ScaDS.AI
DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE

Introduction to 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.

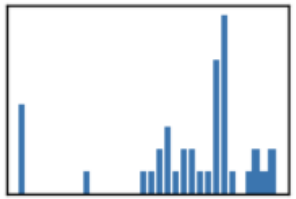
Reusing materials from Johannes Soltwedel, Till Korten, Johannes Soltwedel, Laura Žigutytė (TU Dresden), Ryan Savill (MPI-CBG Dresden), Matthias Täschner (ScaDS.AI/Uni Leipzig) and the Scikit-learn community.

Follow-up: Tiled versus full image processing

How to count labels

```
stackview.insight(result)
```

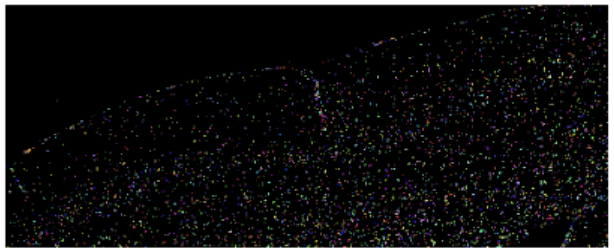
shape	(4, 10)
dtype	float64
size	320.0 B
min	0.10810810327529907
max	135.54044342041016



```
[11]: result.sum()
```

```
[11]: 3548.6168007031083
```

```
image = zarr_image.compute() # this would not work if the image was larger than com
labels = cle.voronoi_otsu_labeling(image, spot_sigma=3.5)
stackview.insight(labels)
```



shape	(2000, 5000)
dtype	uint32
size	38.1 MB
min	0
max	4701
n labels	4701

```
[12]: labels.max()
```

```
[12]: 4701.0
```

Follow-up: Tiled versus full image processing

```python
def count_nuclei(image):
    """
    Label objects in a binary image and produce a pixel-count-map image.
    """
    print("Processing image of size", image.shape)

    # Count nuclei including those which touch the image border
    labels = cle.voronoi_otsu_labeling(image, spot_sigma=3.5)
    label_intensity_map = cle.mean_intensity_map(image, labels)
                          labels
    nuclei_count = label_intensity_map.max()

    # Count nuclei including those which touch the image border
    labels_without_borders = cle.exclude_labels_on_edges(label_intensity_map)
    nuclei_count_excluding_borders = labels_without_borders.max()

    # Both nuclei-count including and excluding nuclei at image borders
    # are no good approximation. We should exclude the nuclei only on
    # half of the borders to get a good estimate.
    # Alternatively, we just take the average of both counts.
    result = np.asarray([[(nuclei_count + nuclei_count_excluding_borders) / 2]])

    print(result)

    return result
```

Exercise

Compare the nuclei count results of both methods.

```
[20]: result.sum()
```

```
[20]: 4474.0
```

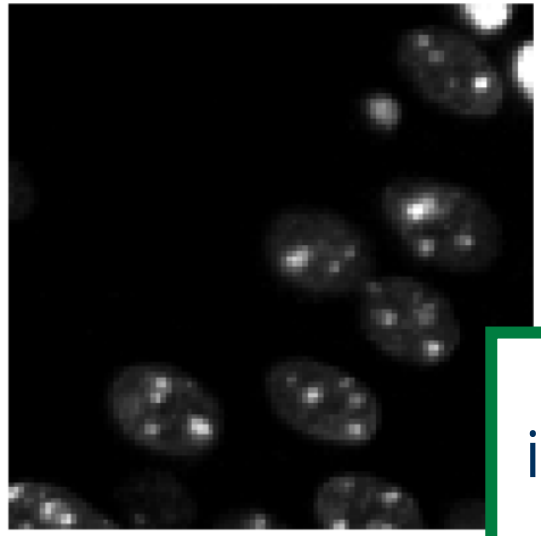
```
[21]: labels.max()
```

```
[21]: 4701.0
```

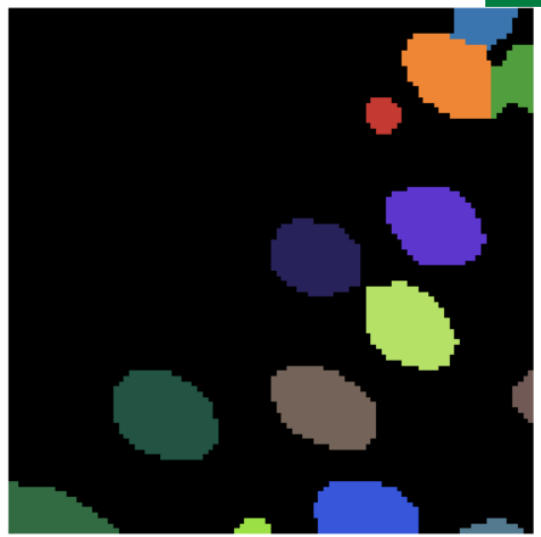
Follow-up: Tiled versus full ima

```
def count_nuclei(image, debug=False):
    """
    Label objects in a binary image and produce a pixel-count-map image.
    """
    print("Processing image of size", image.shape)

    # Count nuclei including those which touch the image border
    labels = cle.voronoi_otsu_labeling((image > 20)*1, spot_sigma=3.5)
    if debug:
        stackview.imshow(labels)
    nuclei_count = labels.max()

    # Count nuclei including those which touch the image border
    labels_without_borders = cle.exclude_labels_on_edges(labels)
    nuclei_count_excluding_borders = labels_without_borders.max()

    # Both nuclei-count including and excluding nuclei at image borders
    # are no good approximation. We should exclude the nuclei only on
    # half of the borders to get a good estimate.
    # Alternatively, we just take the average of both counts.
    result = np.asarray([[(nuclei_count + nuclei_count_excluding_borders) / 2]])

    return result
```

```
[5]: test_image = zarr_image[-100:, -100:]
stackview.imshow(test_image)
count_nuclei(test_image, debug=True)
```



```
Processing image of size (100, 100)
```



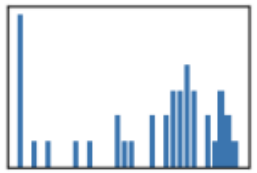
```
[5]: array([[10.5]])
```

Visualize intermediate results!

Follow-up: Tiled versus full image processing

```
stackview.insight(result)
```

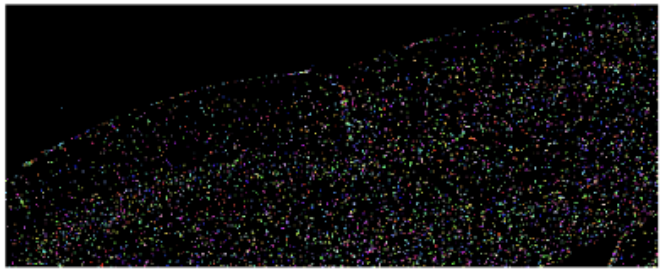
shape	(4, 10)
dtype	float64
size	320.0 B
min	0.0
max	236.0



```
result.sum()
```

```
5499.5
```

```
image = zarr_image.compute() # this would not work if the image was larger than
labels = cle.voronoi_otsu_labeling((image > 20)*1, spot_sigma=3.5)
stackview.insight(labels)
```



shape	(2000, 5000)
dtype	uint32
size	38.1 MB
min	0
max	5534
n labels	5534

```
labels.max()
```

```
5534.0
```

Pixel-Classification

Quiz: How is this task called?

Combinatorial Segmentation

Semantic Segmentation

Instance-Segmentation

Connected-Component Segmentation

ScaDS.AI
DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE

Introduction to Machine Learning for Bio-image Analysis

Robert Haase

Funded by

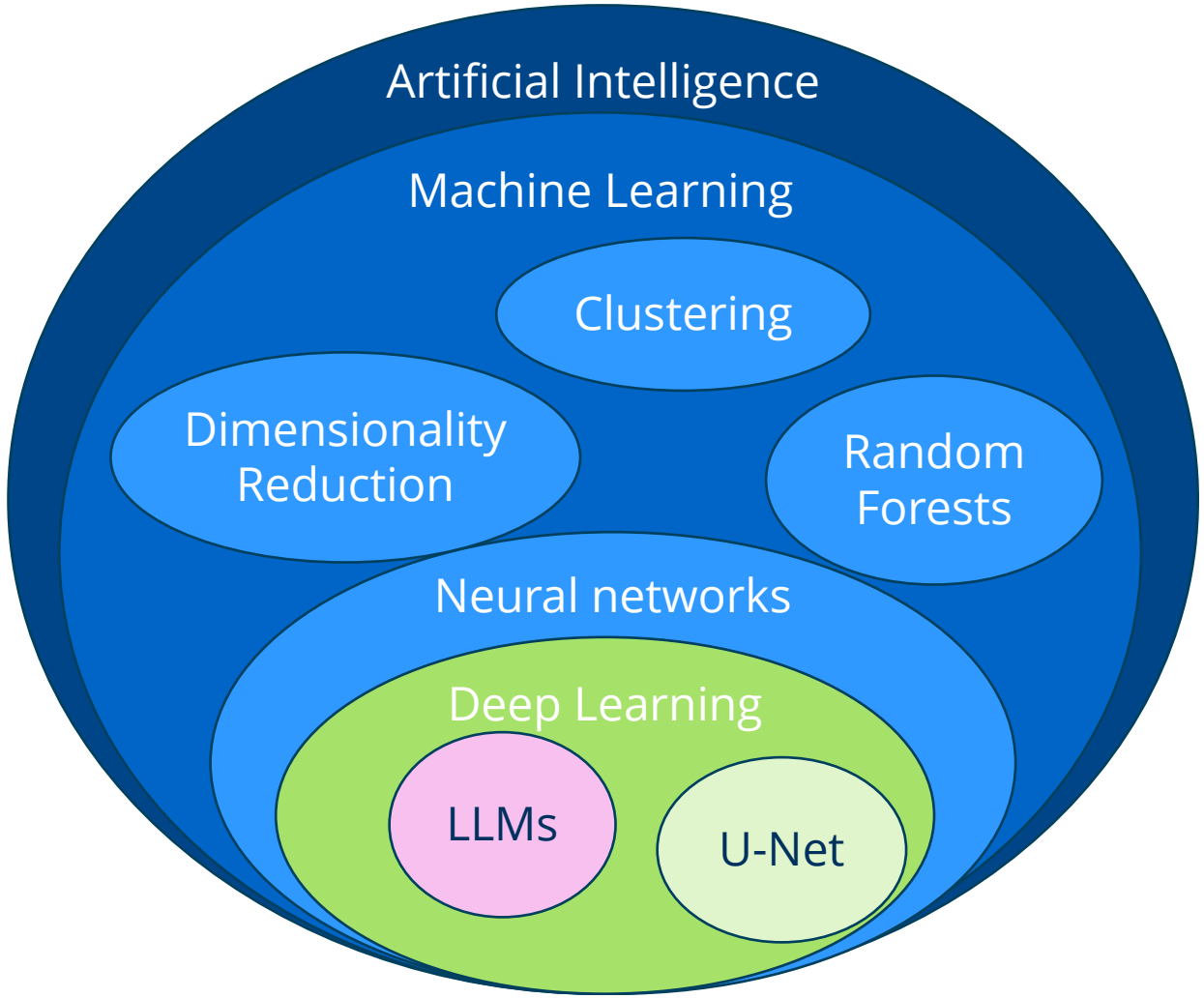
Bundesministerium für Bildung und Forschung

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

Artificial intelligence



Today
Next week
The week after

Artificial intelligence

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

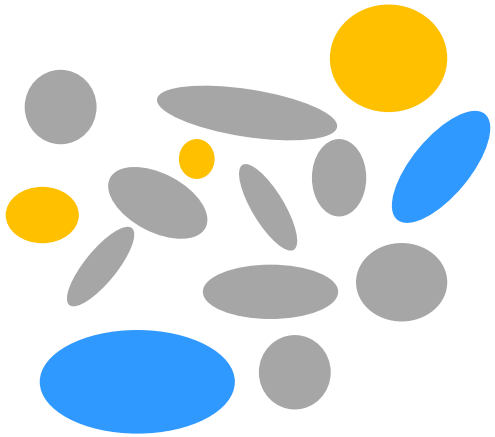
Great for data analysis tasks

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

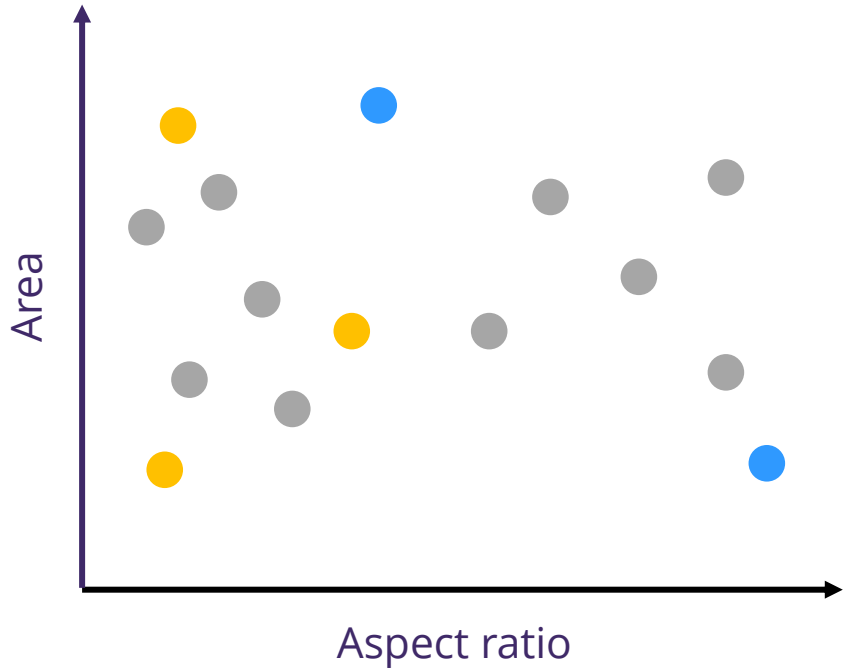
Labelled data

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

Bias?



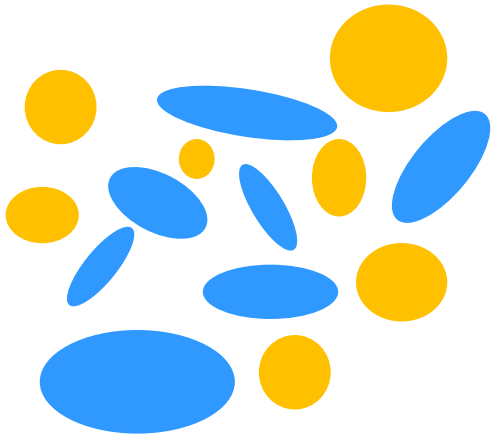
Elongated
Round
Unlabelled



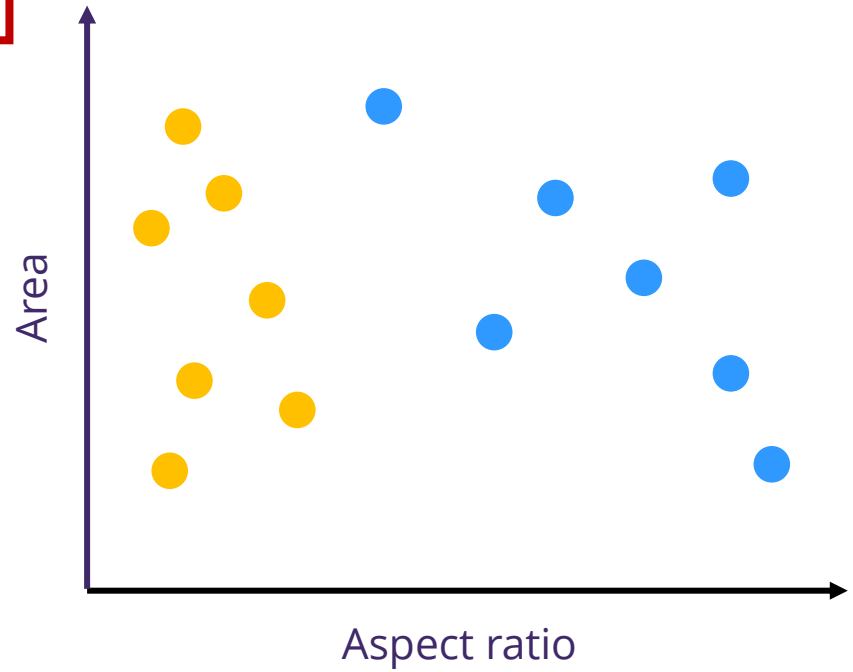
Labelled data

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

Typically expensive



Elongated
Round
Unlabelled

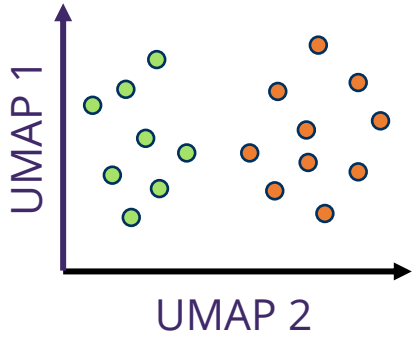


Artificial intelligence

Explorative → Analytic → Generative

Unsupervised ML

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



Supervised ML

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

Generative AI

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

ScaDS.AI
DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE

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

Is circularity the right parameter to measure?

Circularity measurement

Statistics

Hypothesis *generating* quantitative biology

~~Hypothesis: Cell shape can be influenced by modifying X.~~

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

Sample preparation

Imaging

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

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?

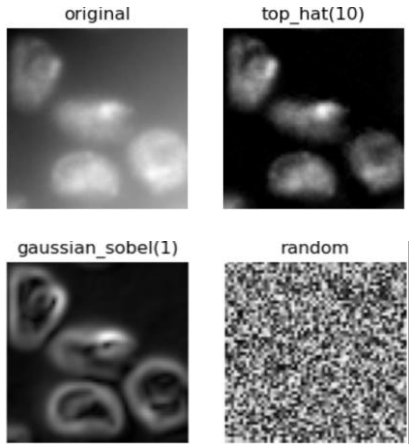
Feature selection

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

- Amplitude
- Energy
- Duration
- …

- Noise
- Tourists jumping on a sensor
- Earthquake approaching

Signal classification



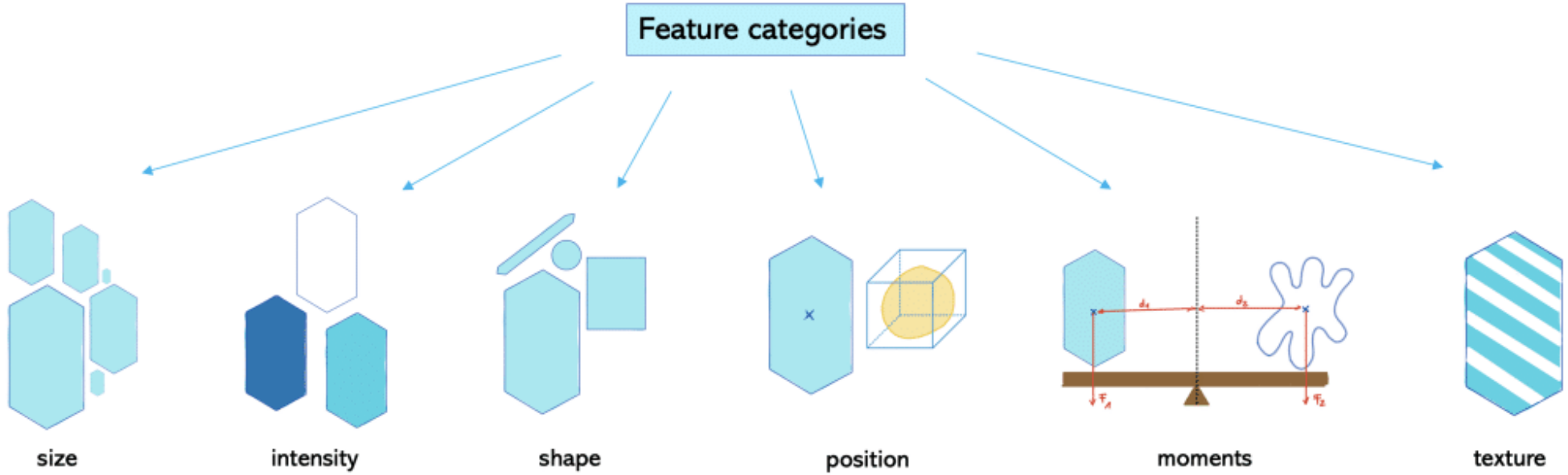
Pixel classification

- Area
- Perimeter
- Aspect ratio
- …

- Round
- Elongated

Object classification

Feature selection



Which of these features reflect the phenotype we are perceiving?

