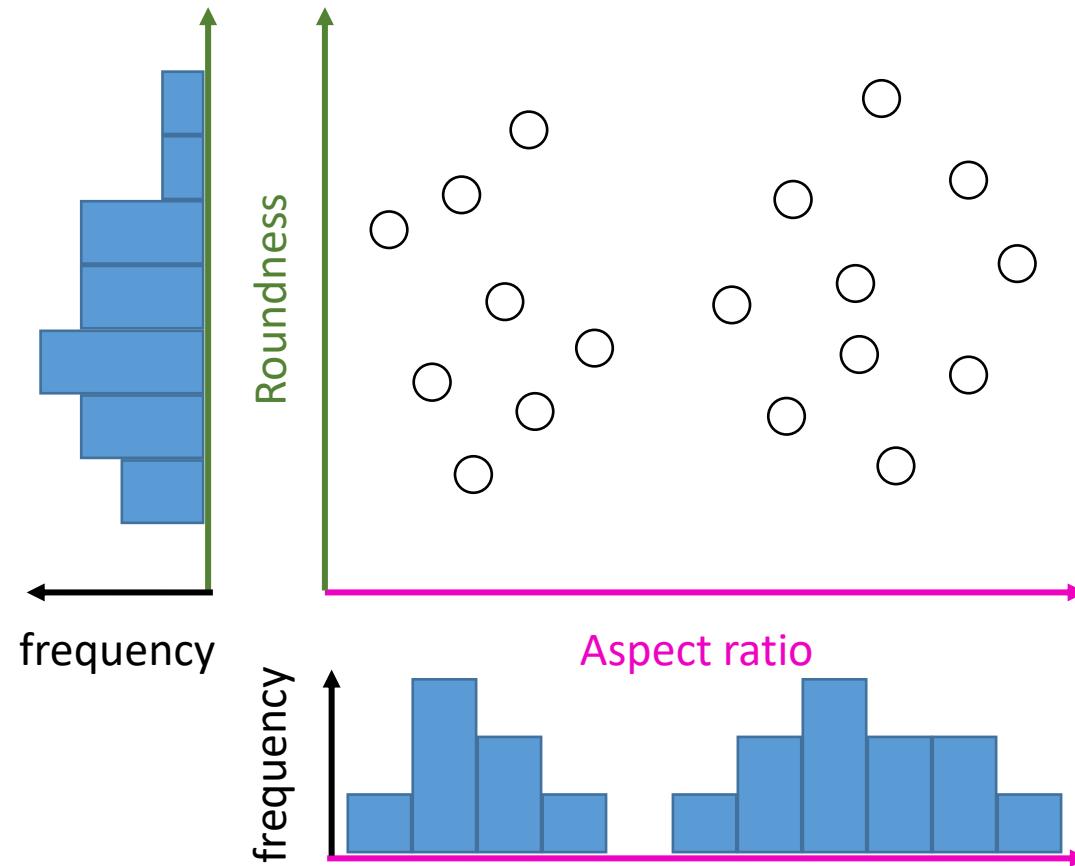
Object classification



- The algorithms work the same but with different
 - Features
 - Number of features
 - Tree / forest parameters
 - Selection criteria

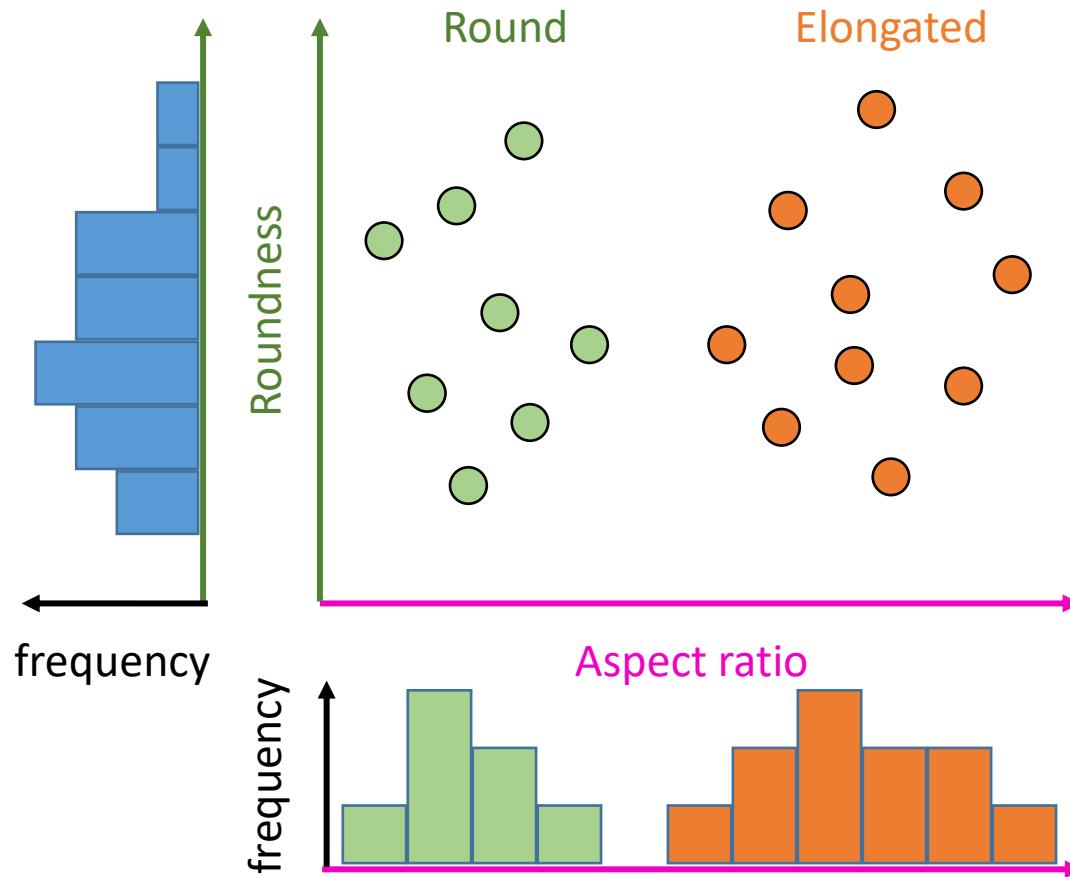
Unsupervised machine learning

- If you don't provide ground truth, the algorithm is *unsupervised*.



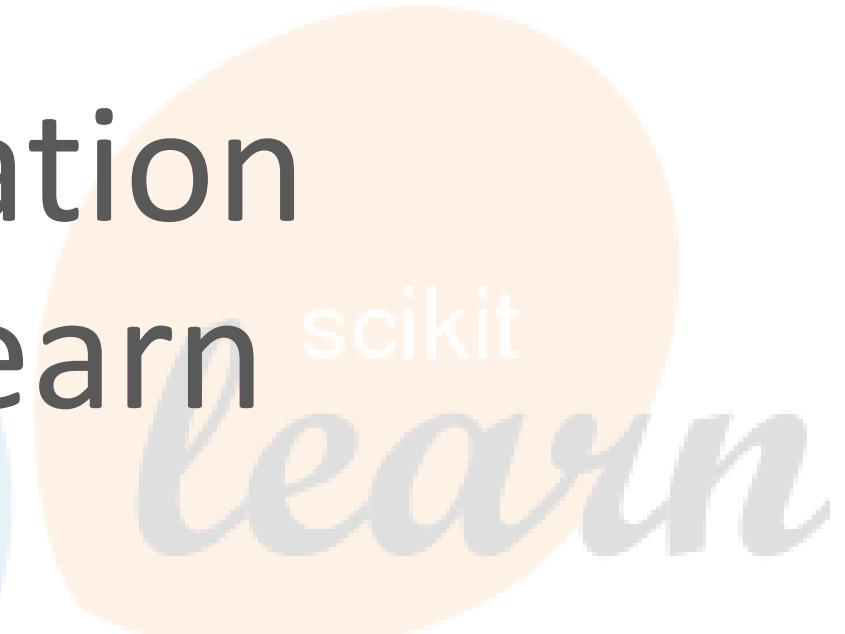
Unsupervised machine learning

- If you don't provide ground truth, the algorithm is unsupervised.
- Nevertheless, algorithms can tell us something about the data



Pixel classification using scikit-learn

Robert Haase



August 2023

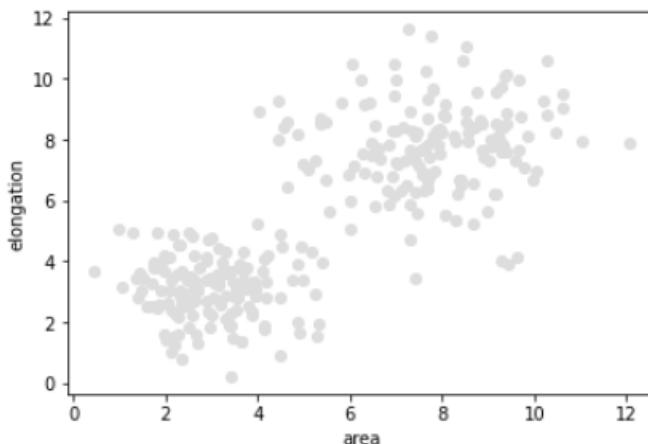
The scikit-learn logo is BSD3 licensed by the scikit-learn developers
https://commons.wikimedia.org/wiki/File:Scikit_learn_logo_small.svg

Tabular object classification

- Classify objects starting from feature vectors (table columns)