Feature selection: challenges

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

Feature selection

Question: Which features shall I analyse?

Challenges:

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

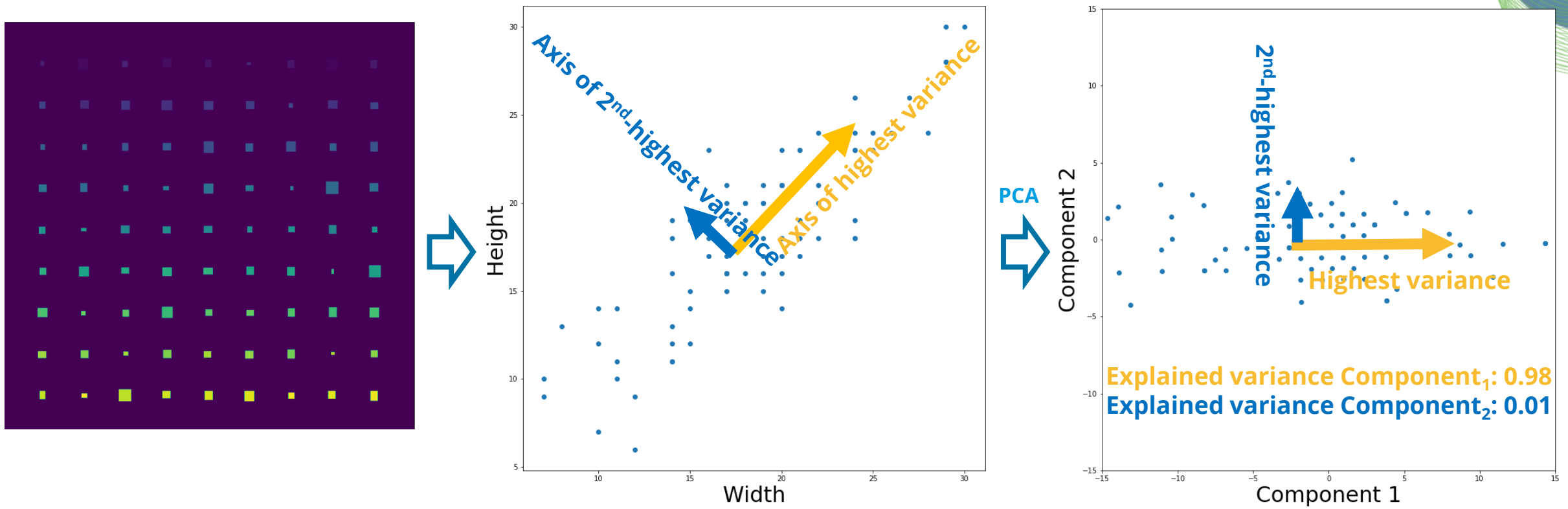
If you have no idea -> unsupervised machine learning

- Dimensionality reduction
- Clustering

PCA: Principal Component Analysis

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

Example: Squares of different size

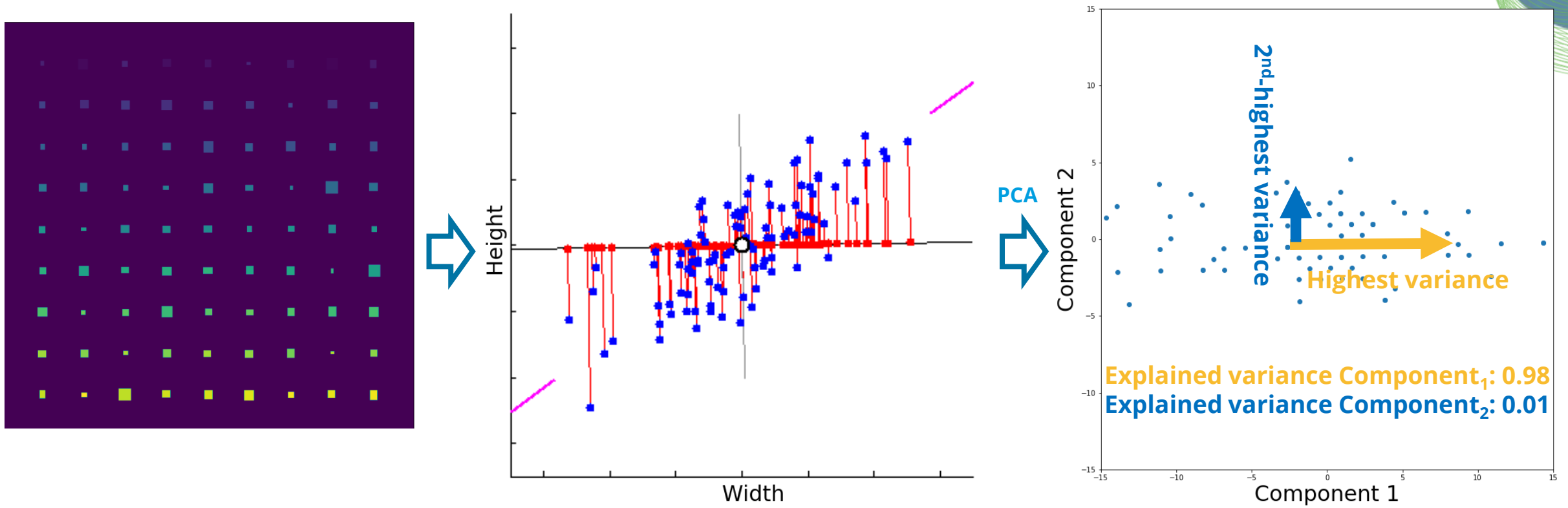


→ PCA transforms width/height measurements into a coordinate system that explains existing variance better

PCA: Principal Component Analysis

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

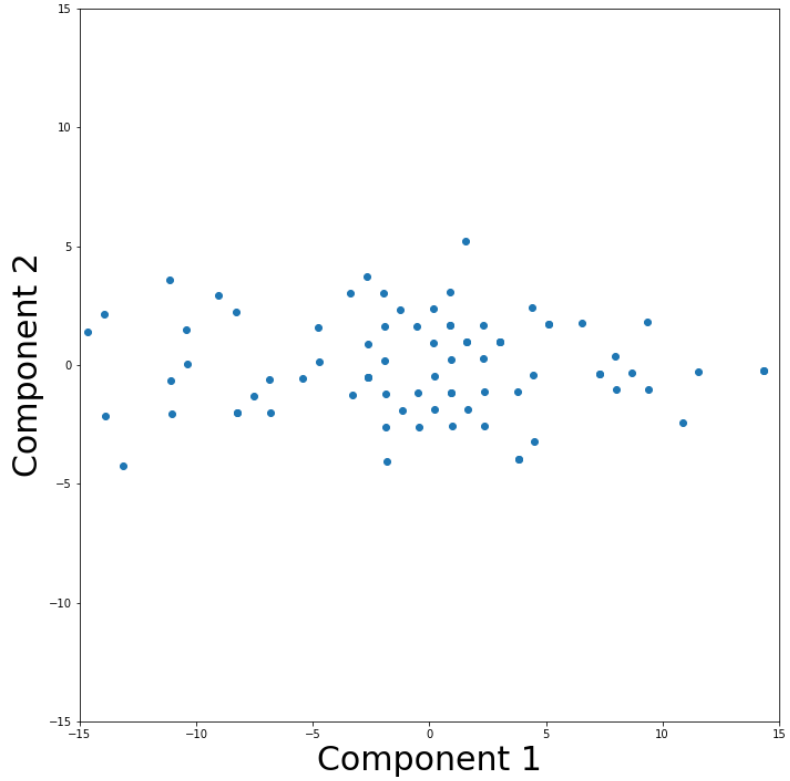
Example: Squares of different size



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

Embeddings

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



PCA in Python: `sklearn.decomposition.PCA`

- Import package

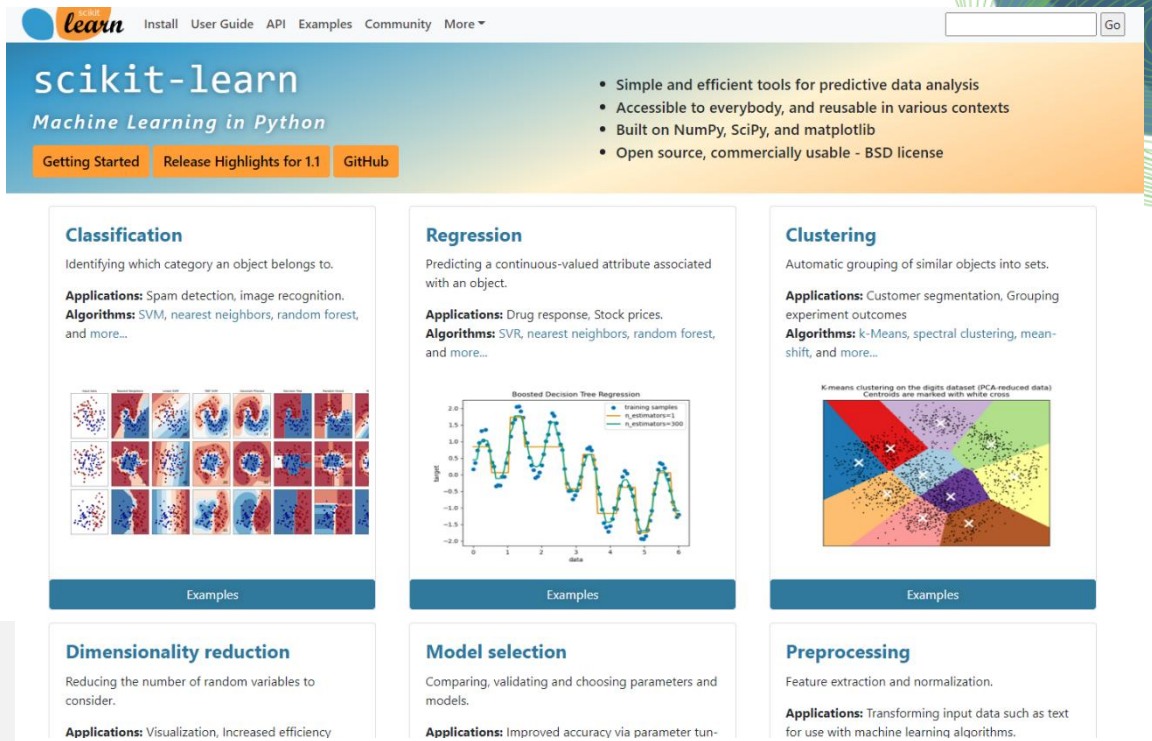
```
from sklearn.decomposition import PCA
```

- Apply PCA

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

- Transform data into new coordinate system

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



Important!

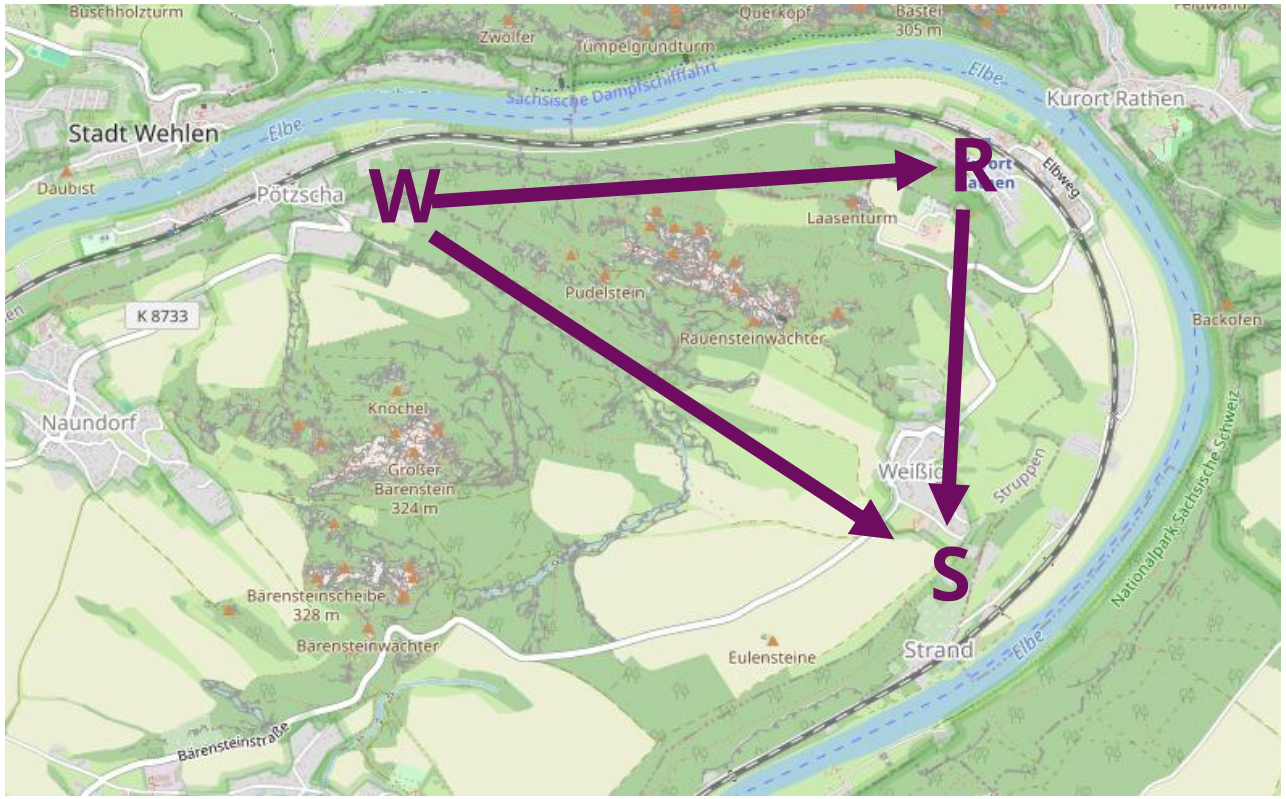
Always check the explained variance along the PCA component axes!

```
pca.explained_variance_ratio_
```

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

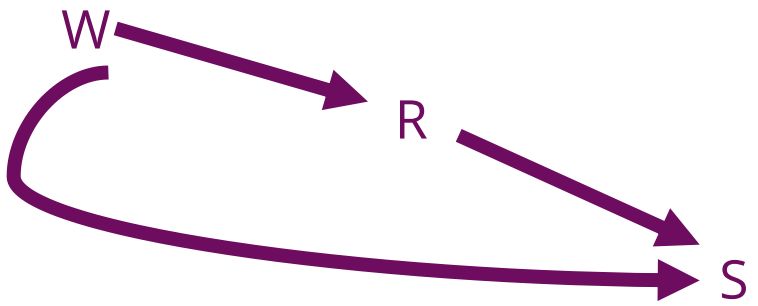
Non-Euclidian spaces

Not all features might be distances



Use travel time between W and S as metric for distance

→ Travelling from **W**ehlen to **S**trand by bike is probably faster if you make a detour through **R**athen



Uniform Manifold Approximation Projection (UMAP)

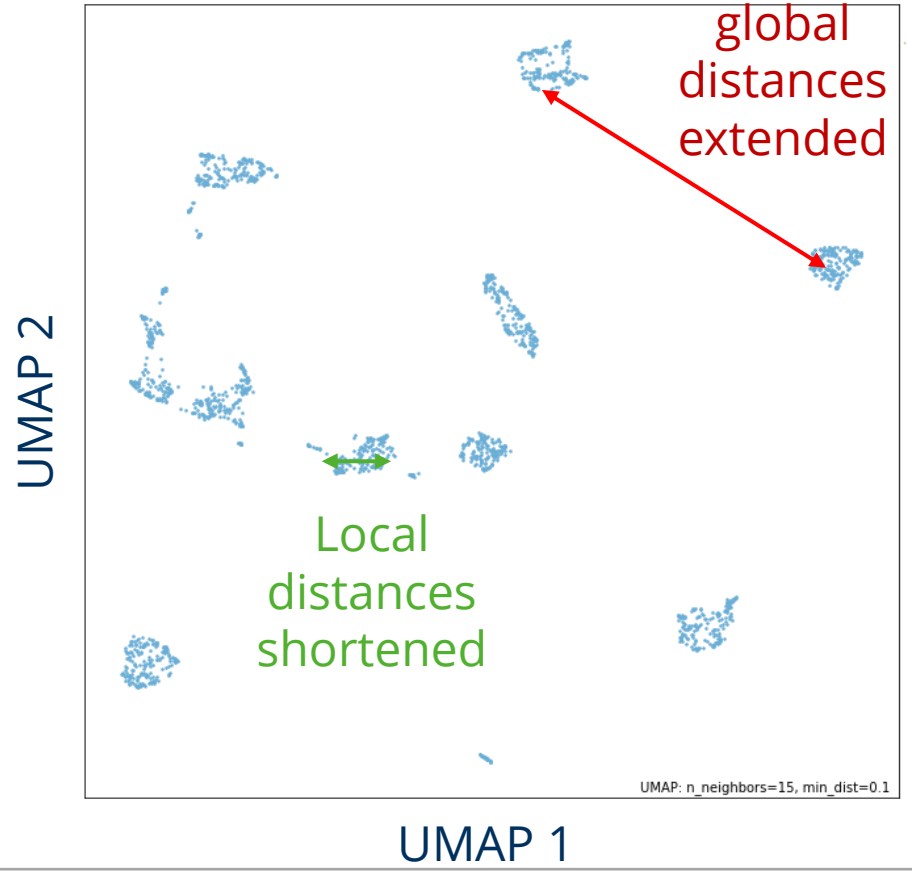
Structural, hierarchical, non-linear transformation

Modifies density of data points.