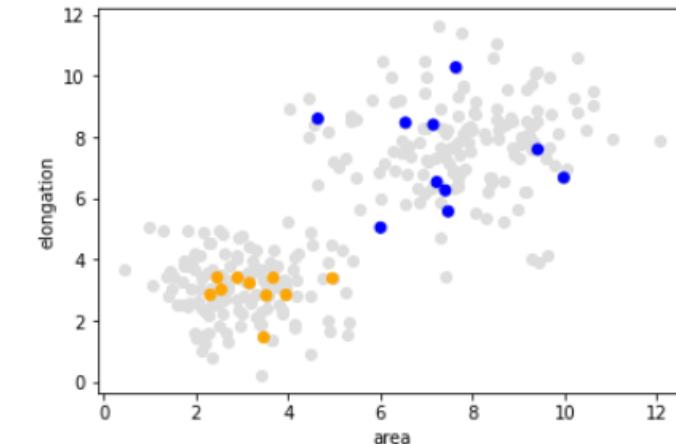
Raw data

	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, 2, 2,
```

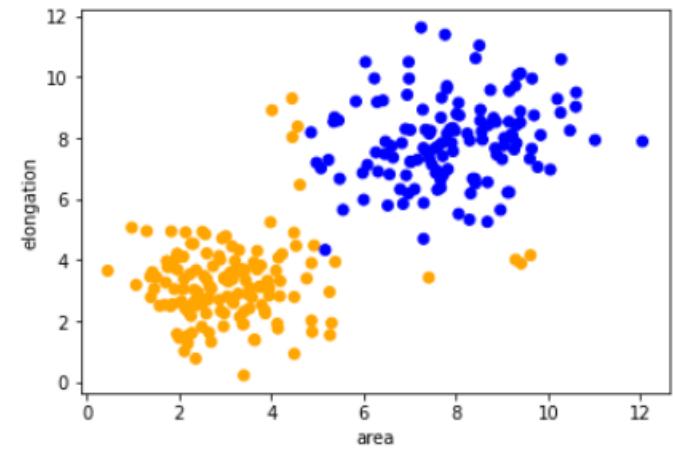


Classifier training

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

Classifier prediction

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



Interactive pixel classification

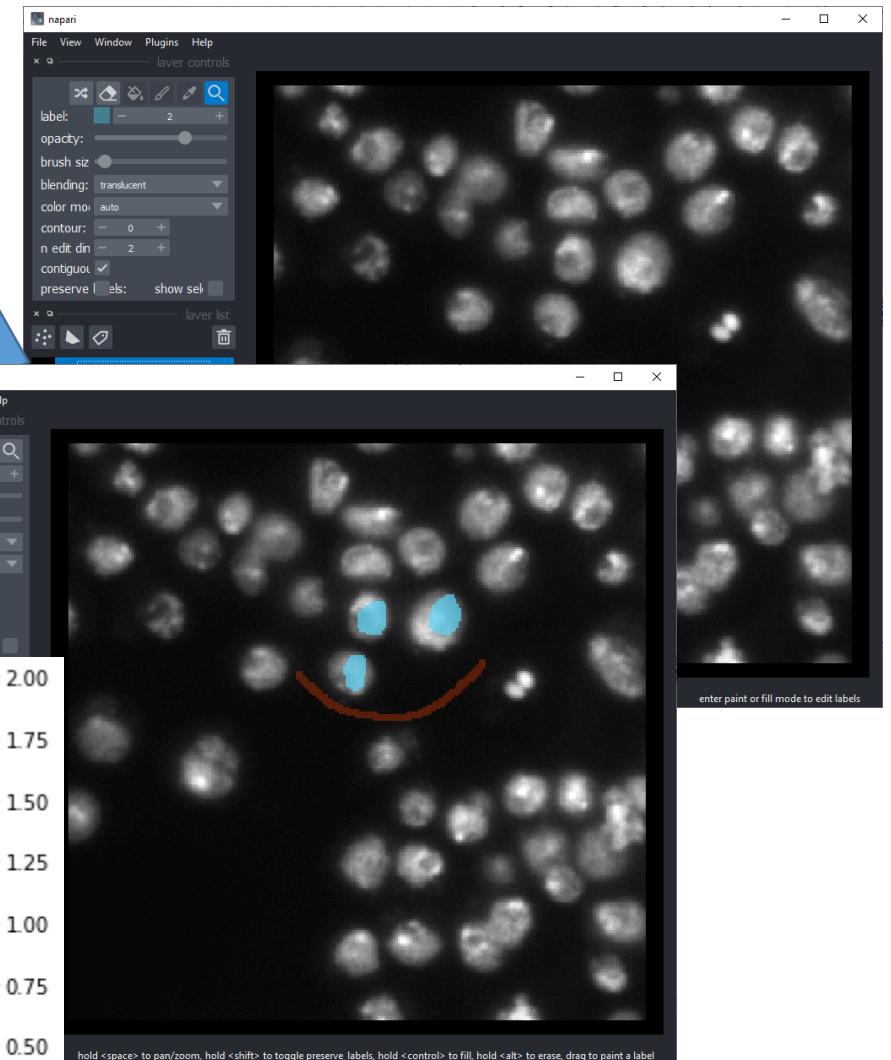
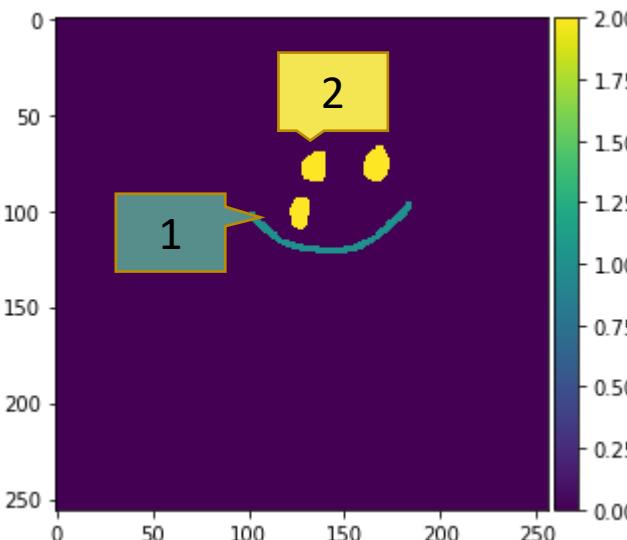
- Prepare an empty layer for annotations and keep a reference

```
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)
```



- Pixel classification using scikit-learn
 - Expects one-dimensional arrays for
 - every feature individually
 - 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

Image data

y_ = classifier.predict(X)

prediction

Interactive pixel classification

- Pixel classification using scikit-learn

- Expects one-dimensional arrays for
 - every feature individually
 - ground truth

```
# for training, we need to generate features
```

```
feature_stack = generate_feature_stack(image)
```

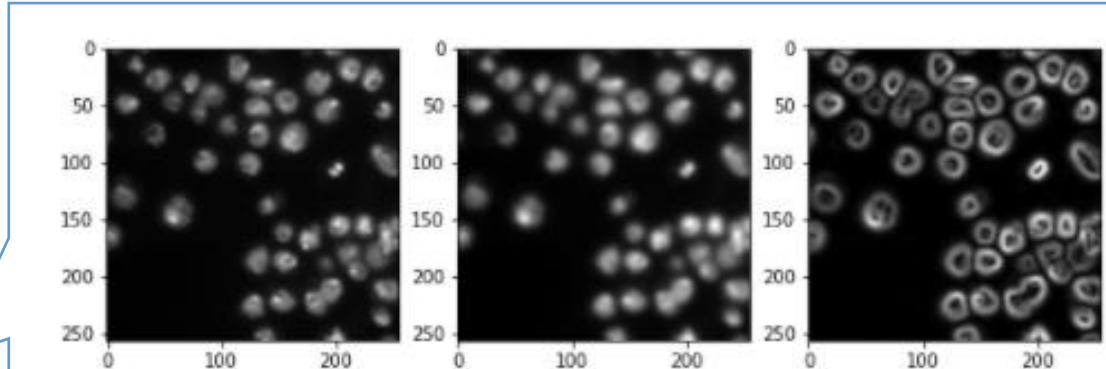
```
X, y = format_data(feature_stack, manual_annotations)
```

```
# train classifier
```

```
from sklearn.ensemble import RandomForestClassifier
```

```
classifier = RandomForestClassifier(max_depth=2, random_state=0)
```

```
classifier.fit(X, y)
```

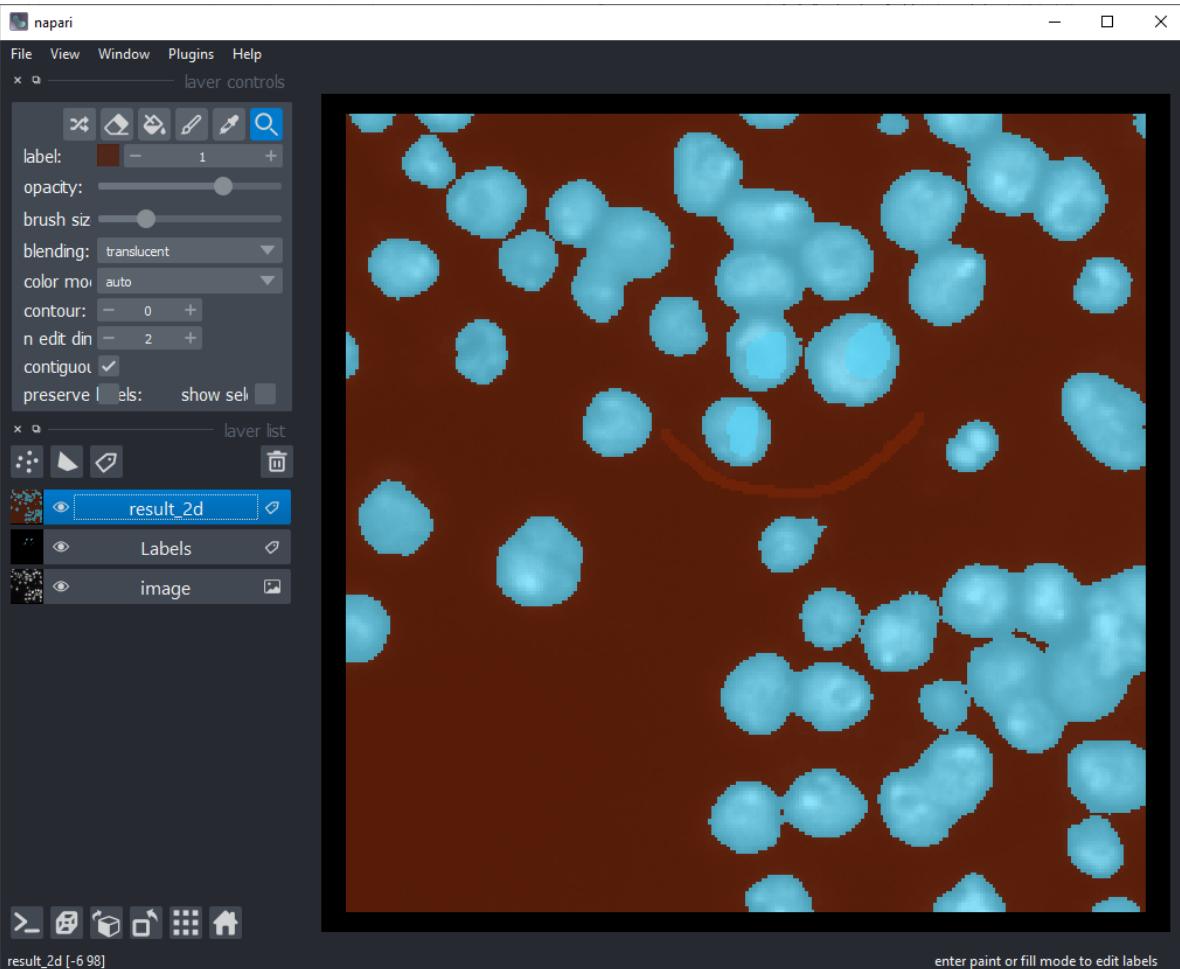


Interactive pixel classification

- Pixel classification using scikit-learn

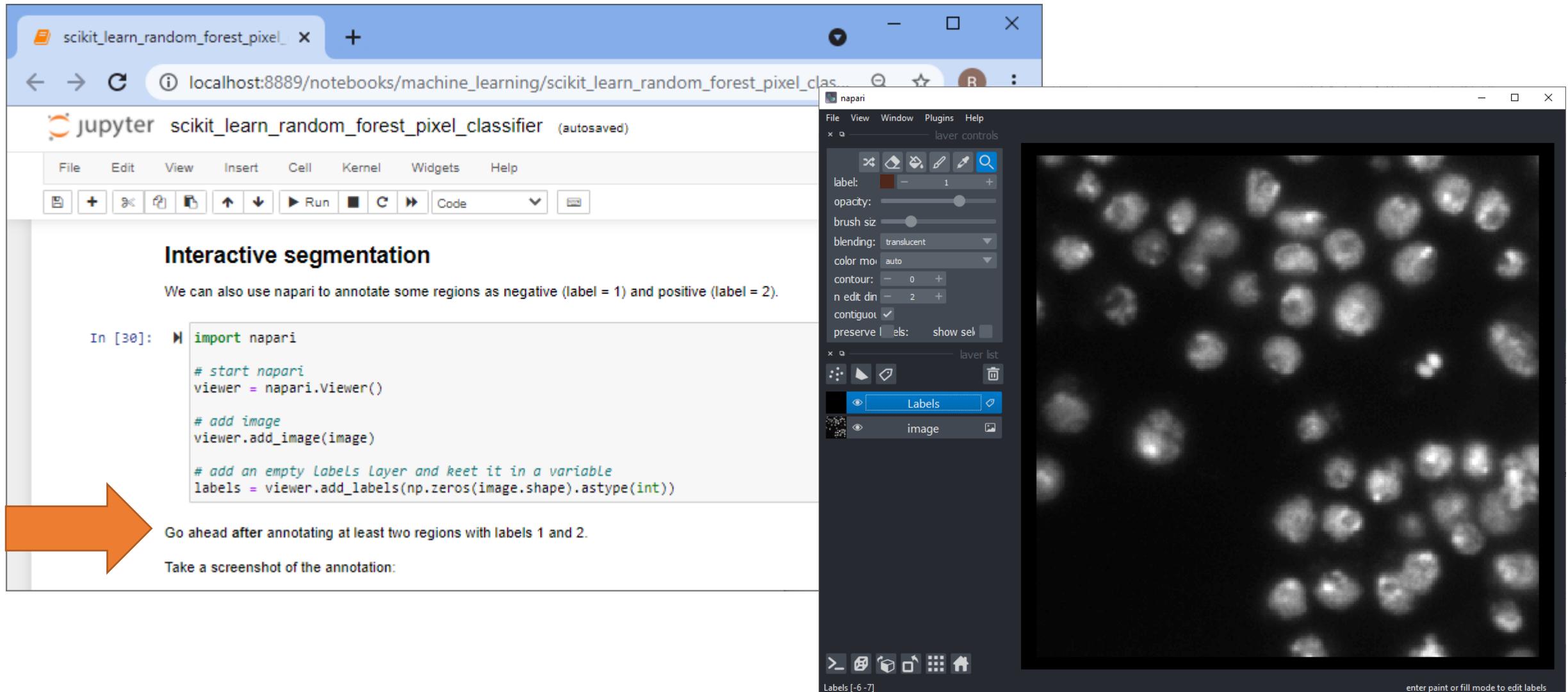
```
# process the whole image and show result
result_1d = classifier.predict(feature_stack.T)
result_2d = result_1d.reshape(image.shape)

viewer.add_labels(result_2d)
```



Interactive pixel classification

- Jupyter notebooks and napari side-by-side



The screenshot shows a Jupyter notebook interface and the napari image viewer running side-by-side.

Jupyter Notebook:

- Tab title: scikit_learn_random_forest_pixel_
- URL: localhost:8889/notebooks/machine_learning/scikit_learn_random_forest_pixel_clas...
- Code cell (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))
```
- Text below code:

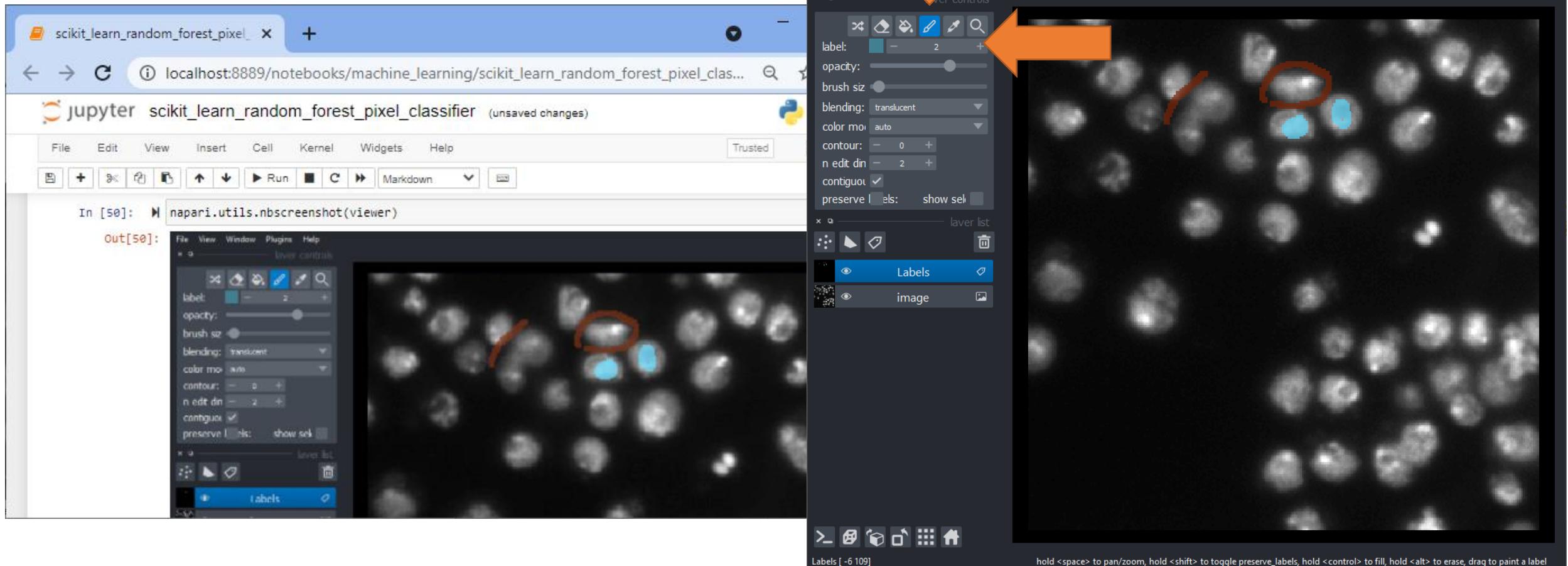
Go ahead after annotating at least two regions with labels 1 and 2.
Take a screenshot of the annotation:

napari Interface:

- Top bar: File, View, Window, Plugins, Help
- Left sidebar:
 - Layer controls: label (dropdown), opacity (slider), brush size (slider), blending (dropdown), color mode (dropdown), contour (slider), n edit dist (slider), contiguous (checkbox), preserve labels (checkbox).
 - Layer list: Labels (selected), image.
- Right panel: A grayscale image of cells with bright spots, representing the input image for segmentation.
- Bottom status bar: Labels [-6 -7], enter paint or fill mode to edit labels.