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

Many dimensions

UMAP

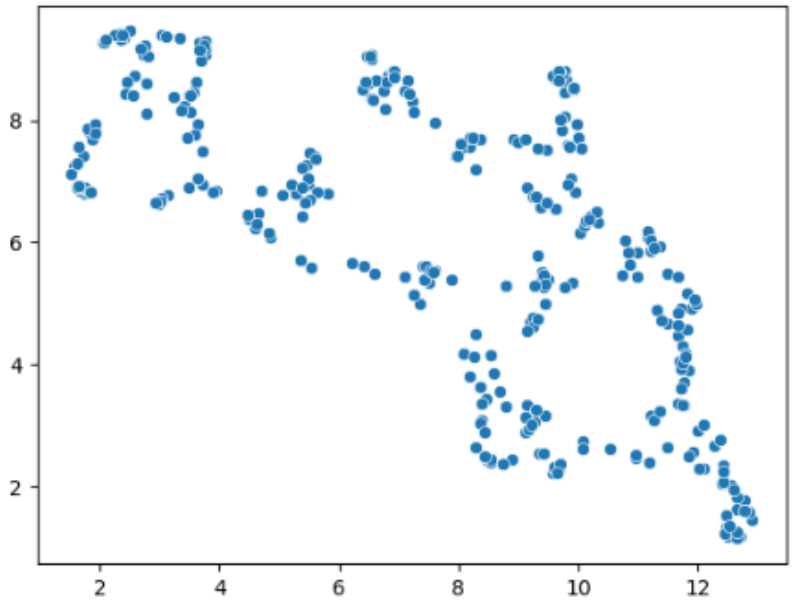


Uniform Manifold Approximation Projection (UMAP)

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

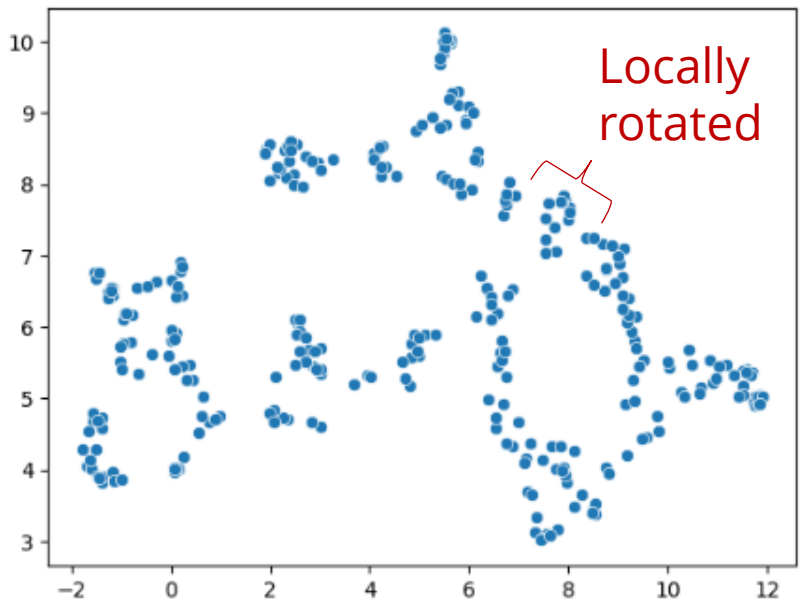
[11]:
```
reducer = umap.UMAP()
embedding2 = reducer.fit_transform(scaled_statistics)

seaborn.scatterplot(x=embedding2[:, 0],
                    y=embedding2[:, 1])
```

[11]: <AxesSubplot: >



[12]:
```
reducer = umap.UMAP()
embedding2 = reducer.fit_transform(scaled_statistics)

seaborn.scatterplot(x=embedding2[:, 0],
                    y=embedding2[:, 1])
```

[12]: <AxesSubplot: >



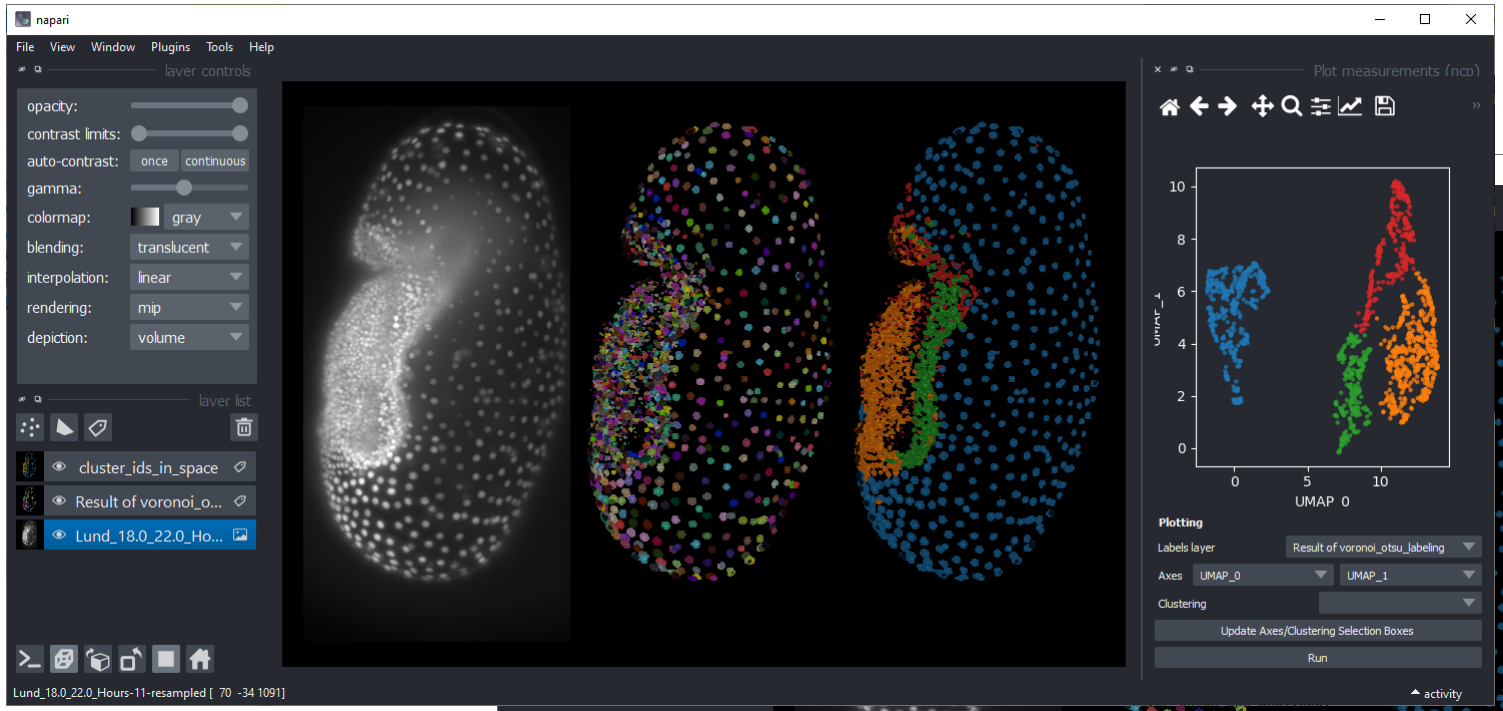
Globally rotated

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

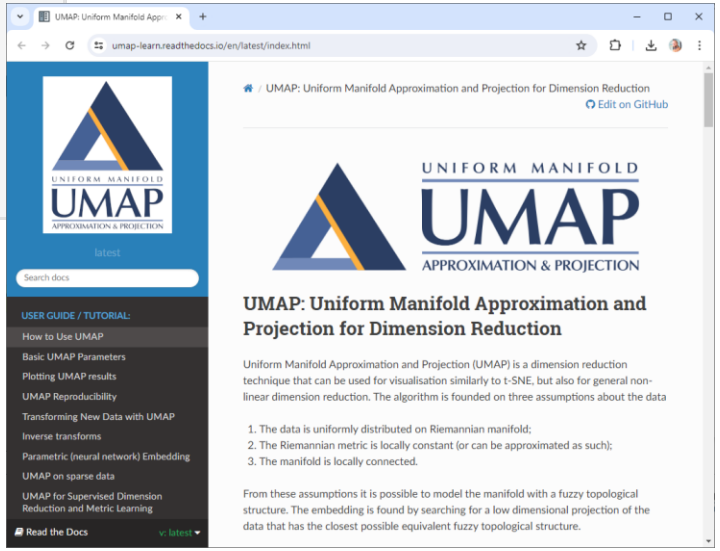
[9]:

	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

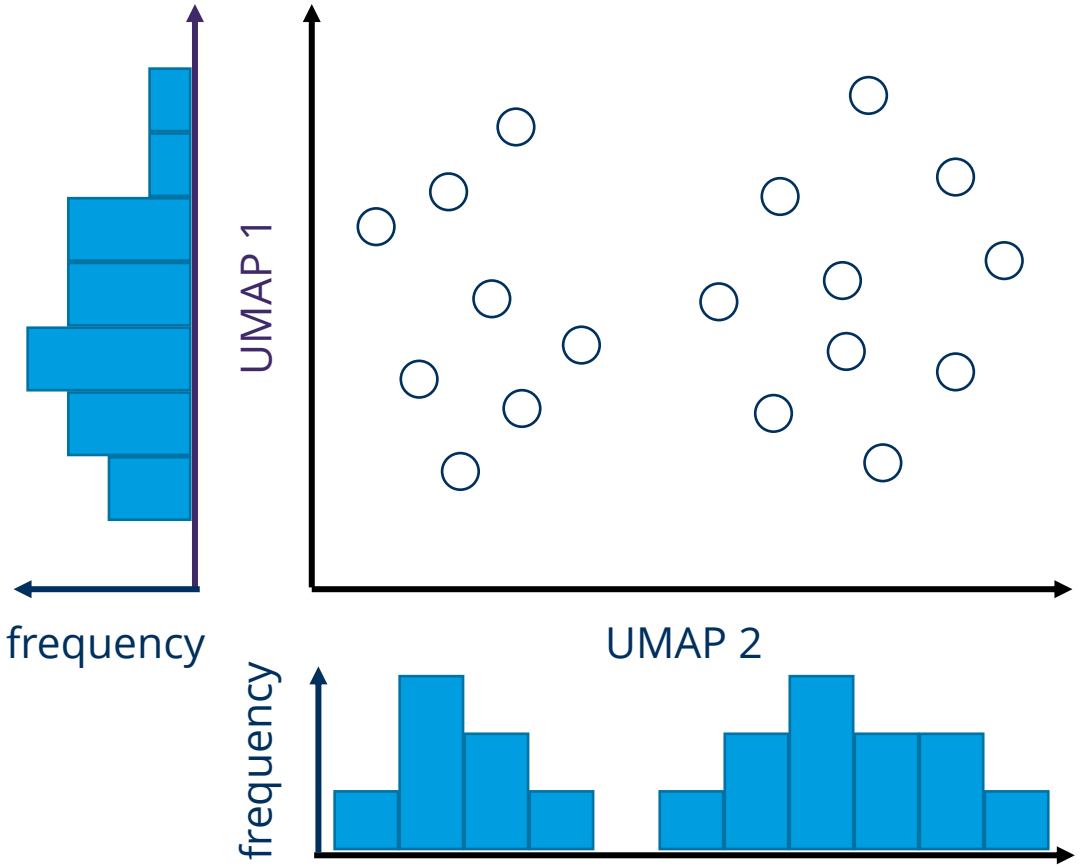
[10]:
```
# configure UMAP algorithm
umap = UMAP(n_neighbors=5, n_components=2)

# 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



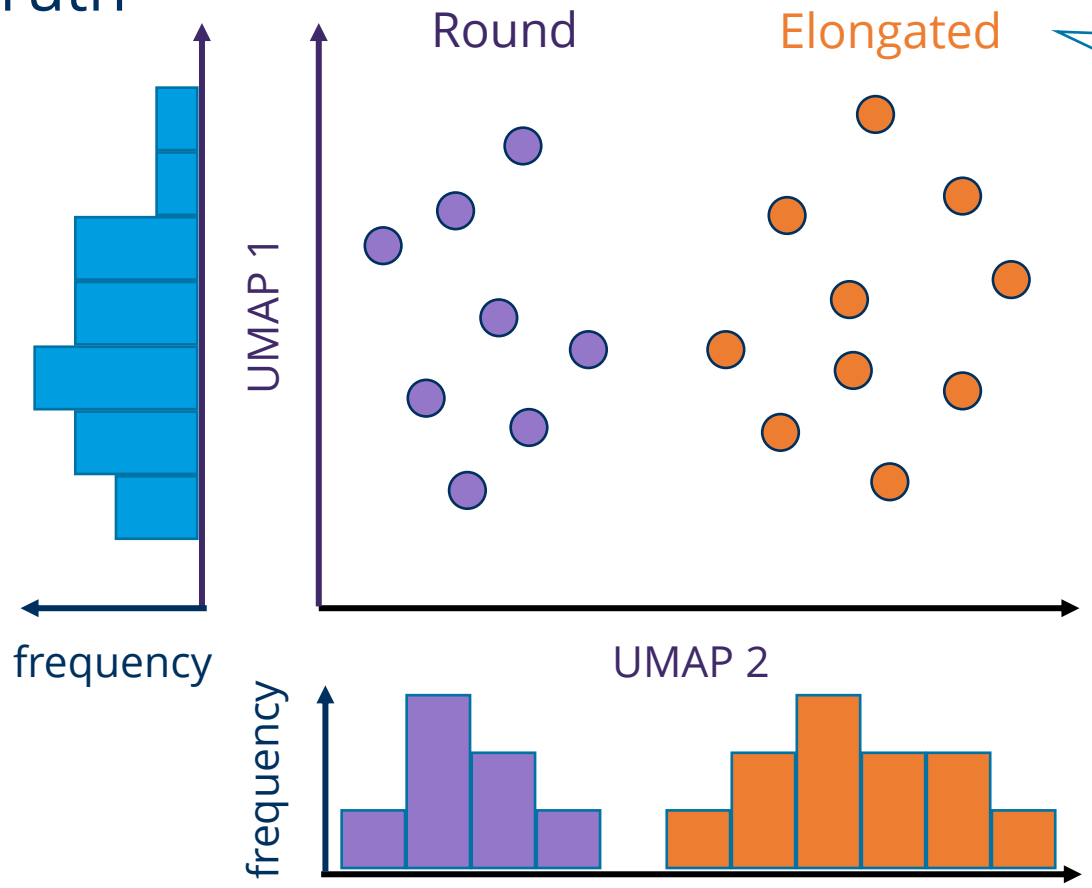
Clustering

Unsupervised machine learning may include grouping objects without given ground truth



Clustering

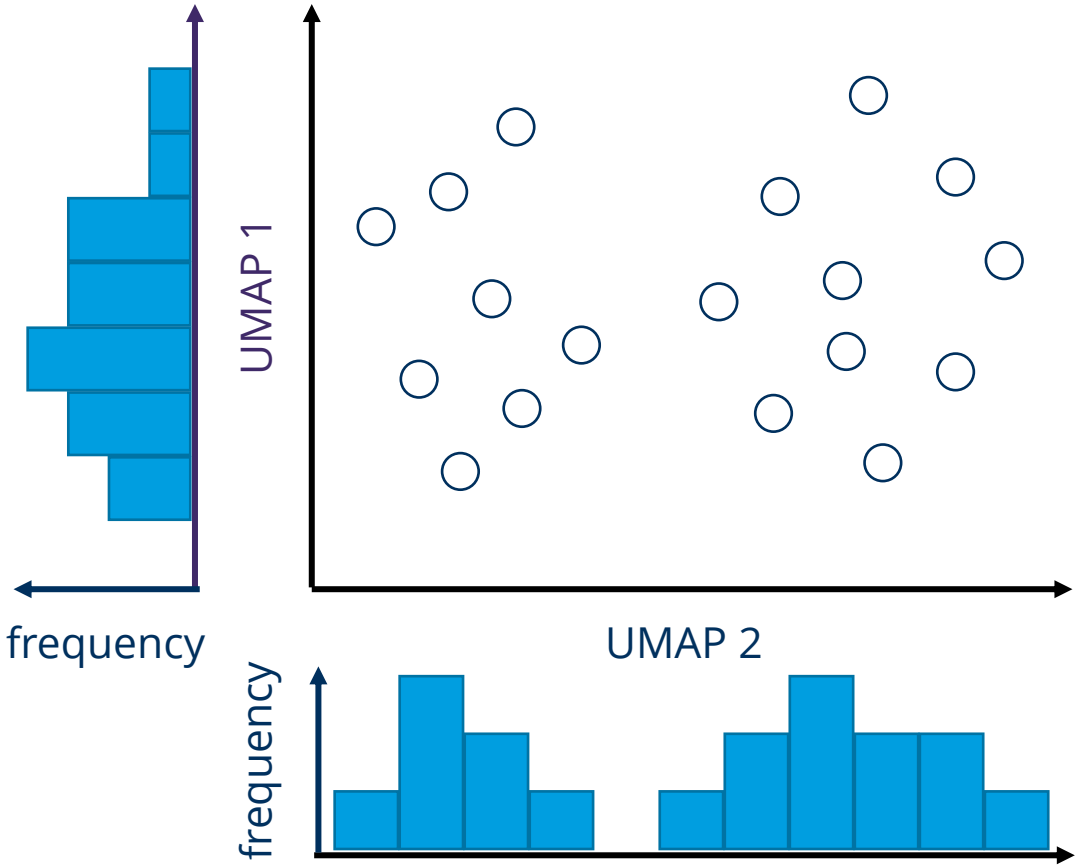
Unsupervised machine learning may include grouping objects without given ground truth



Names given by human observer after grouping / clustering

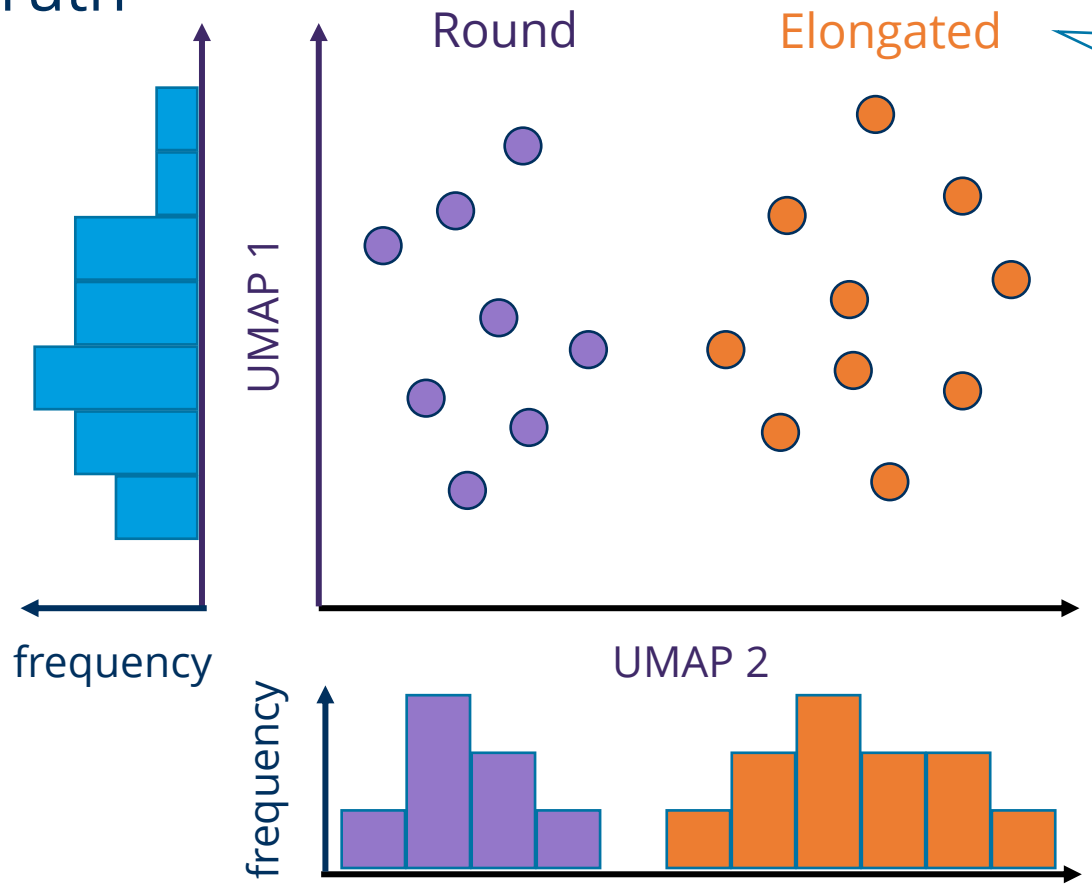
Clustering

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

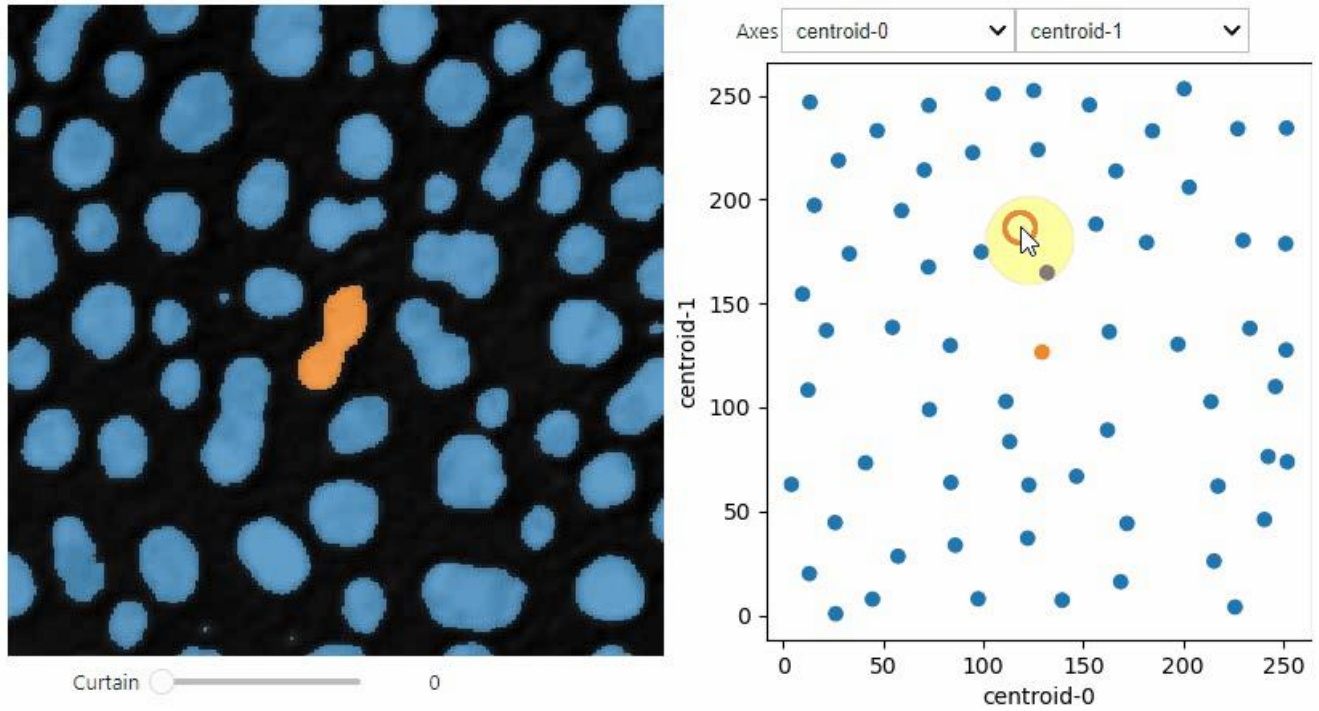
Clustering: Manual

[9]:

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

[6]:
```
stackview.clusterplot(image=image,
                      labels=labeled_image,
                      df=df,
                      column_x="area",
                      column_y="aspect_ratio",
                      zoom_factor=1.6,
                      alpha=0.7)
```

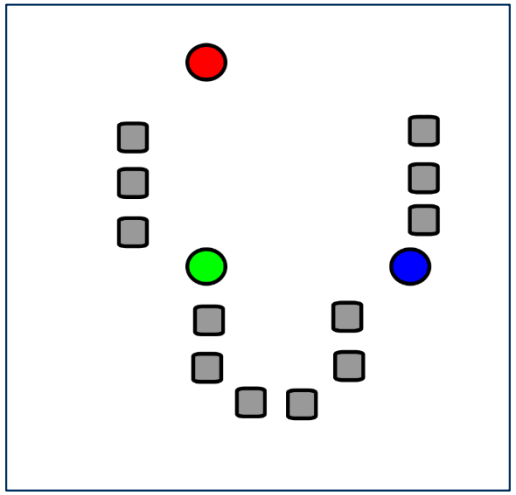
[6]:



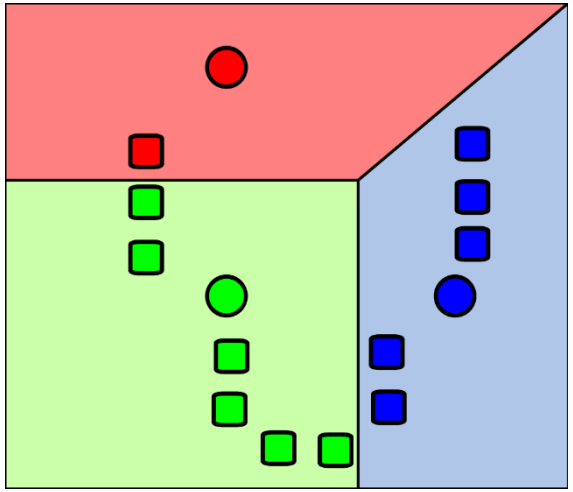
K-Means Clustering

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

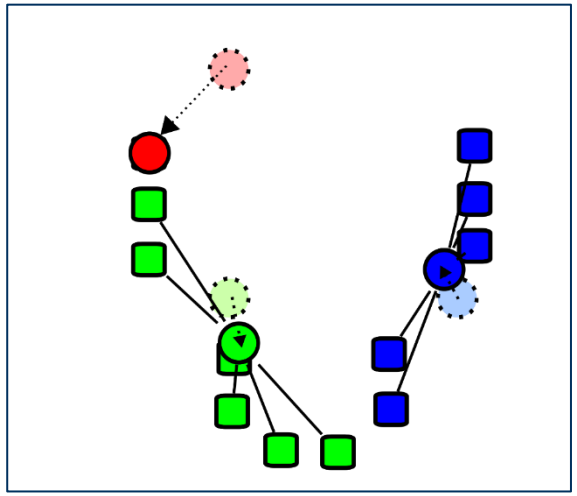
Step1: Random initialization of cluster centers



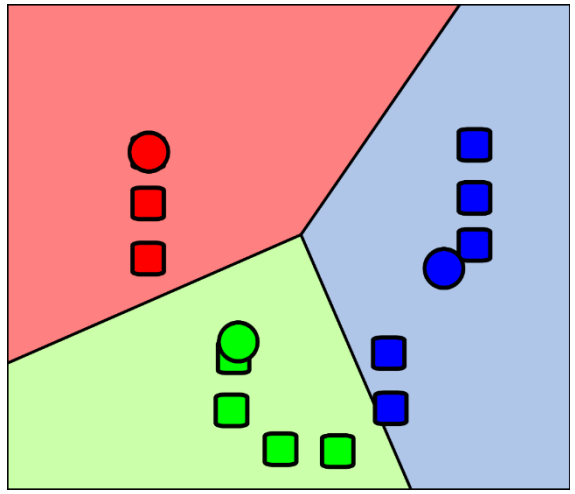
Step2: Tessellation of space into cluster regions



Step3: Replace cluster center with centroids

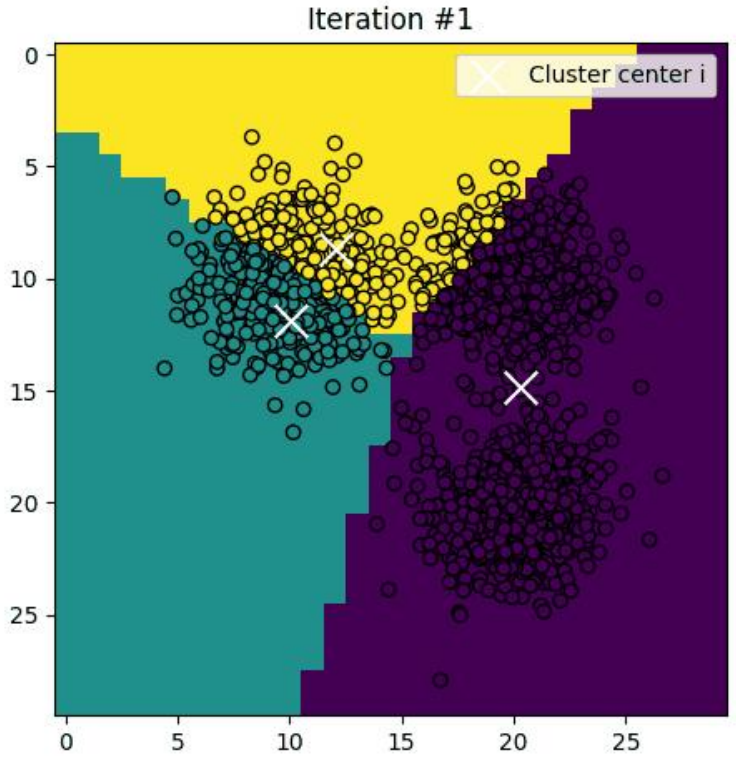


Step4: Repeat 2&3 until convergence



K-Means Clustering

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



K-Means Clustering

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

In Python:

```
from sklearn import cluster
```

Create

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

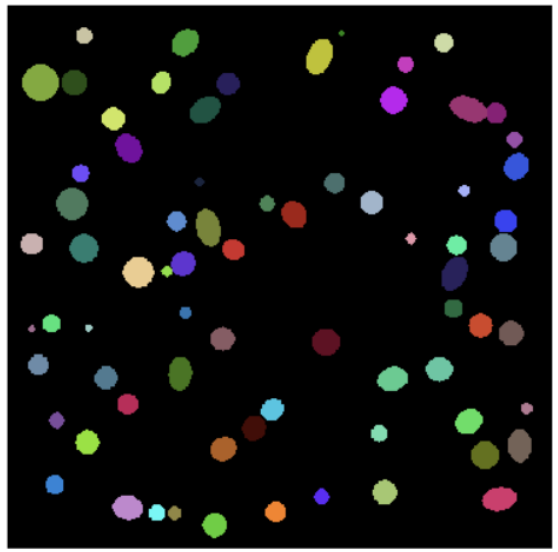
Predict

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

Walk-through: Data Exploration

Goal: Understand shape measurements

Data: Shape measurements from *randomly* shaped blobs.



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

...

Walk-through: Data Exploration

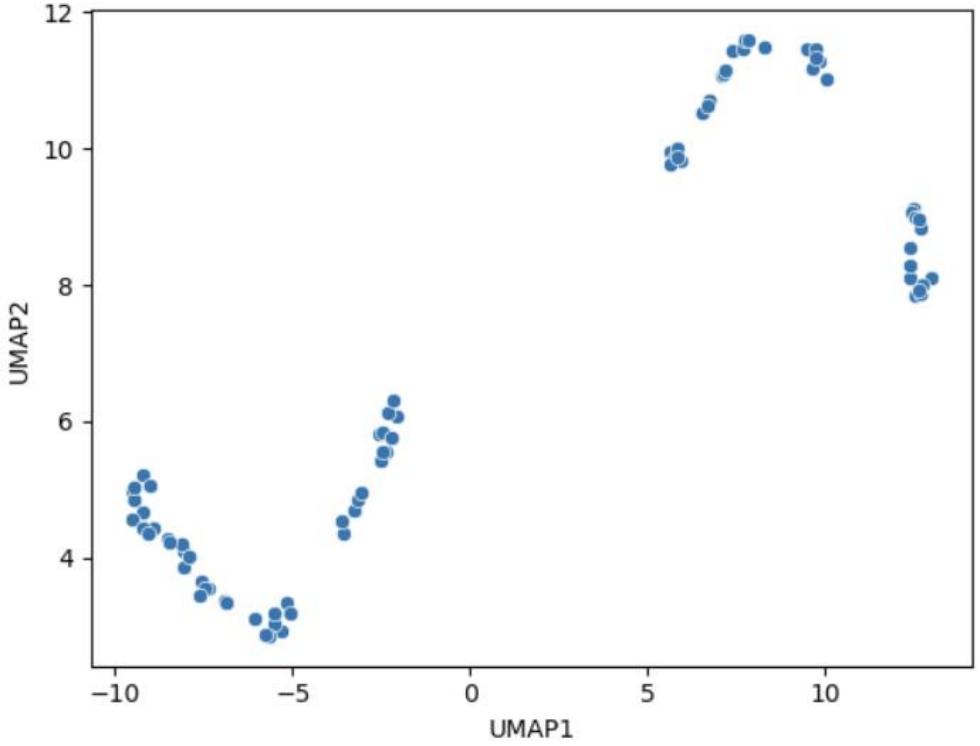
Step 1: Dimensionality reduction (UMAP)

Observation: There appear to be *2 distinct groups*

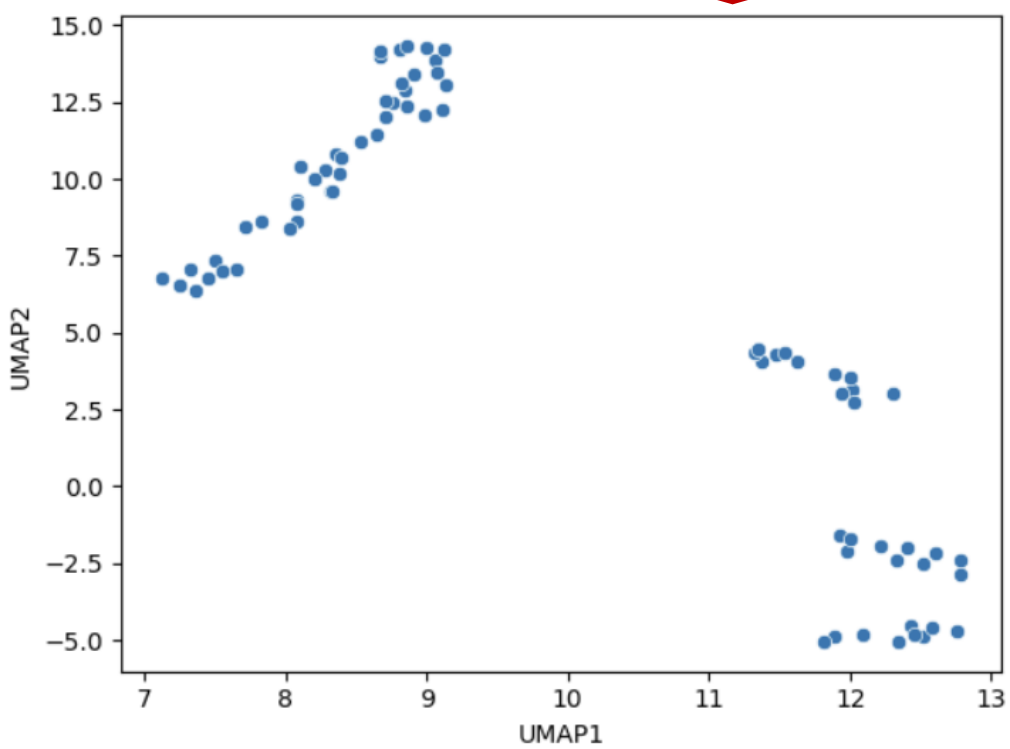
Pinning the random seed is no solution to this general problem.

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

Run 1:



Run 2:



Walk-through: Data Exploration

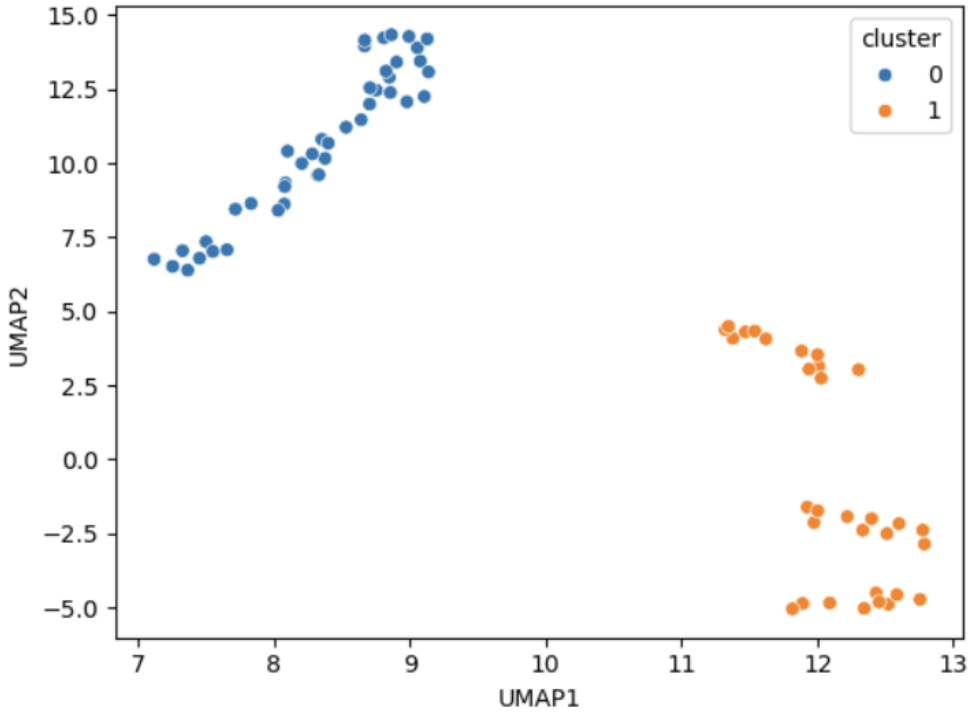
Step 2: Clustering data into 2 clusters

Using K-Means clustering

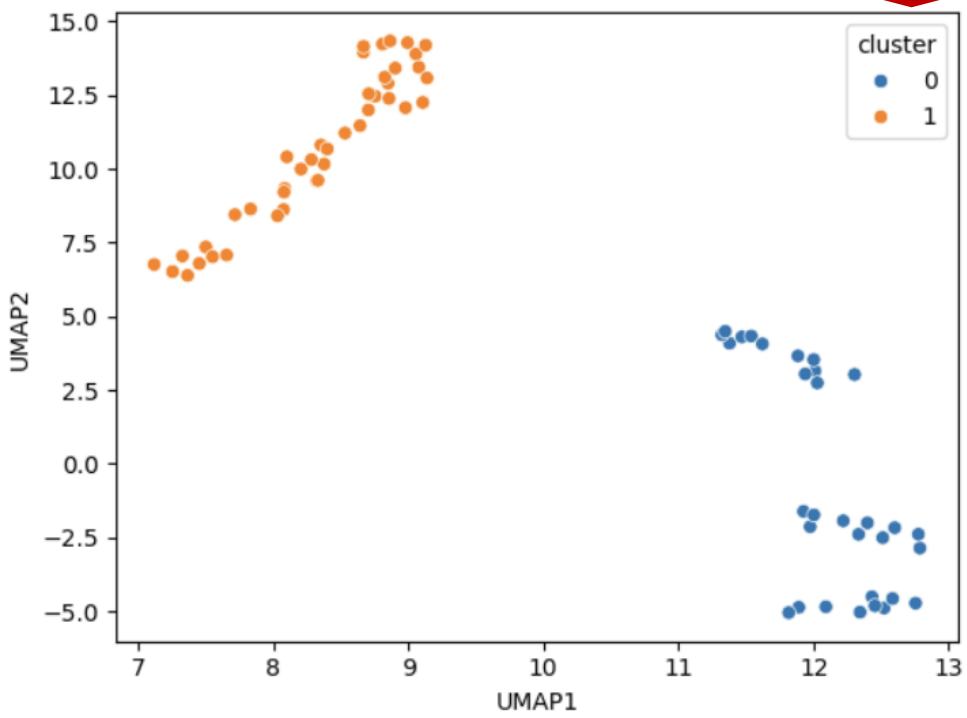
Pinning the random seed is no solution to this general problem.

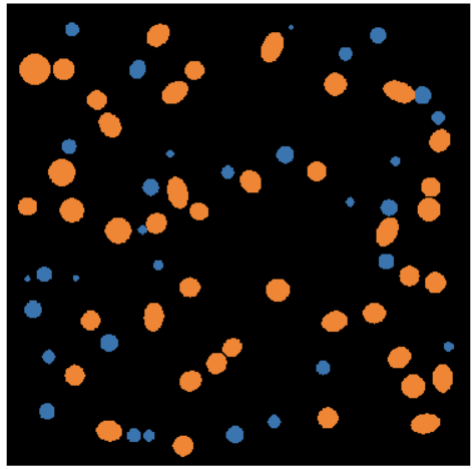
Beware: Clustering-algorithms are non-deterministic. Different runs lead to different results.

Run 1:



Run 2:



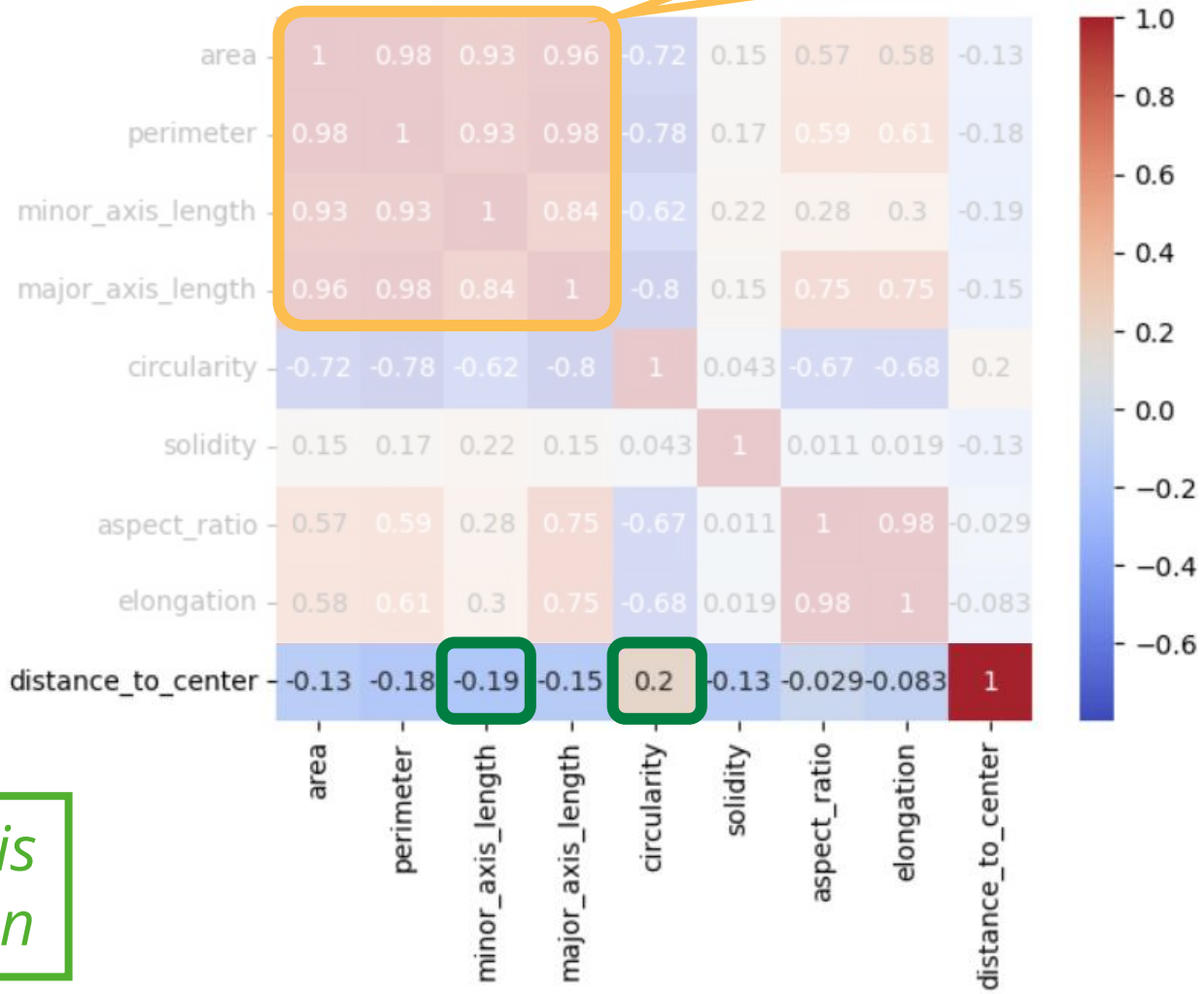


Walk-through: Data Exploration

Step 3: Feature selection

Based on correlation
with distance to cluster-centers

Side note: beware of feature correlation.



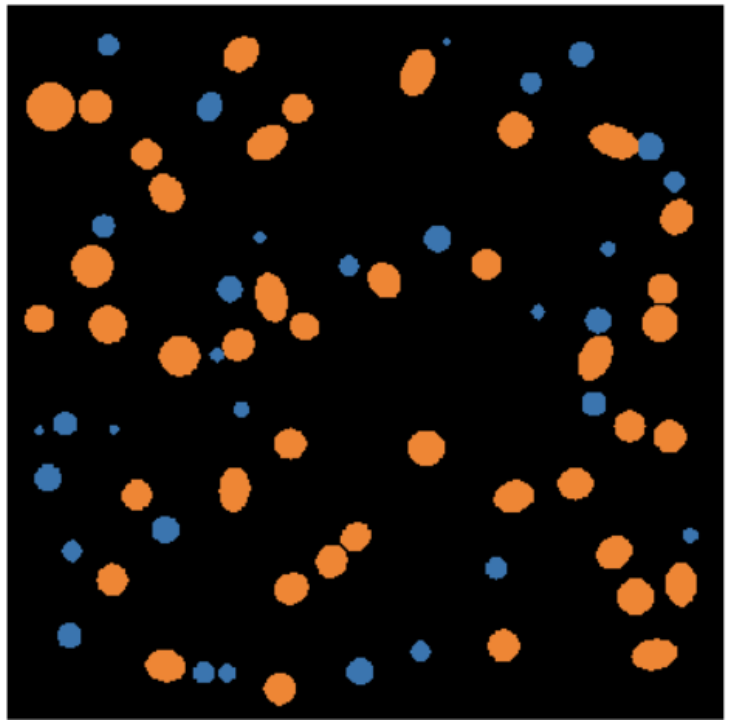
Hypothesis:
“Circularity and minor_axis_length allow to predict round vs. elongated classification.”

Hypothesis generation

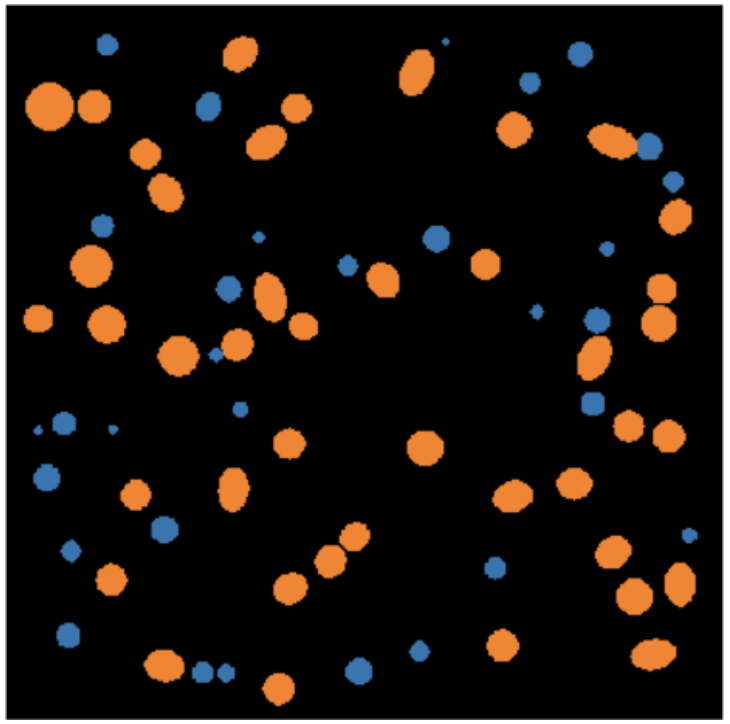
Walk-through: Data Exploration

Step 4: Train a classifier (supervised ML)

Goal: Eliminate non-determinism



Clustering result (non-deterministic)



Classification result (deterministic, repeatable)

ScaDS.AI
DRESDEN LEIPZIG

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.

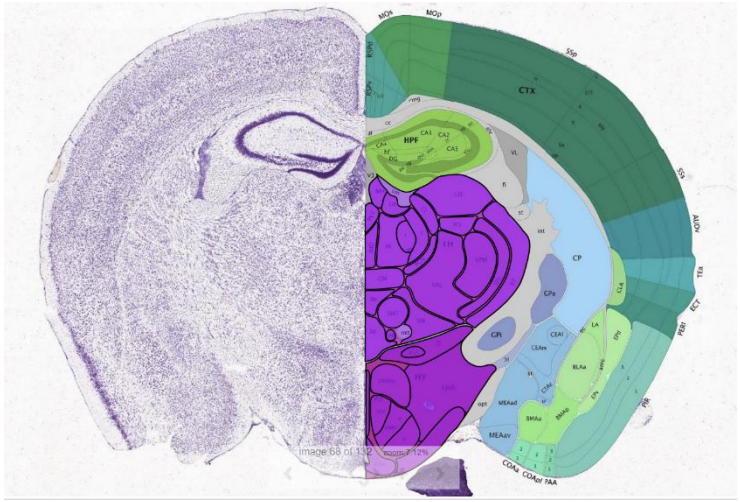
Recap: Terminology

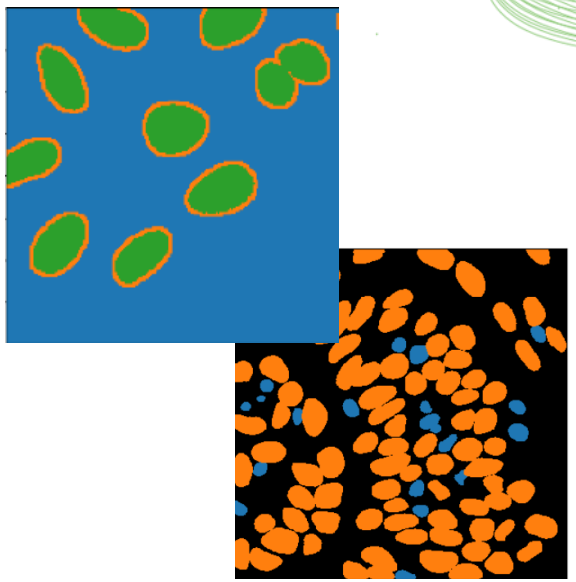
Instance segmentation

Instances:

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

Semantic segmentation