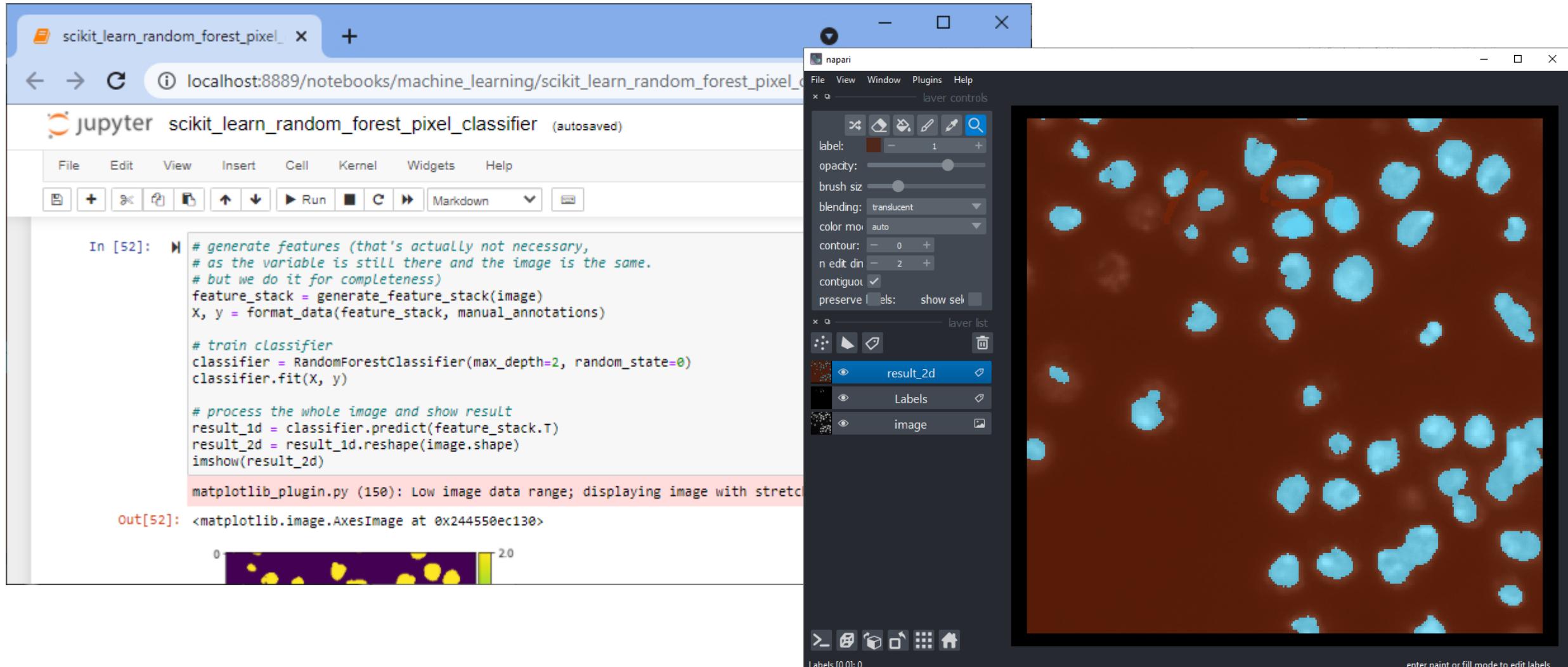
Interactive pixel classification

- Jupyter notebooks and napari side-by-side



Interactive pixel classification

- Jupyter notebooks and napari side-by-side



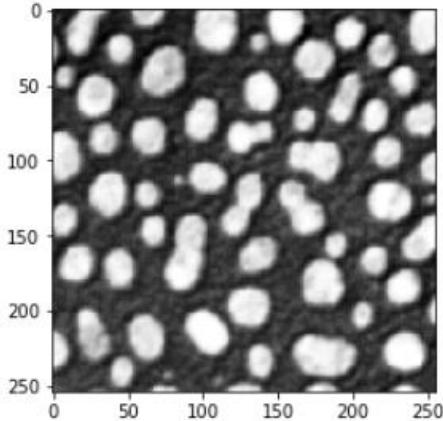
Accelerated pixel and object classification (APOC)

Robert Haase

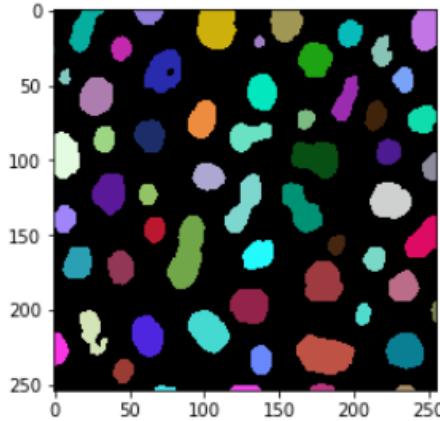
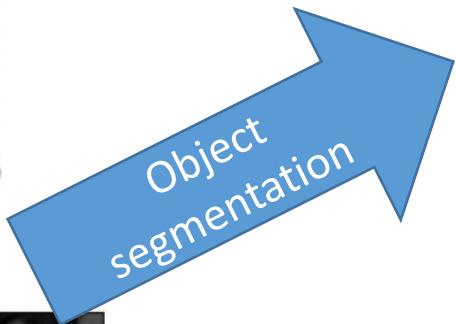
August 2023

Accelerated pixel and object classification

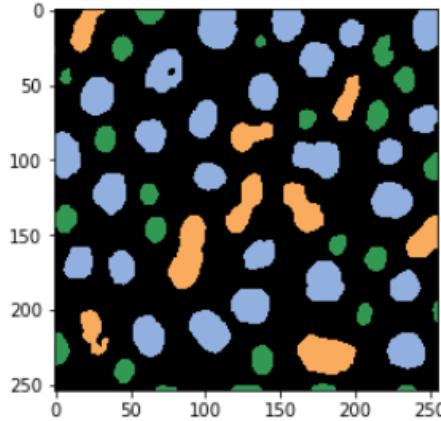
- APOC is a python library that makes use of OpenCL-compatible Graphics Cards to accelerate pixel and object classification



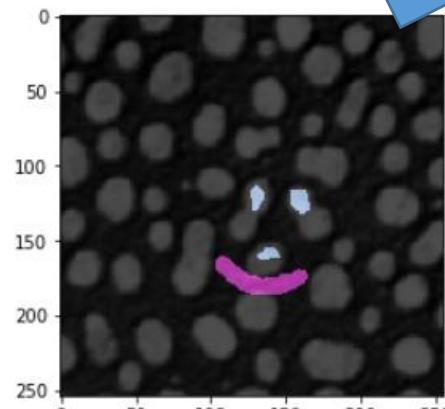
Raw image



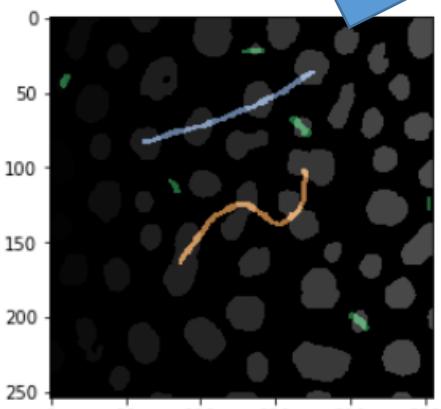
Object label image



Class label image



Pixel annotation

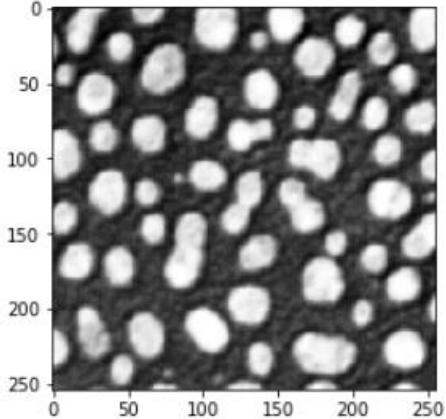


Object annotation

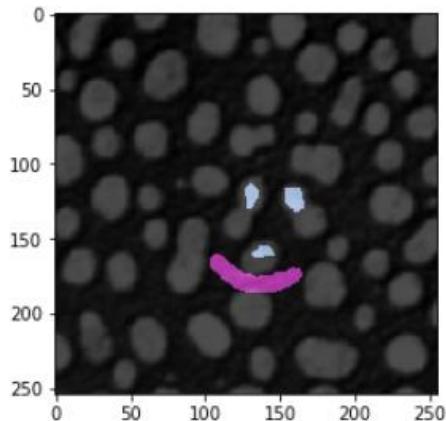


Object segmentation

- Pixel classification + connected component labeling



Raw image



Pixel annotation

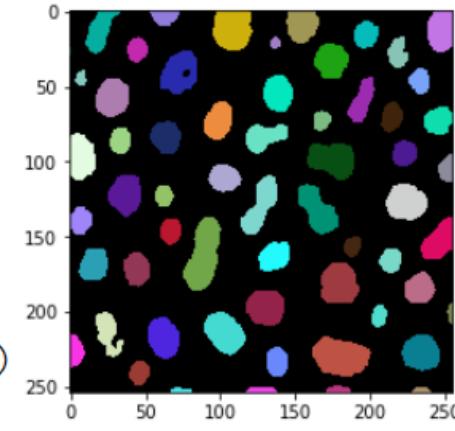
```
# define features
features = "gaussian_blur=1 gaussian_blur=5 sobel_of_gaussian_blur=1"

# 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)
```

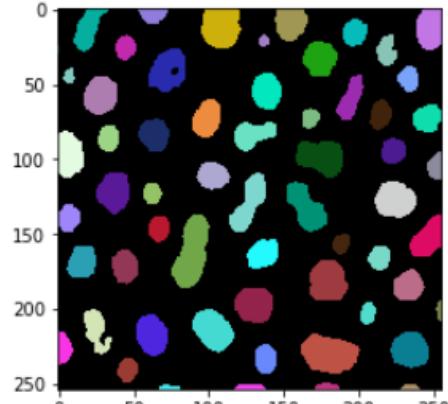


Object label image

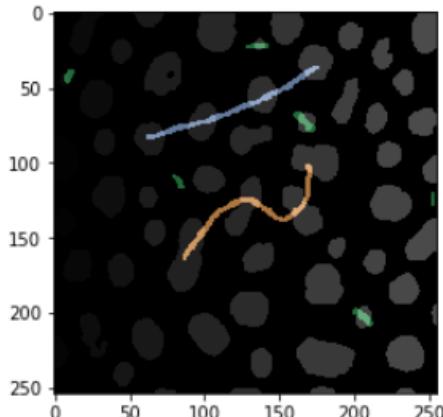
Object segmentation

Object classification

- Feature extraction + tabular classification



Object label image



Object annotation

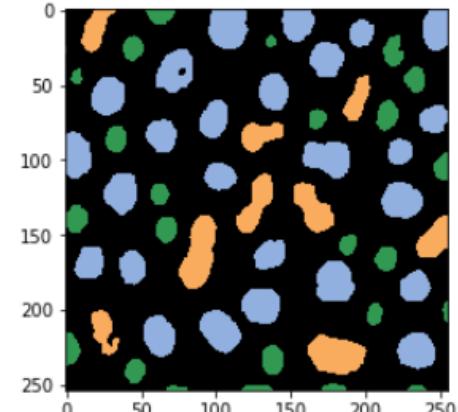
```
# for the classification we define size and shape as criteria
features = 'area mean_max_distance_to_centroid_ratio'

# 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)
```