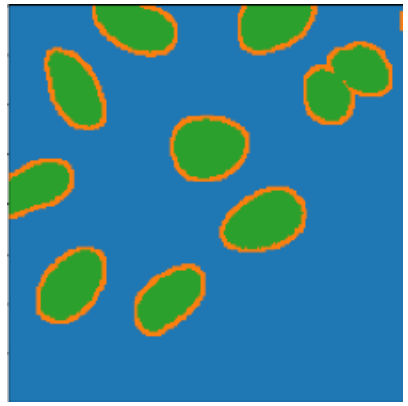
Regions:

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

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

Recap: Terminology

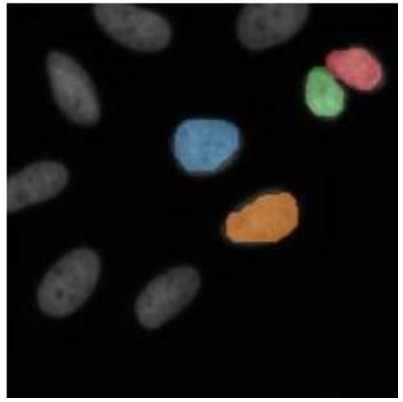
Instance segmentation

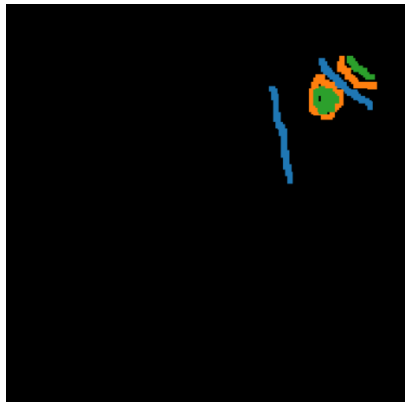


Semantic segmentation

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

Sparse instance annotation





Sparse semantic annotation

Image segmentation using thresholding

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

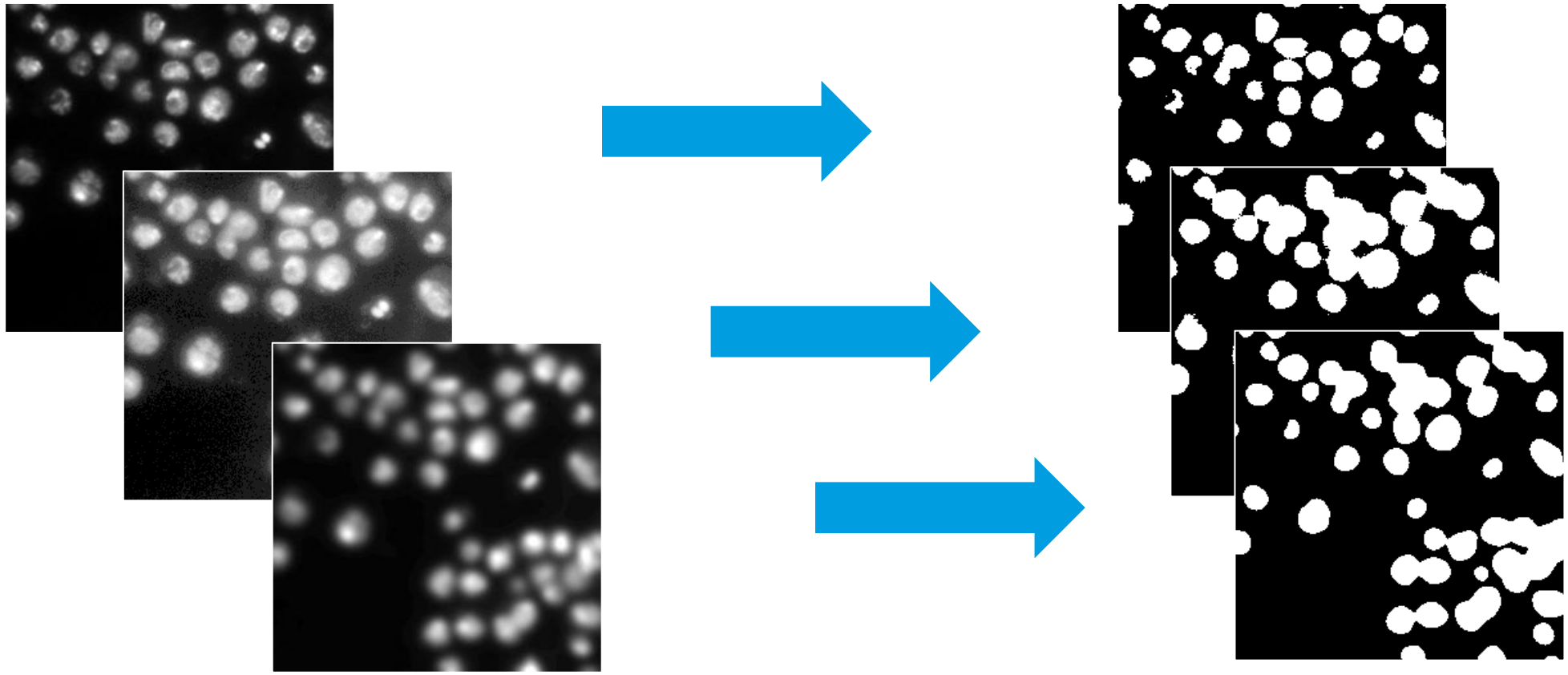


Image data source: BBBC039v1, available from the Broad Bioimage Benchmark Collection (Caicedo et al., Nature Methods, 2019].

Image segmentation using thresholding

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

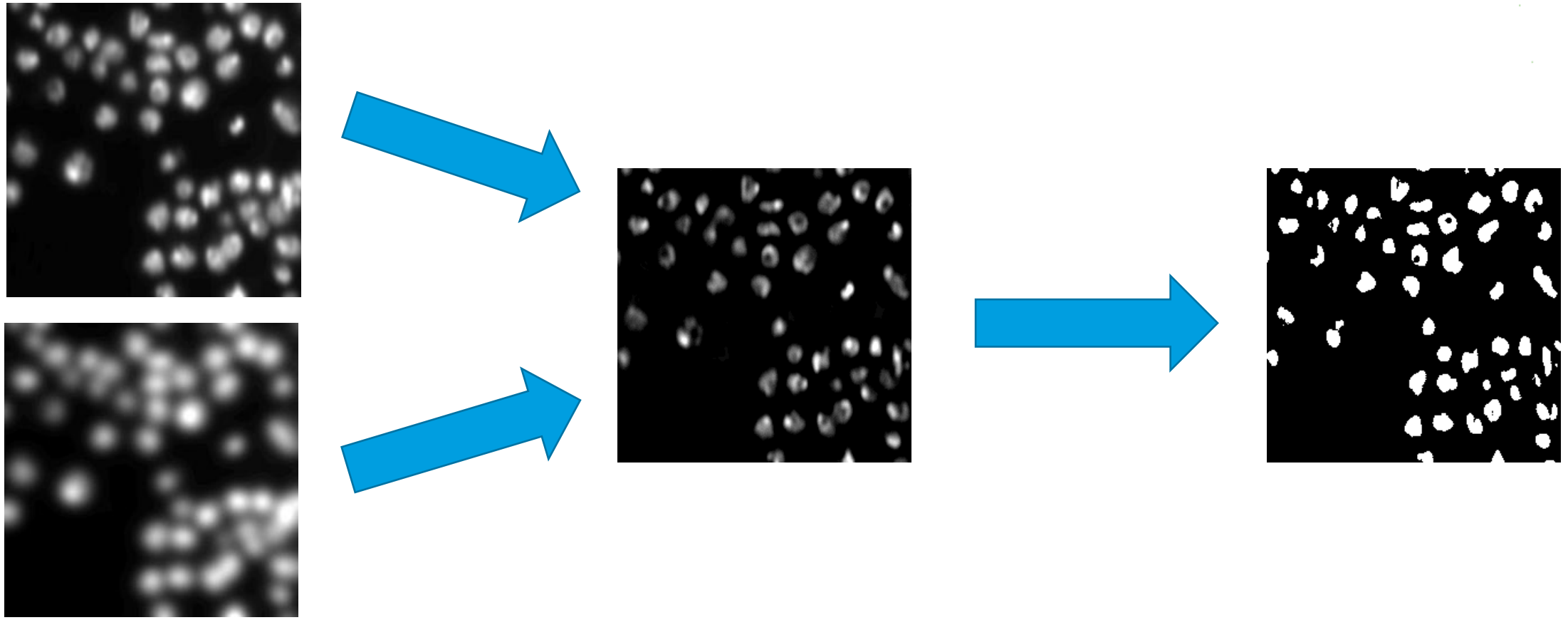


Image data source: BBBC039v1, available from the Broad Bioimage Benchmark Collection (Caicedo et al., Nature Methods, 2019].

Image segmentation using thresholding

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

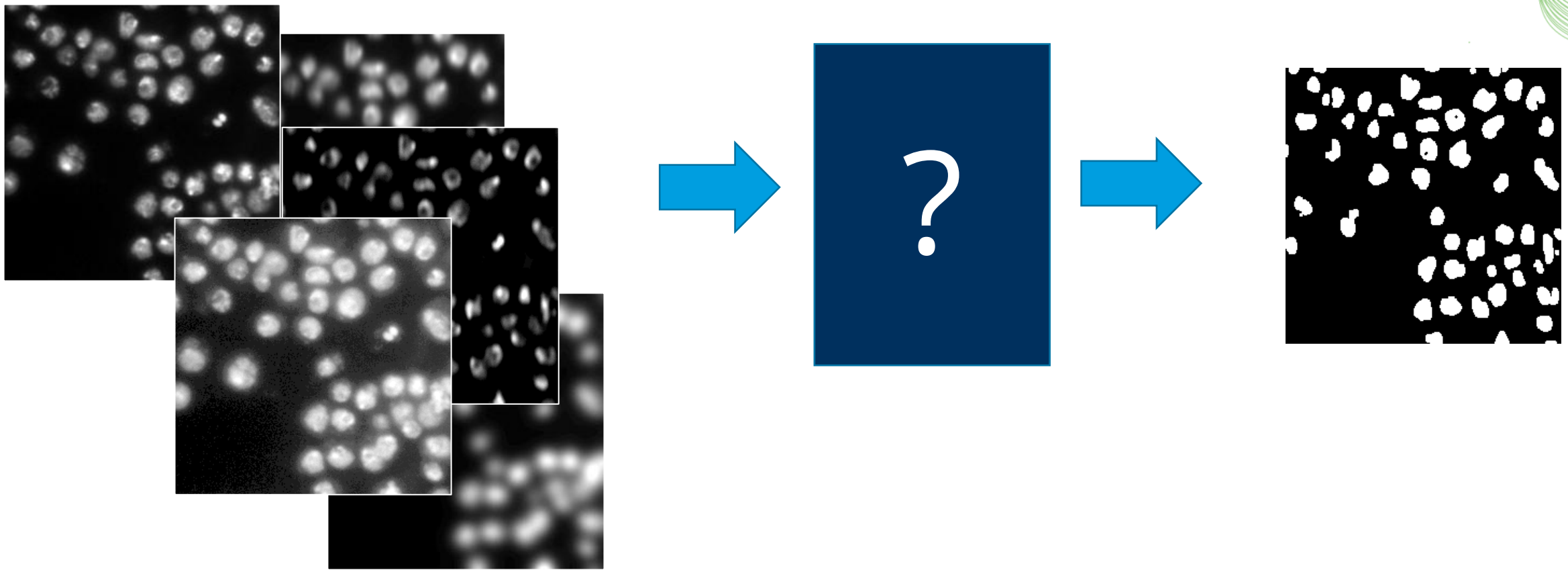
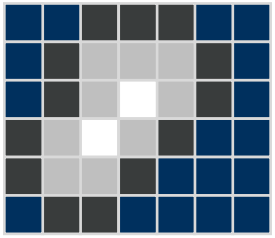
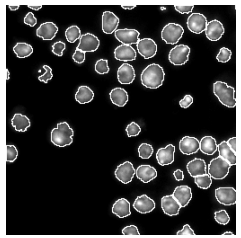


Image data source: BBBC039v1, available from the Broad Bioimage Benchmark Collection (Caicedo et al., Nature Methods, 2019].

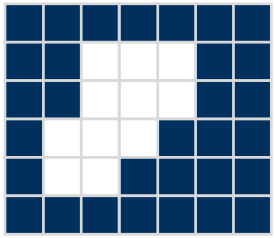
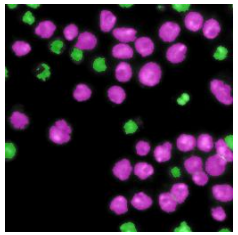
Supervised Machine learning

Automatic construction of predictive models from given data

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

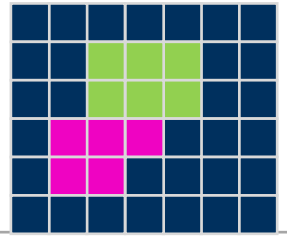




Semantic Segmentation | Object classification | Image classification

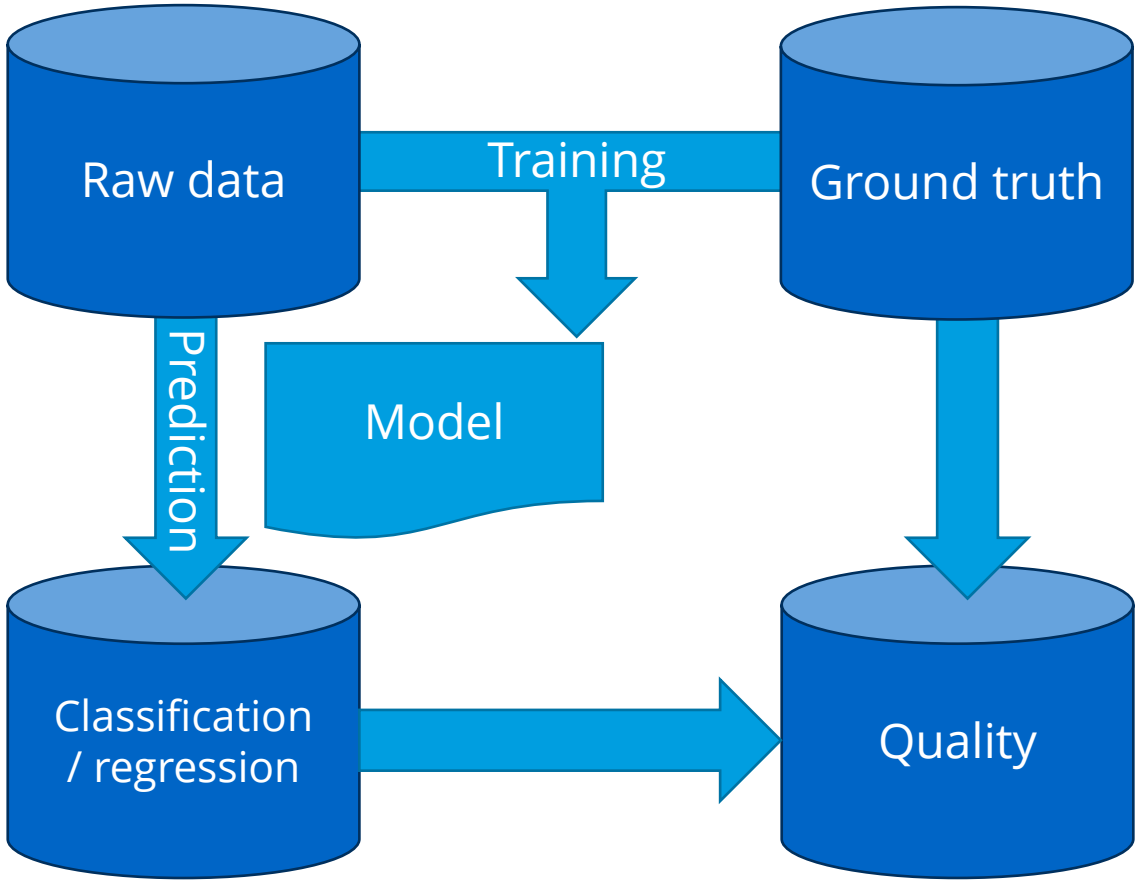



"Cat"

Instance segmentation | Cont. quantity



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



Annotated raw data, usually generated by humans

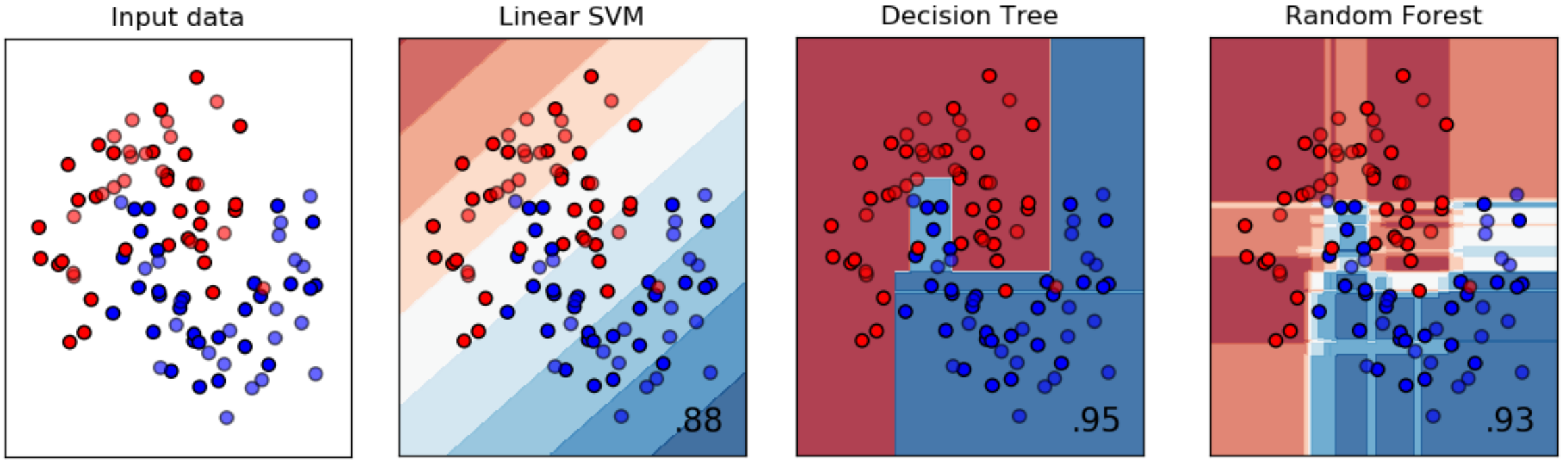
Precision, Recall

Quiz

What is the difference between classification and regression?

Goal

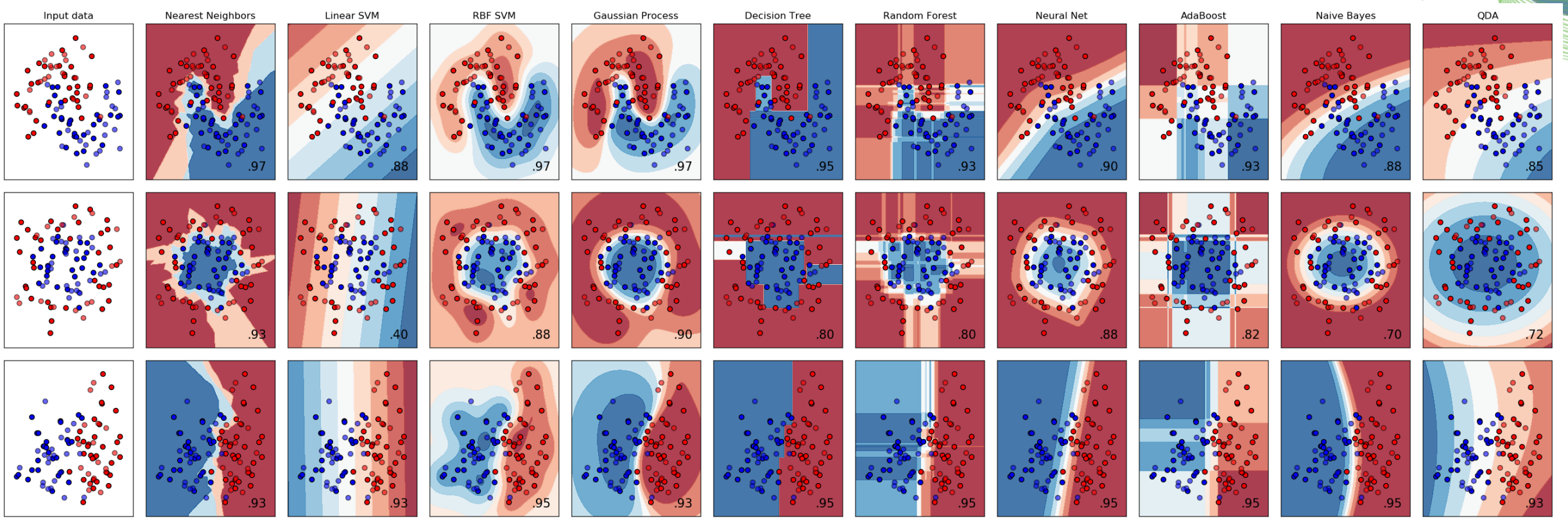