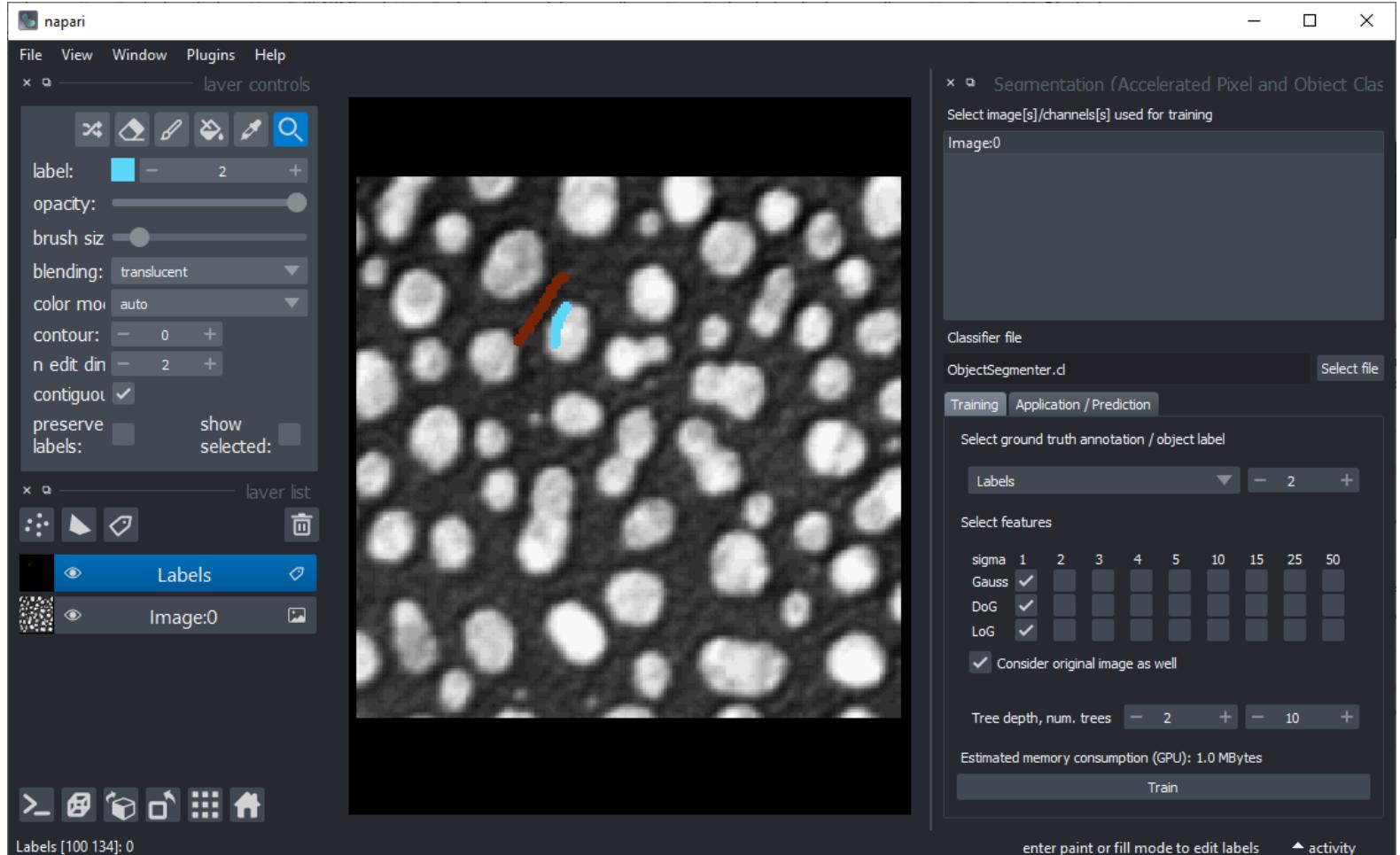


Class label image

Object classification

Graphical user interface

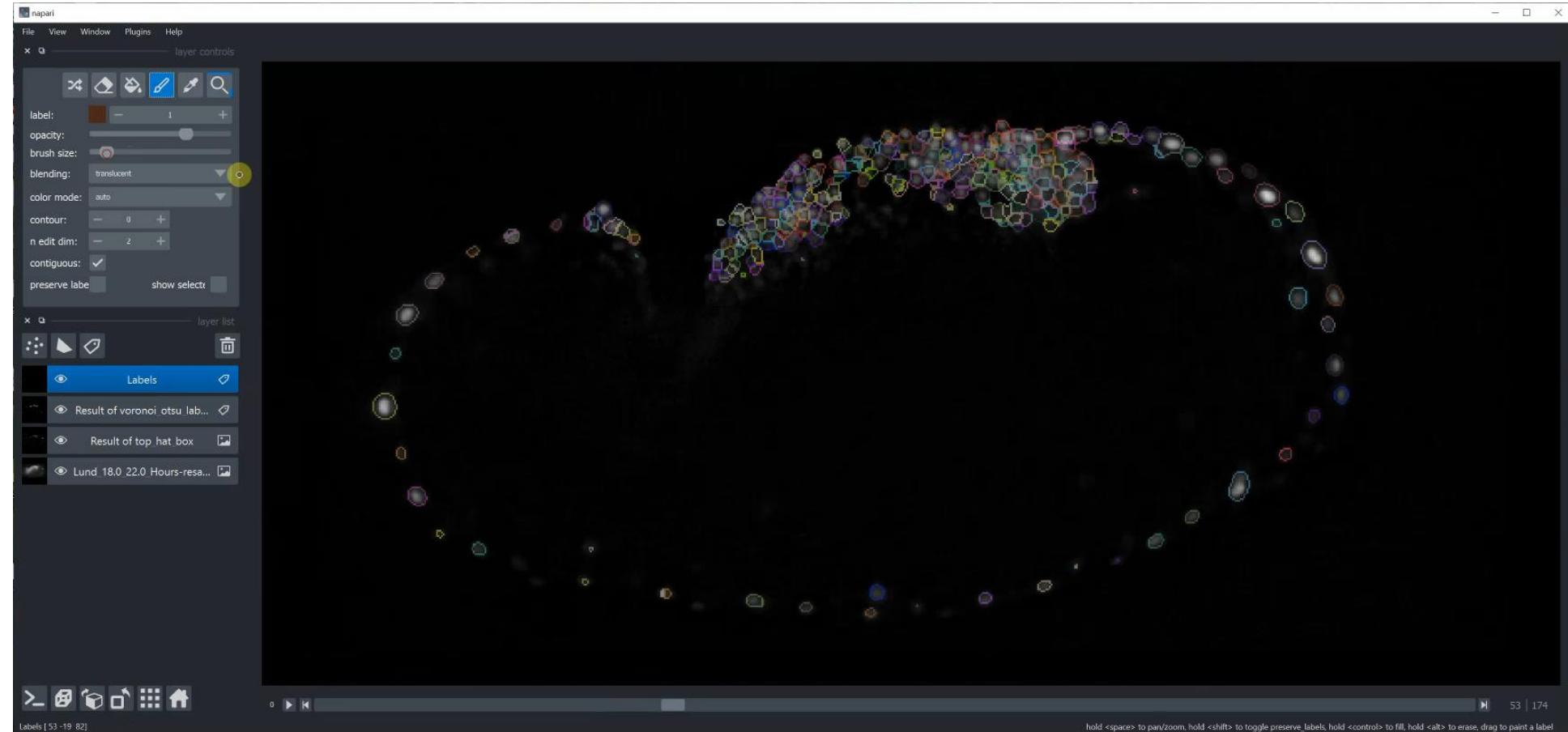
- Object segmentation
- <https://github.com/haesleinhuepf/napari-accelerated-pixel-and-object-classification#object-and-semantic-segmentation>



Supervised machine learning for tissue classification

Random Forest
Classifiers based on

- scikit-learn and
- clesperanto



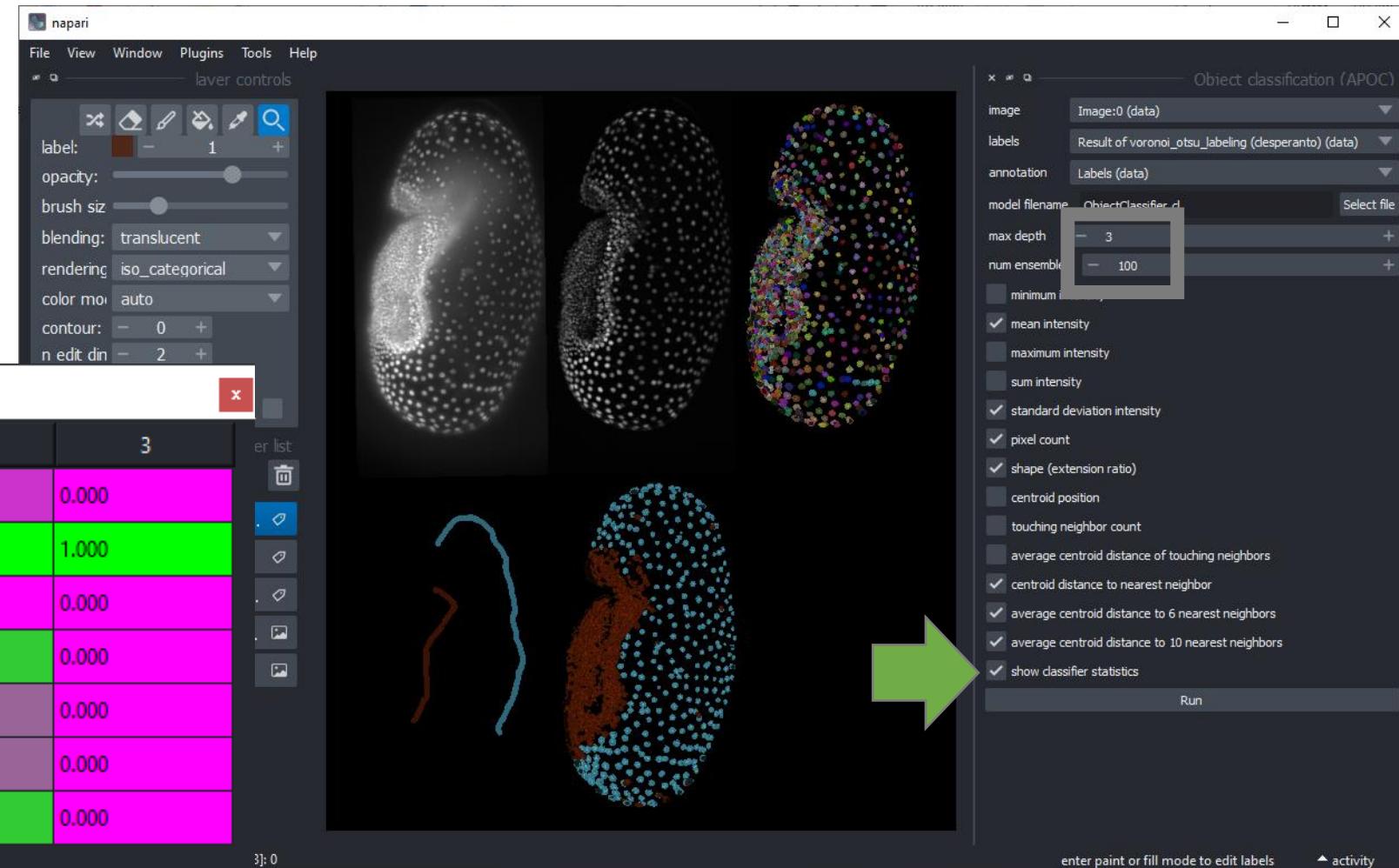
@haesleinhuepf
@PoLDresden

<https://github.com/haesleinhuepf/napari-accelerated-pixel-and-object-classification>

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

Data exploration / supervised machine learning

- Inspect how the random forest classifier makes decisions
- Note: Beware of correlated parameters!



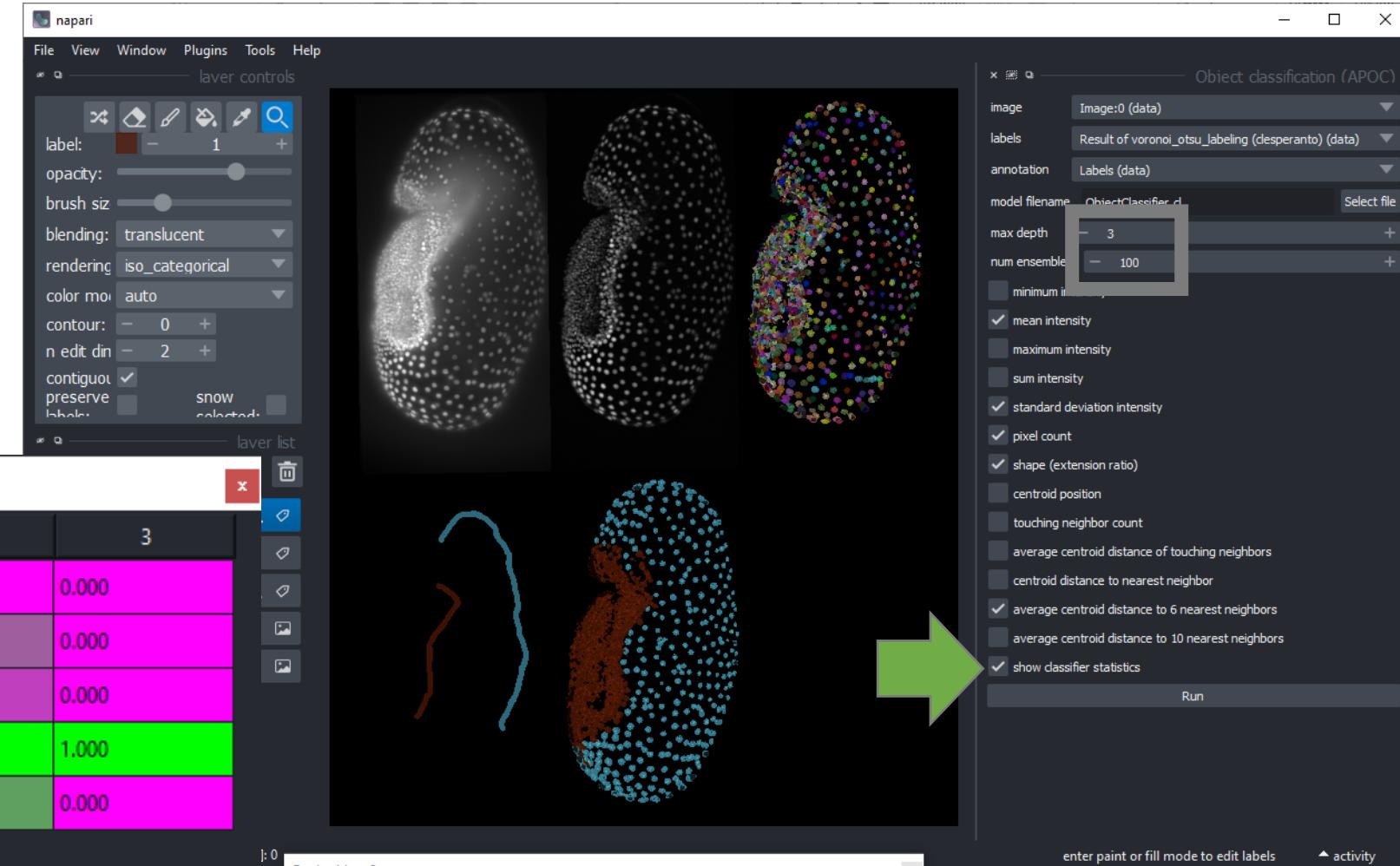
@haesleinhuepf
@PoLDresden

<https://github.com/haesleinhuepf/napari-accelerated-pixel-and-object-classification>

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

Data exploration / supervised machine learning

- Inspect how the random forest classifier makes decisions
- Note: Beware of correlated parameters!



@haesleinhuepf
@PoLDresden

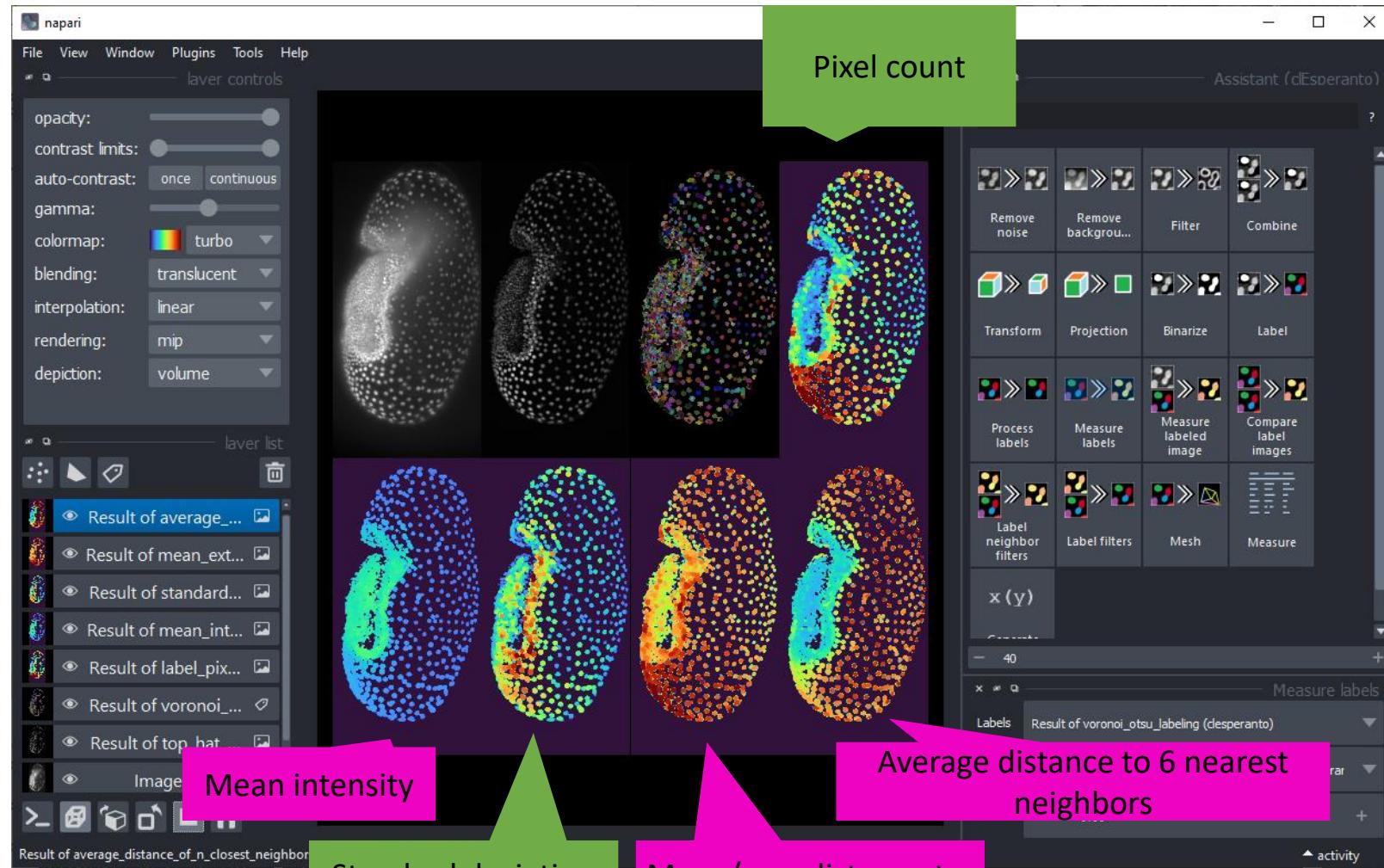
<https://github.com/haesleinhuepf/napari-accelerated-pixel-and-object-classification>

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

Data exploration / supervised machine learning

- Inspect how the random forest classifier makes decisions
- Note: Beware of correlated parameters!