Guess classification (color) from position of a sample in parameter space.



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



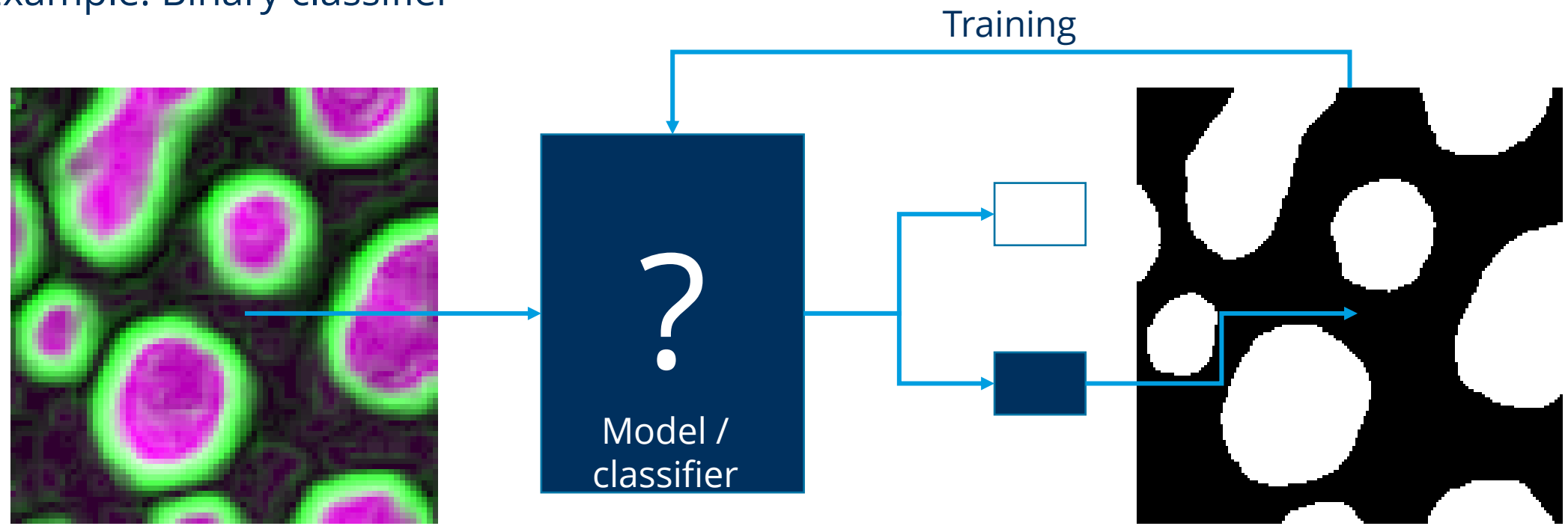
Adapted from https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html

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



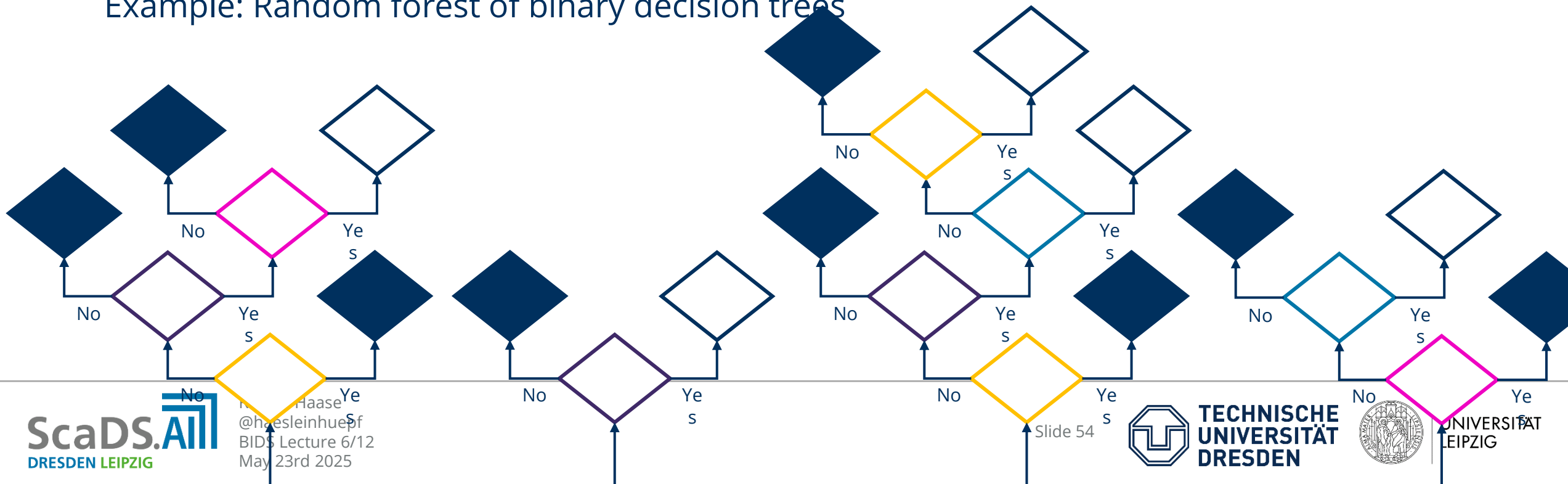
Random forest based image segmentation

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

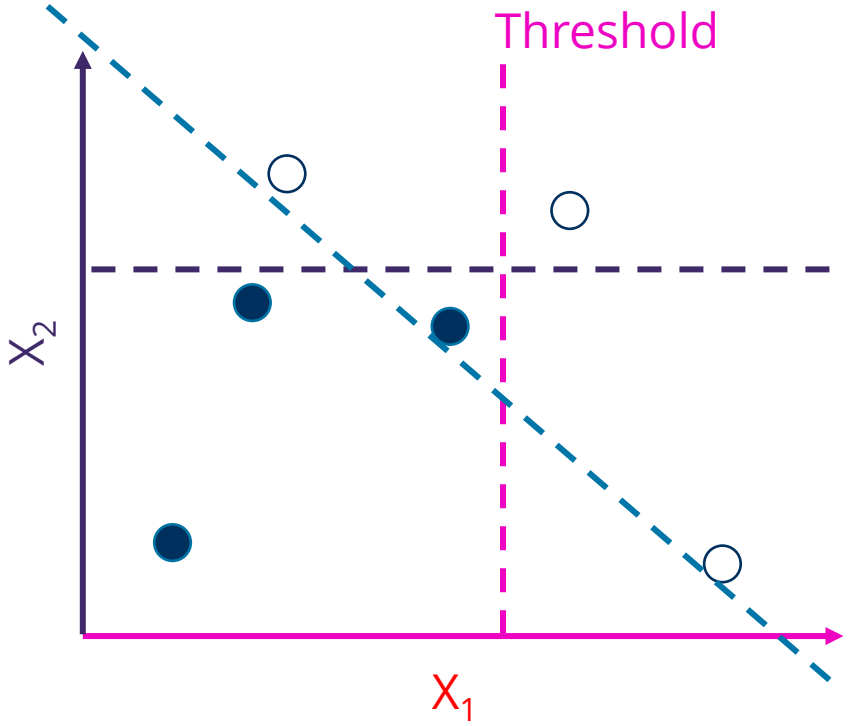
Example: Random forest of binary decision trees

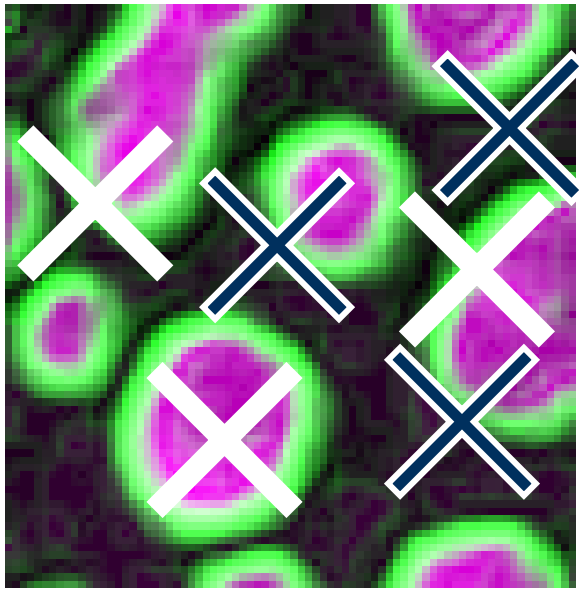


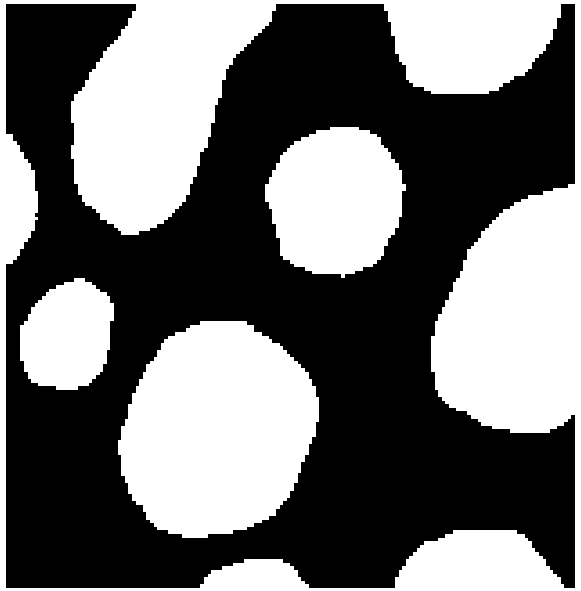
Deriving random decision trees

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

- Individual pixels, their intensity and their classification



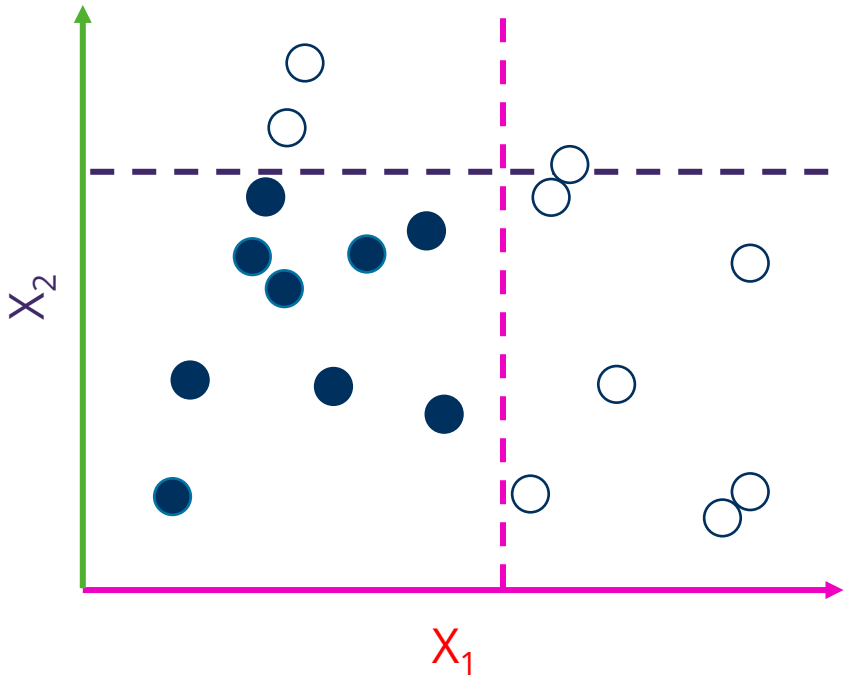


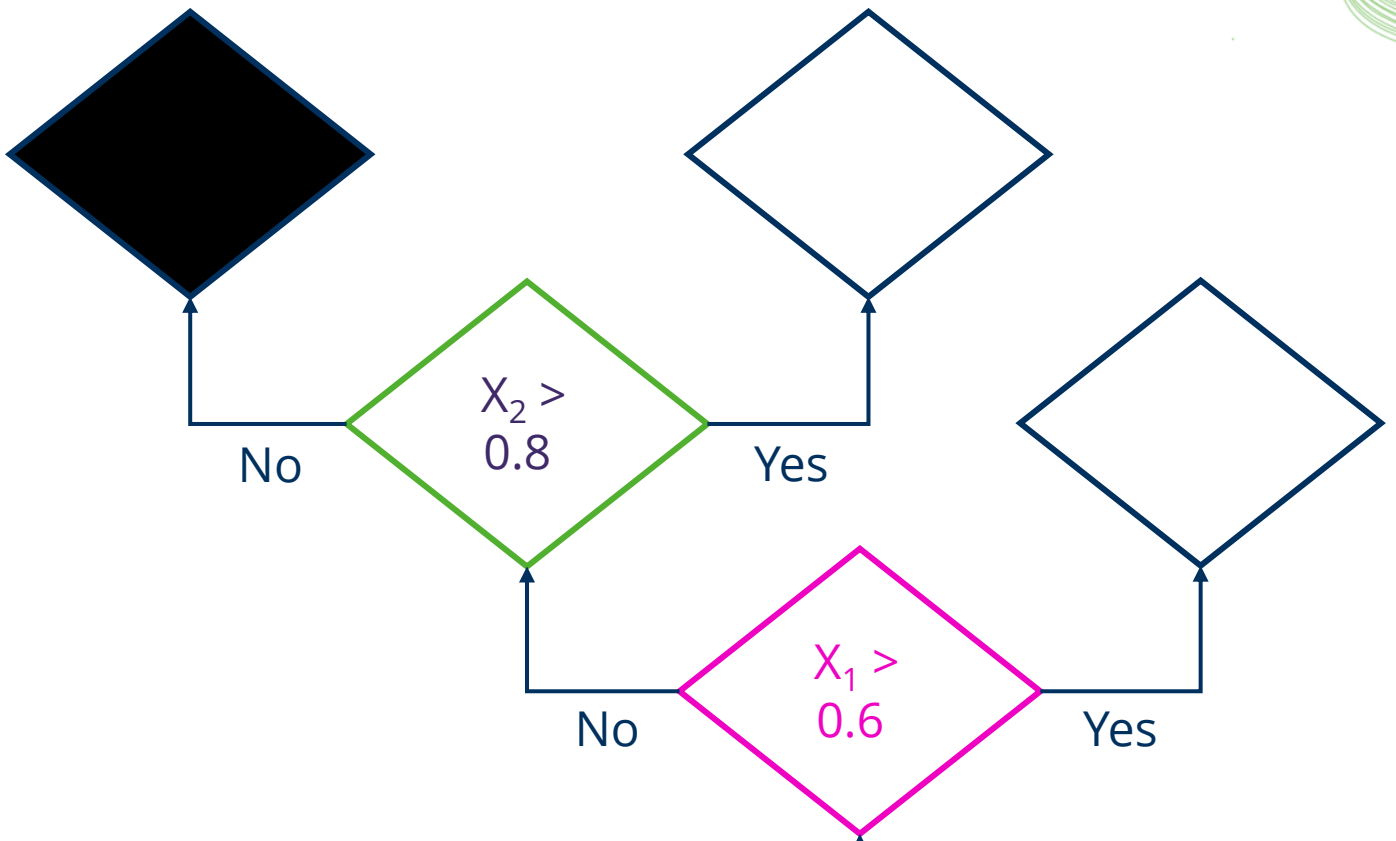


Note: You cannot use a single threshold to make the decision correctly

Deriving random decision trees

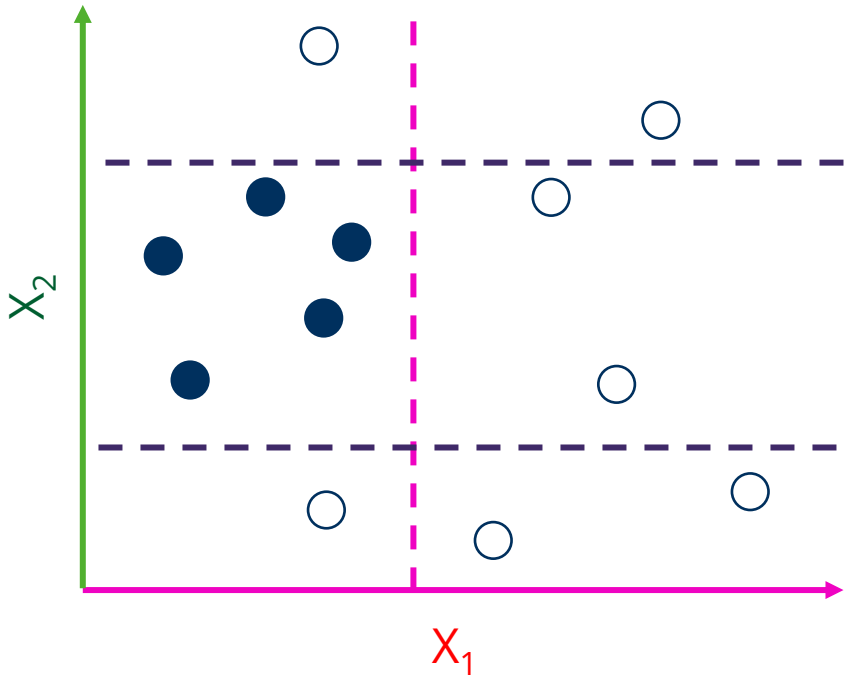
Decision trees combine several thresholds on several parameters

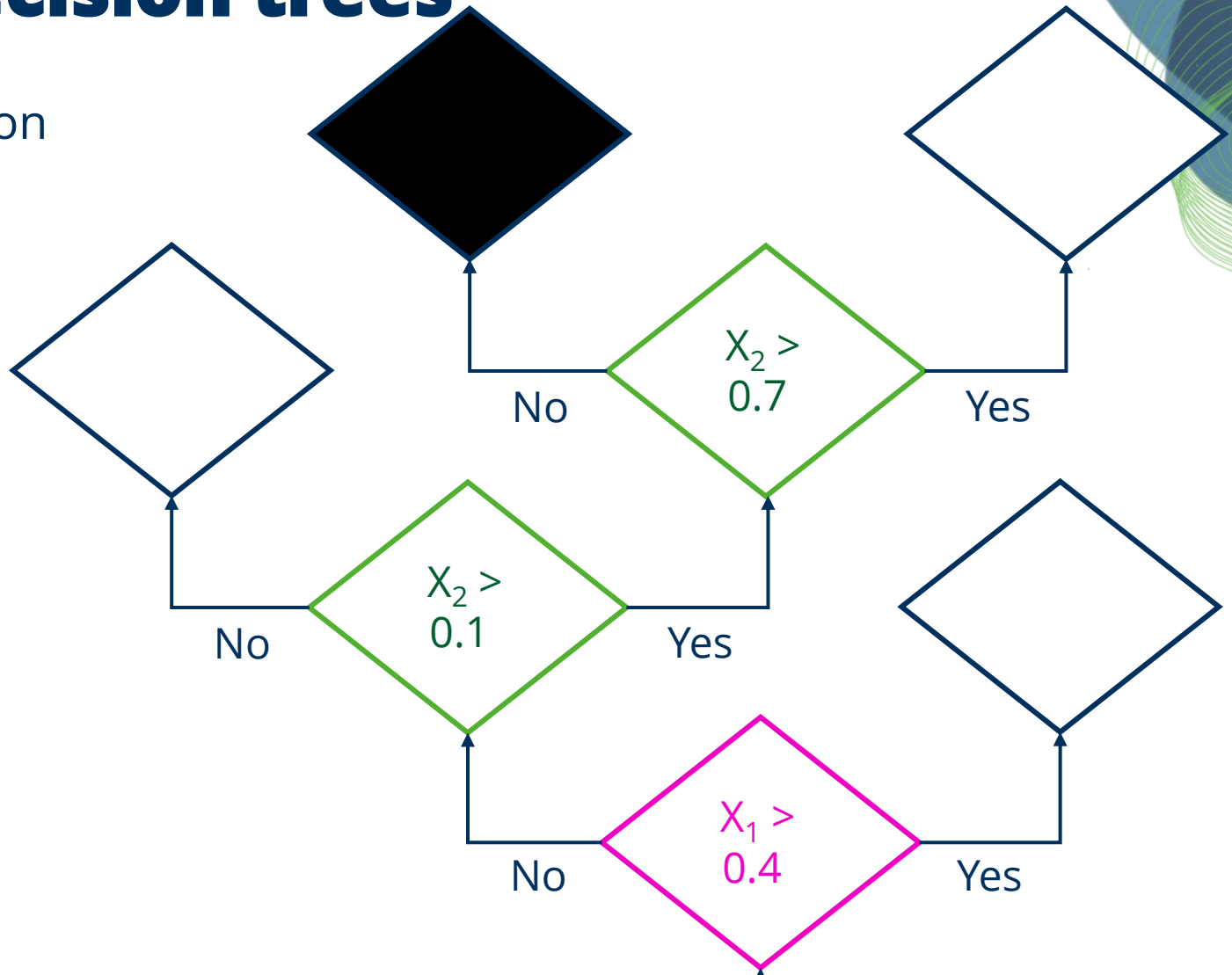




Deriving random decision trees

Depending on sampling, the decision trees are different





Random Forest Pixel Classifiers

By training many decision trees, errors are equilibrated

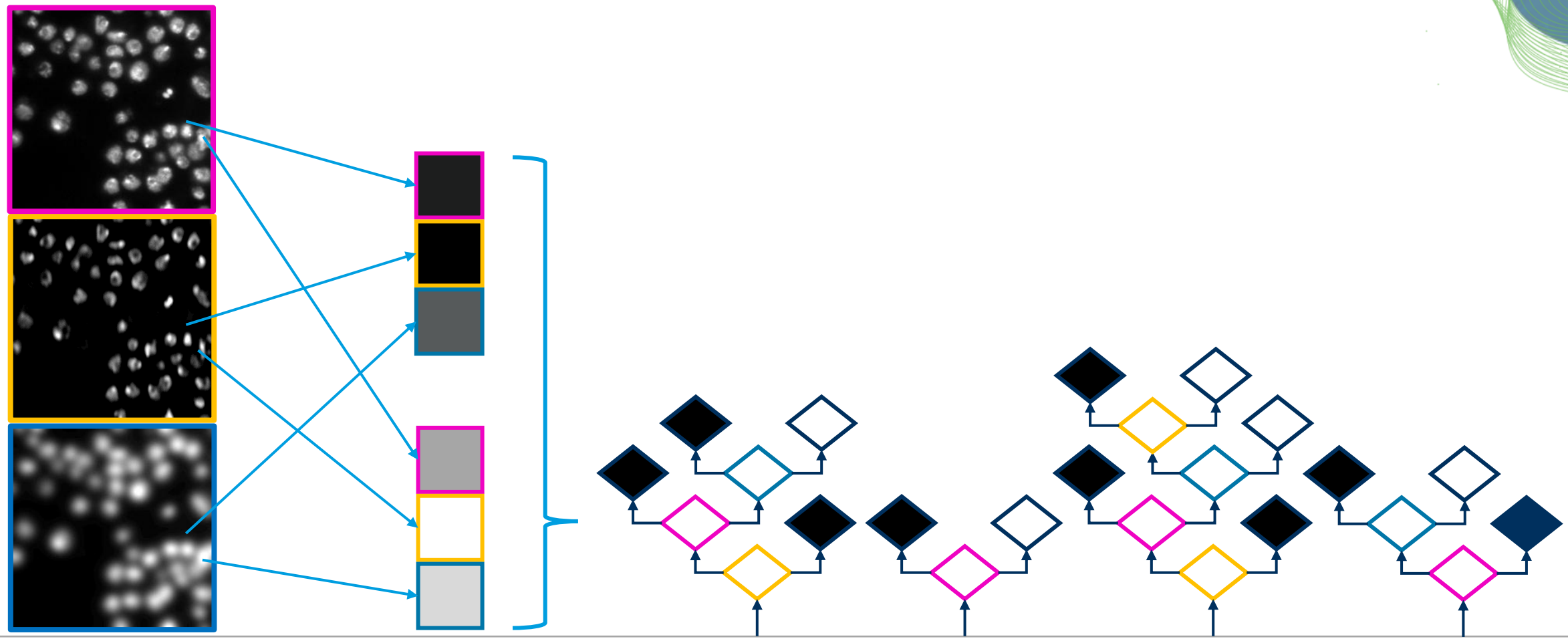
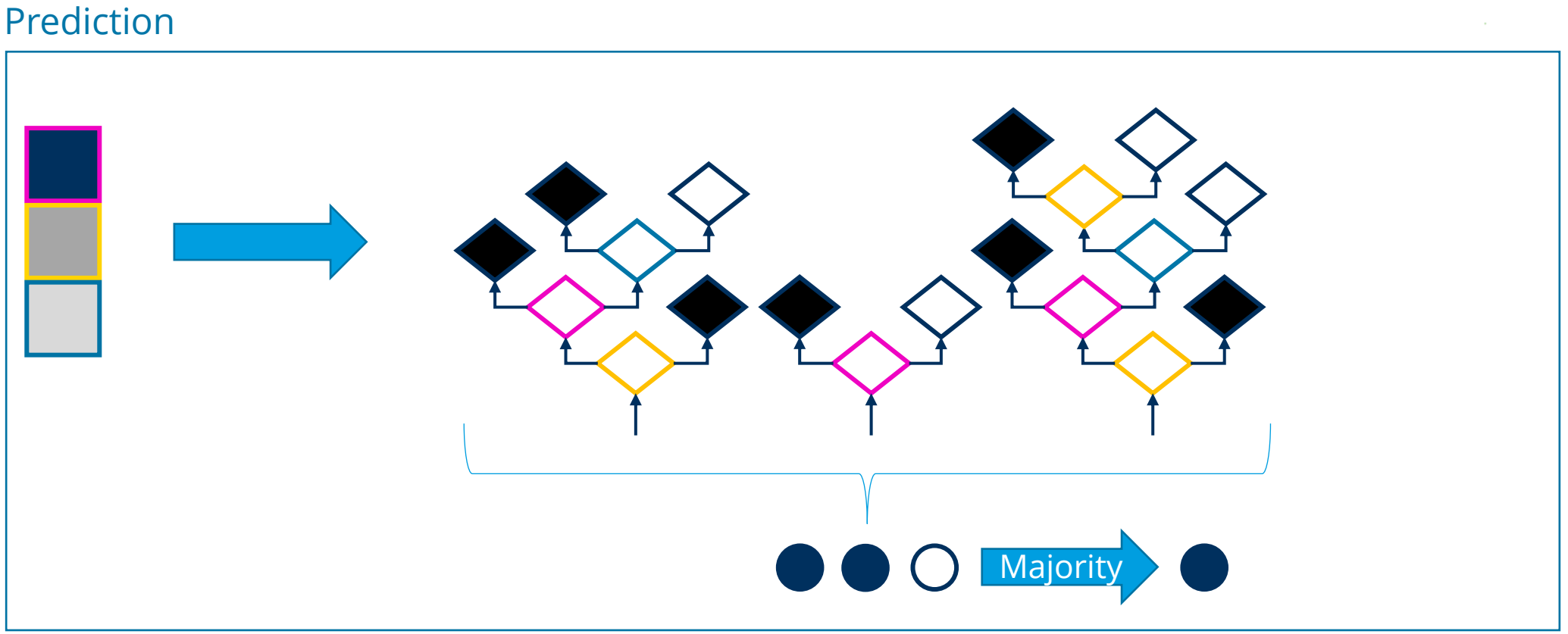


Image data source: BBBC038v1, available from the Broad Bioimage Benchmark Collection (Caicedo et al., Nature Methods, 2019)

Random Forest Pixel Classifiers

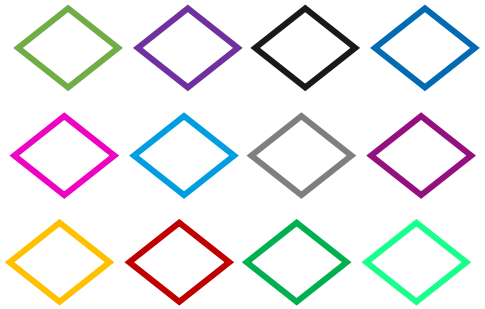
Combination of individual tree decisions by voting or max / mean



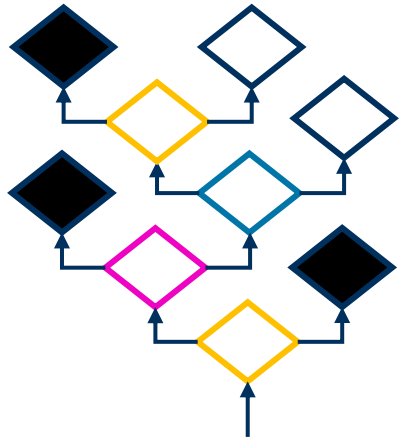
Random Forest Pixel Classifiers

Typical numbers for pixel classifiers in microscopy

Available features:

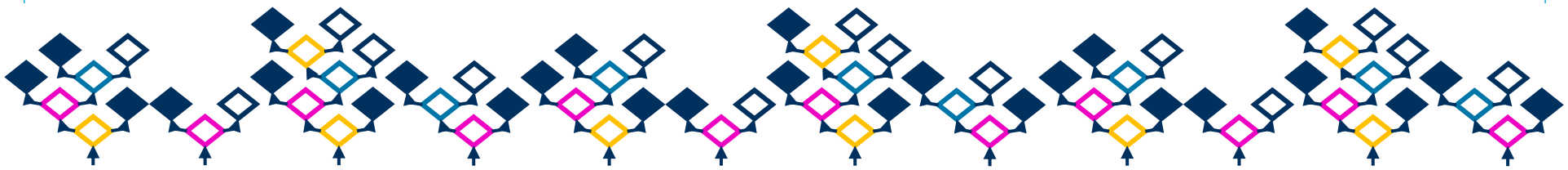


- Gaussian blur image
- DoG image
- LoG image
- Hessian
- ....



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)

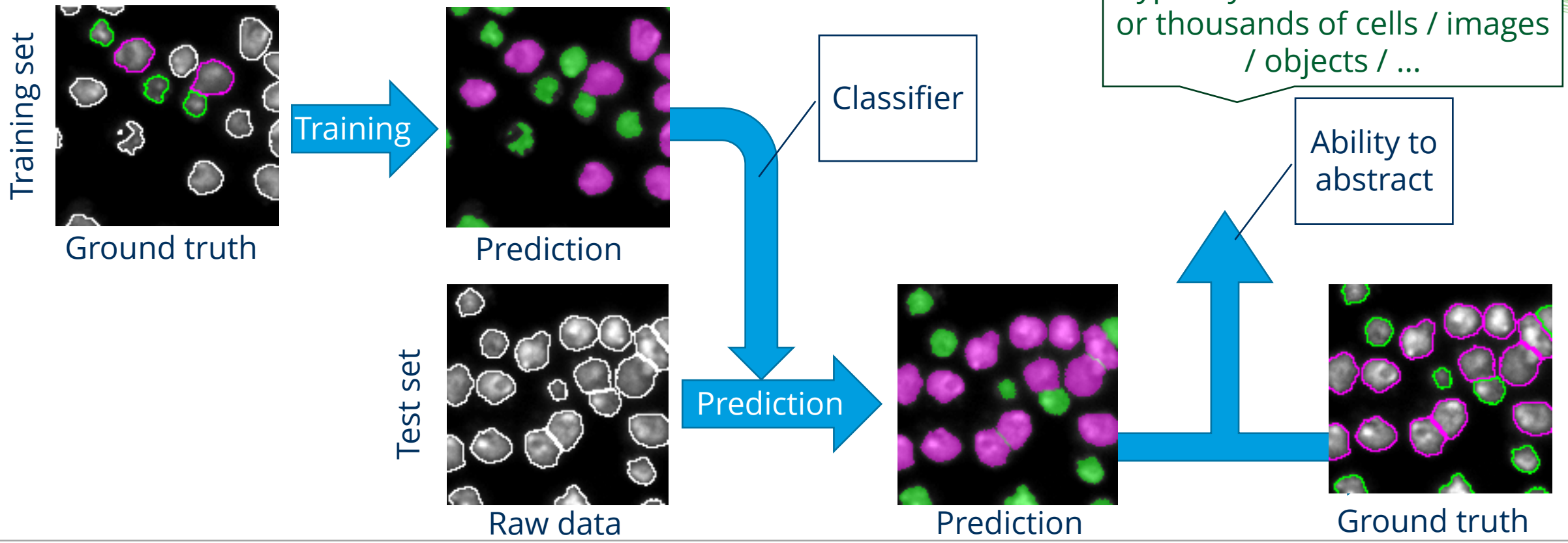


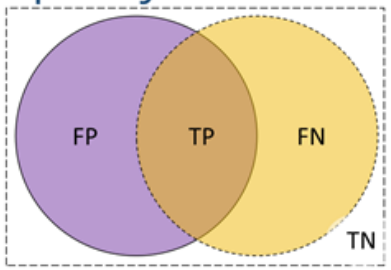