	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



https://github.com/clEsperanto/napari_pyclesperanto_assistant

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

Exercises

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

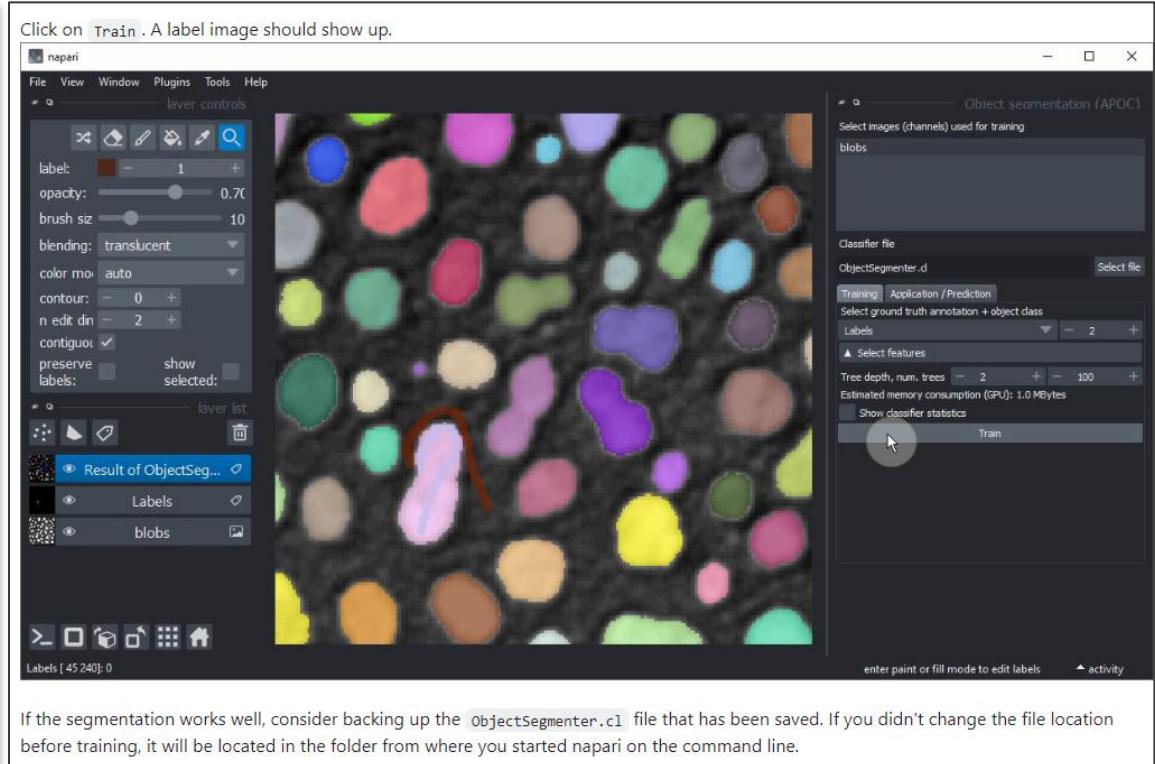
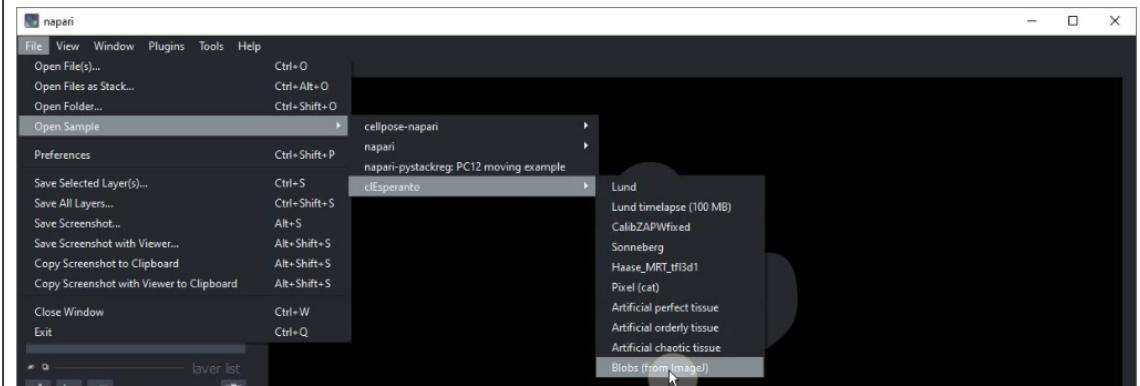
Open a terminal window and activate your conda environment:

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

Afterwards, start up Napari:

```
napari
```

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



https://biapol.github.io/PoL-BioImage-Analysis-TS-Early-Career-Track/day2b_machine_learning_apoc/interactive_pixel_classification/intro.html

Object classification

- Use Napari to group 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:

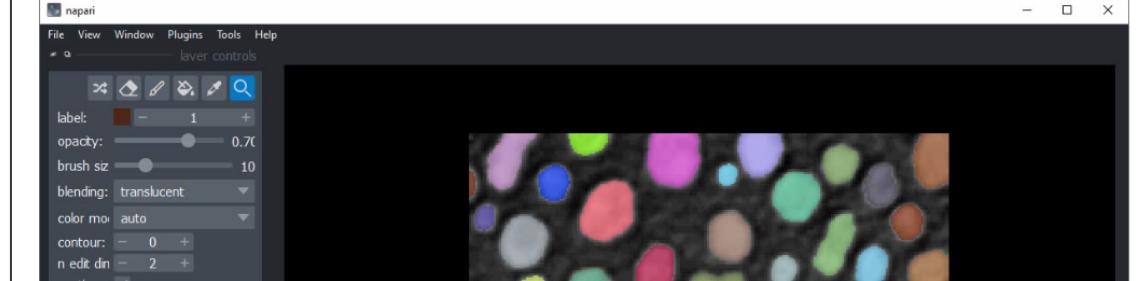
```
napari
```

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

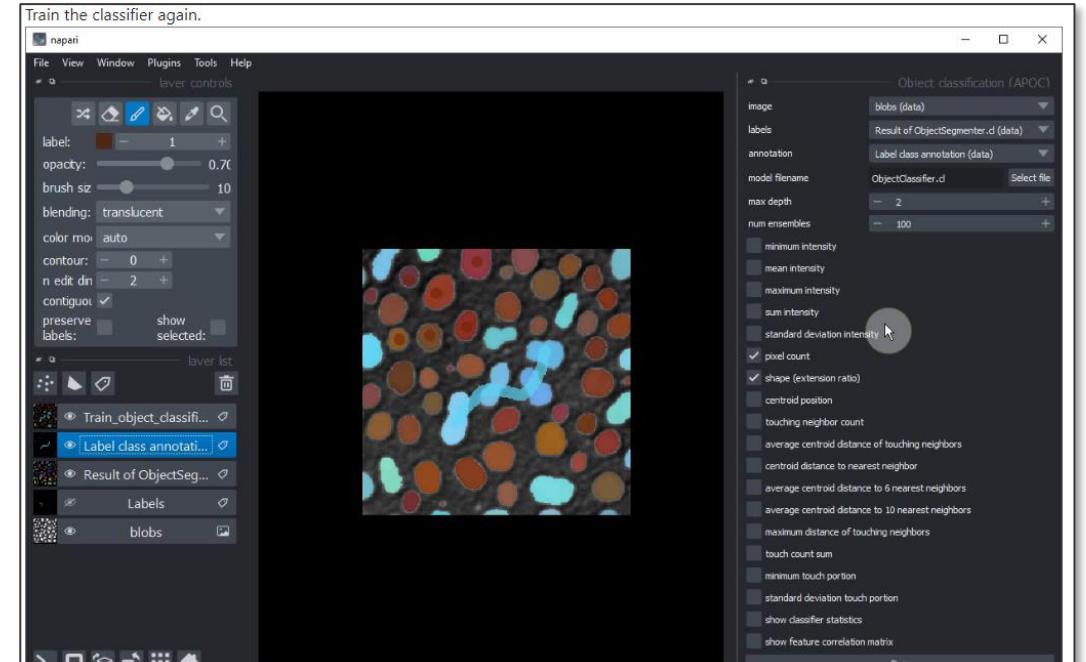
We furthermore need a label image. You can create it using the pixel classifier trained earlier or using the menu [Tools > Segmentation / labeling > Gauss-Otsu Labeling \(clesperanto\)](#).

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

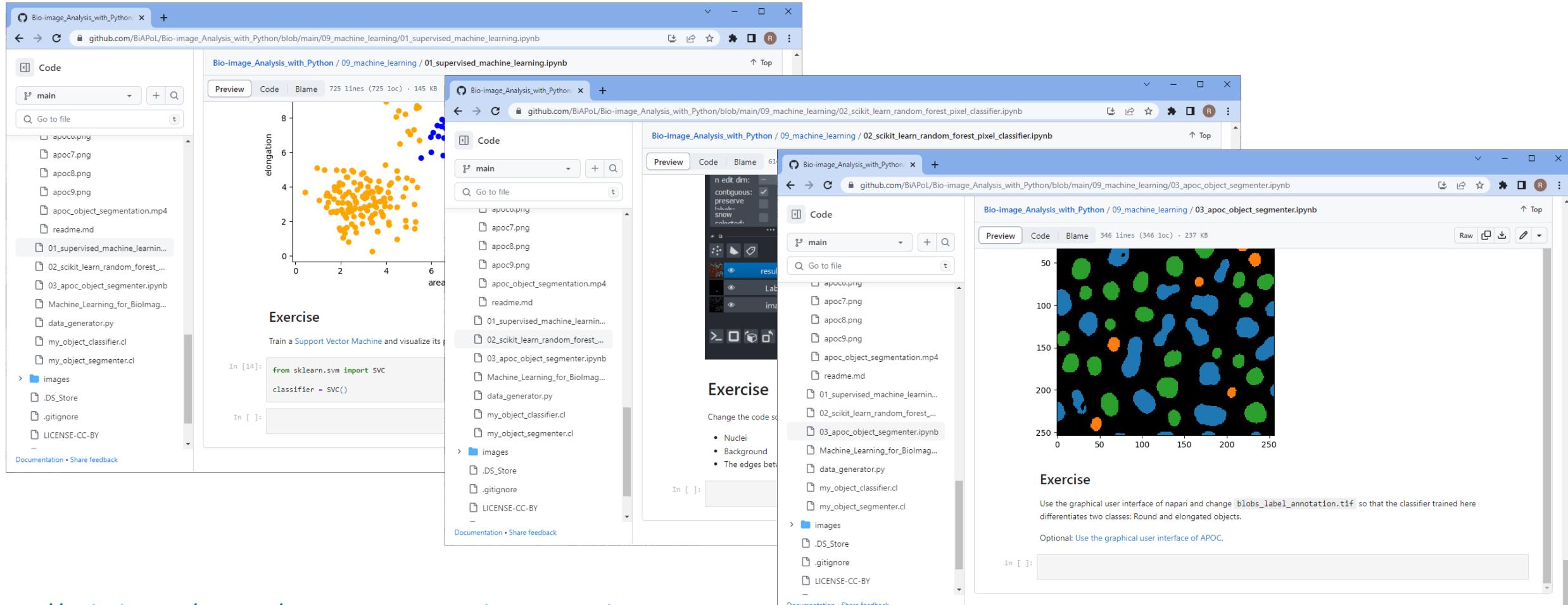
Retrain the classifier so that it can differentiate three different classes:

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

https://biapol.github.io/PoL-BioImage-Analysis-TS-Early-Career-Track/day2b_machine_learning_apoc/interactive_object_classification/intro.html
August 2023

Machine learning using Python

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



The image shows three Jupyter Notebook windows side-by-side, each displaying a different aspect of machine learning analysis:

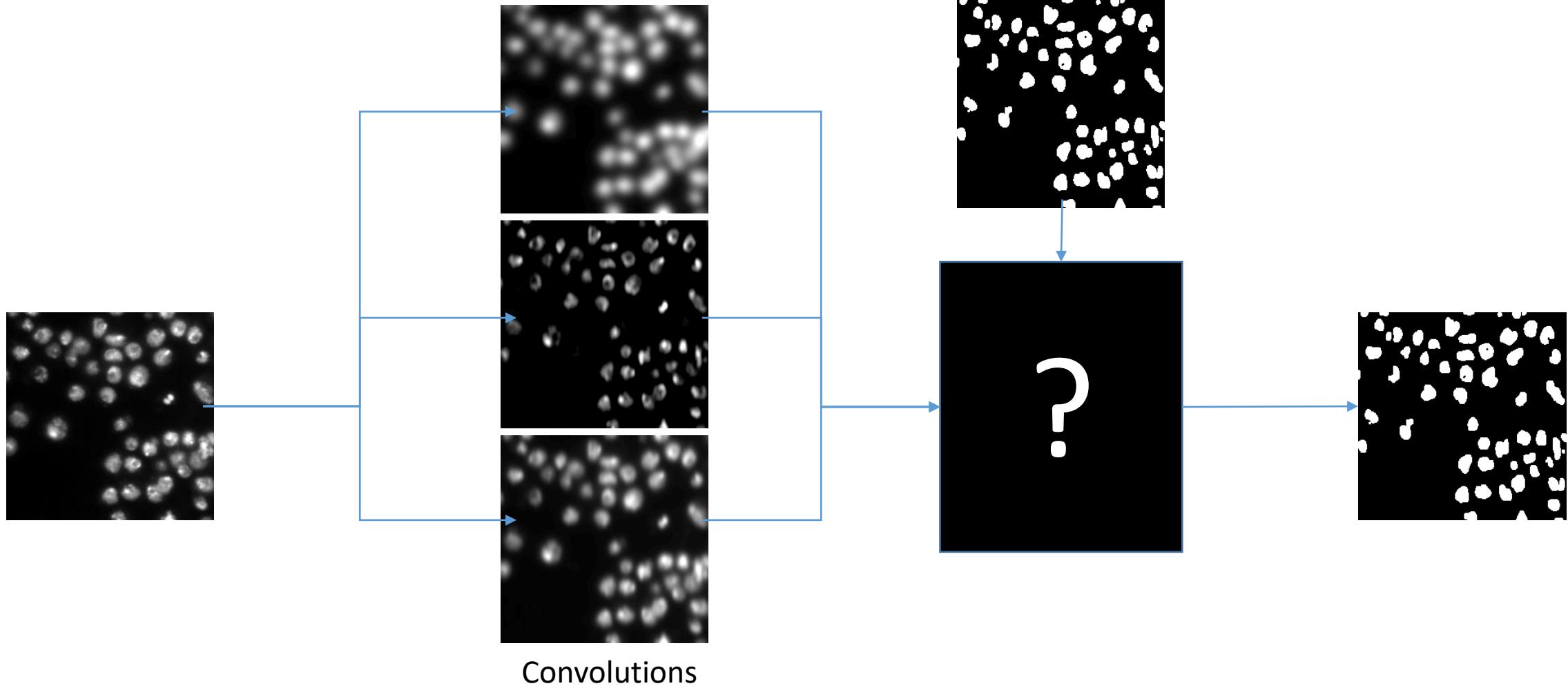
- Left Notebook:** Titled "Bio-image_Analysis_with_Python / 01_supervised_machine_learning.ipynb". It contains a scatter plot of "elongation" vs "area" with two classes of points (orange and blue), and a code cell for training a Support Vector Machine (SVC) classifier.
- Middle Notebook:** Titled "Bio-image_Analysis_with_Python / 02_scikit_learn_random_forest_pixel_classifier.ipynb". It displays a grayscale image of biological objects and a corresponding heatmap of predicted labels (Nuclei, Background, Edges).
- Right Notebook:** Titled "Bio-image_Analysis_with_Python / 03_apoc_object_segmenter.ipynb". It shows a binary mask segmentation of the same image, where different regions are colored (blue, green, orange) based on their shape characteristics.

https://github.com/BiAPoL/PoL-BioImage-Analysis-TS-Early-Career-Track/blob/main/docs/day2b_machine_learning_apoc/04_demo_object_segmenter.ipynb

Summary & outlook

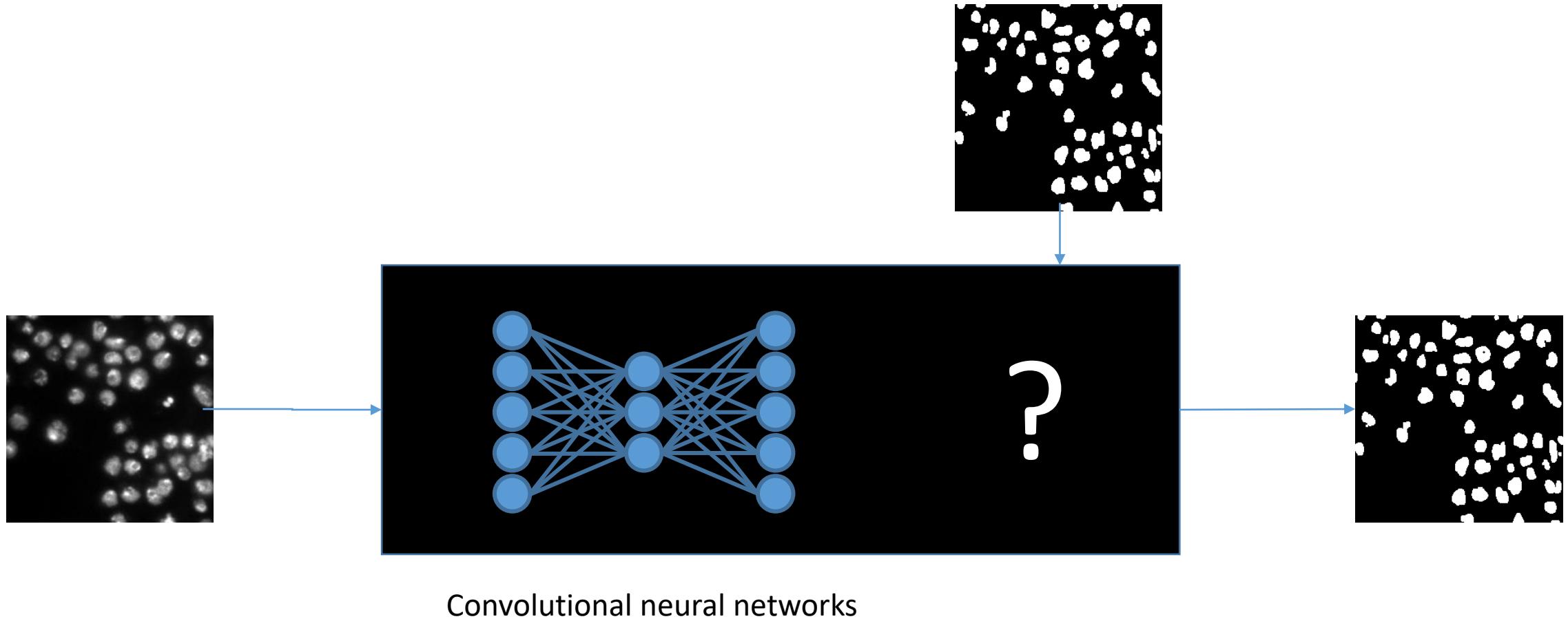
Machine learning for image analysis

- In classical machine learning, we typically select features for training our classifier



Outlook: Deep learning for image analysis

- In deep learning, this selection becomes part of the black box



Today, you learned

- Machine learning for Pixel and Object segmentation
- Python
 - Scikit-learn / napari
 - Accelerated pixel and object classifiers (APOC)

Coming up next:

- Feature extraction