Image data source: BBBC038v1, available from the Broad Bioimage Benchmark Collection (Caicedo et al., Nature Methods, 2019)

Model validation

Recap

Segmentation quality estimation

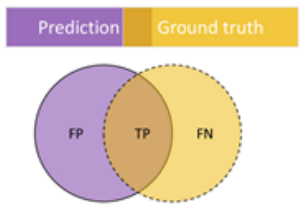
- In general
  - Define what's positive and what's negative.
  - Compare with a reference to figure out what was true and false
  - Welcome to the Theory of Sets



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

Pixel-wise versus Object-wise evaluation

- Pixel wise: Segmentation quality



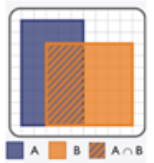
$$IoU(A,B) = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} = \frac{|A \cap B|}{|A \cup B|}$$

True-positive: 4

False-negative: 5

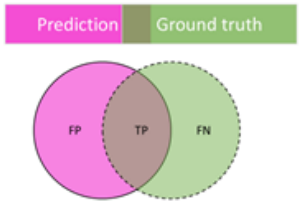
False-positive: 2

IoU = 4 / 11



Pixel-wise versus Object-wise evaluation

- Object wise: Detection quality



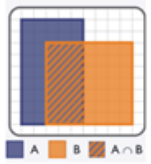
$$IoU(A,B) = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} = \frac{|A \cap B|}{|A \cup B|}$$

True-positive: 3

False-negative: 1

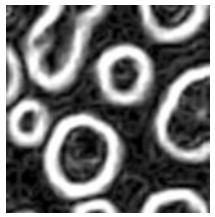
False-positive: 2

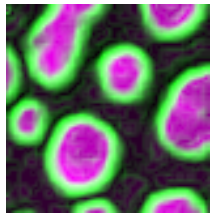
IoU = 1 / 2

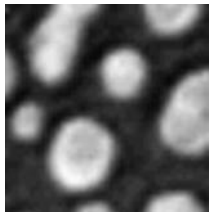


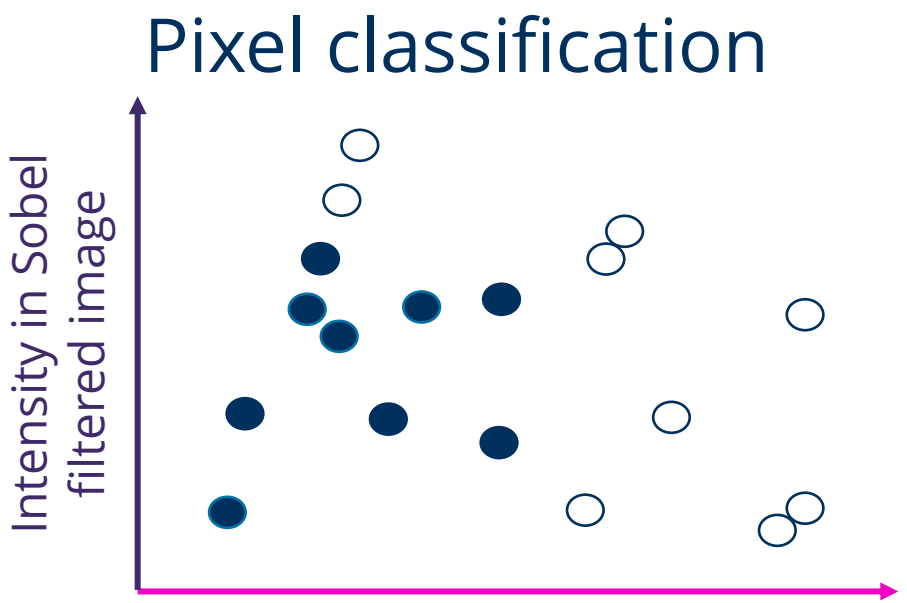
Object classification

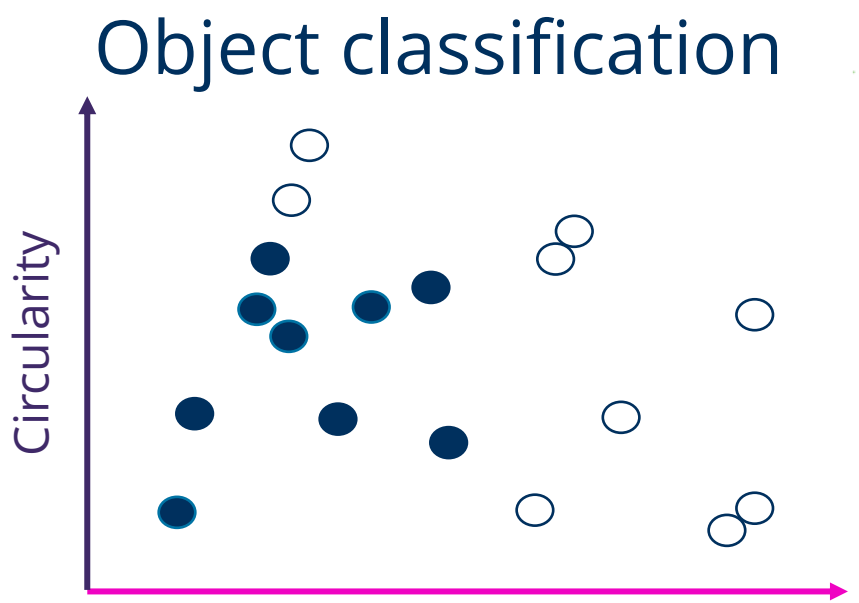
What if we exchange pixel features with object features?











- The algorithms work the same using with different features

ScaDS.AI
DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE

Supervised and Unsupervised Machine Learning for Bio-image Analysis

Using Python

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.

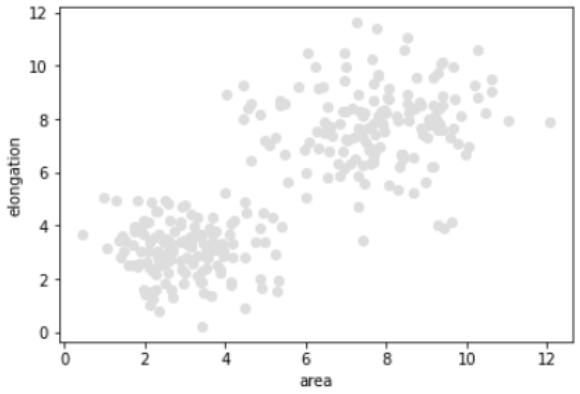
Reusing materials from Johannes Soltwedel, Till Korten, Johannes Müller, Laura Žigutytė (TU Dresden), Ryan Savill (MPI-CBG), Matthias Täschner (ScaDS.AI/Uni Leipzig) and the Scikit-learn community.

Tabular object classification

Classify objects starting from feature vectors (table columns)

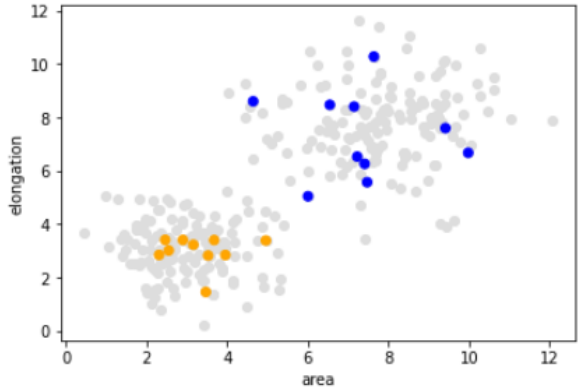
Raw data

	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,
```

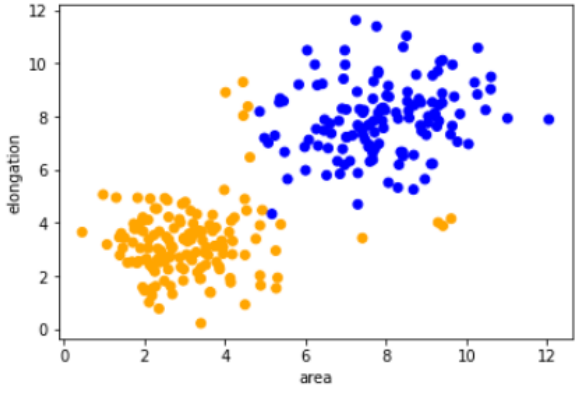


Classifier training

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

Classifier prediction

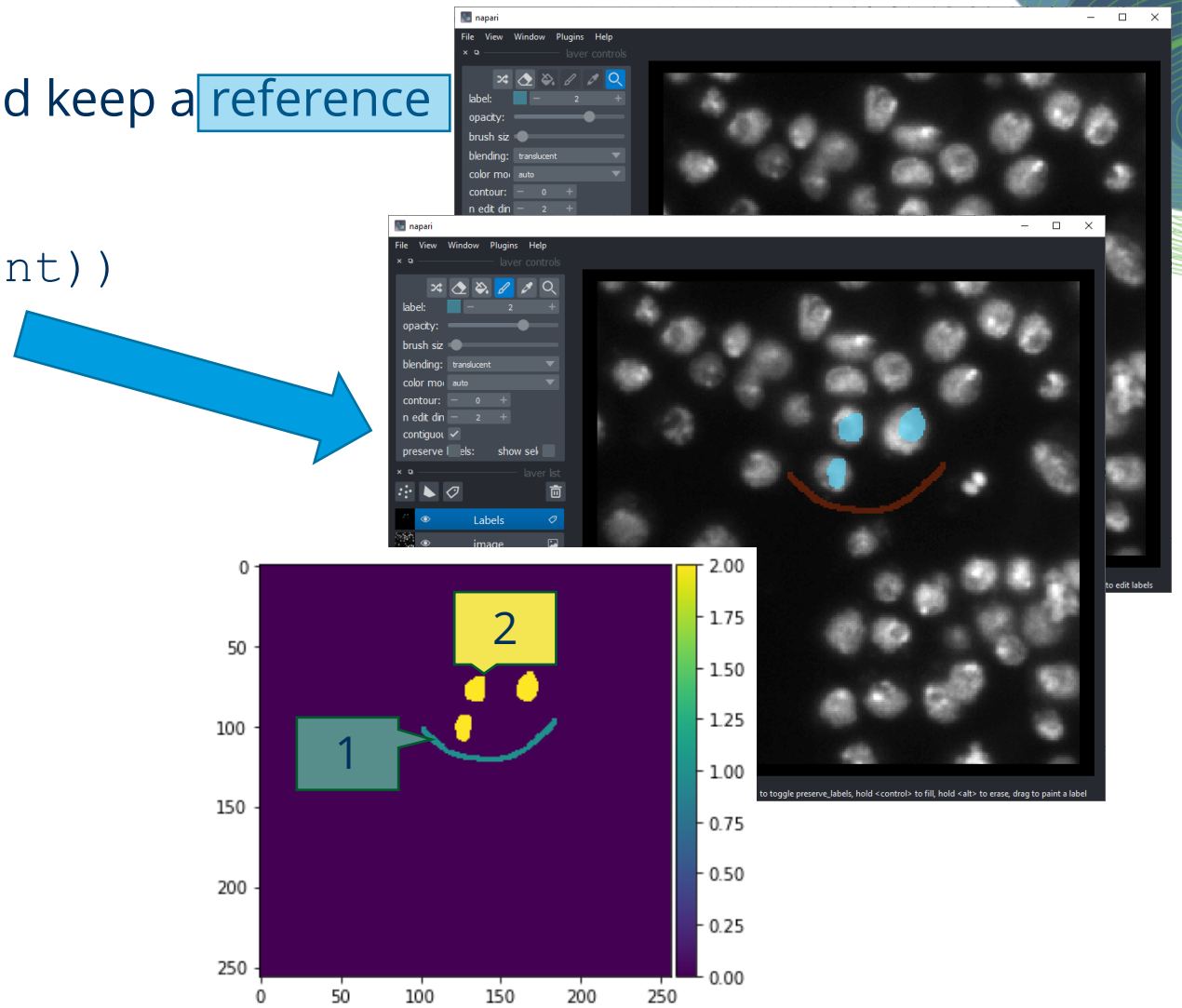
```
result = classifier.predict(validation_data)
```



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
```

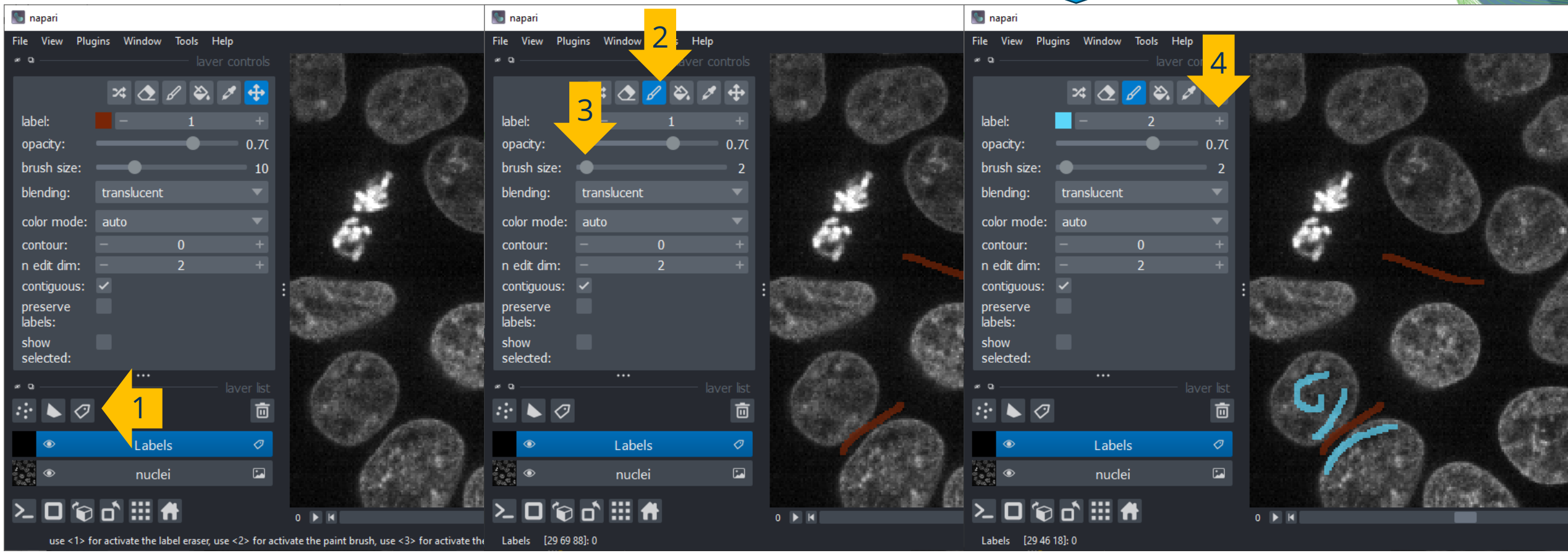
```
from skimage.io import imshow
imshow(manual_annotations,
               vmin=0, vmax=2)
```


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)
```

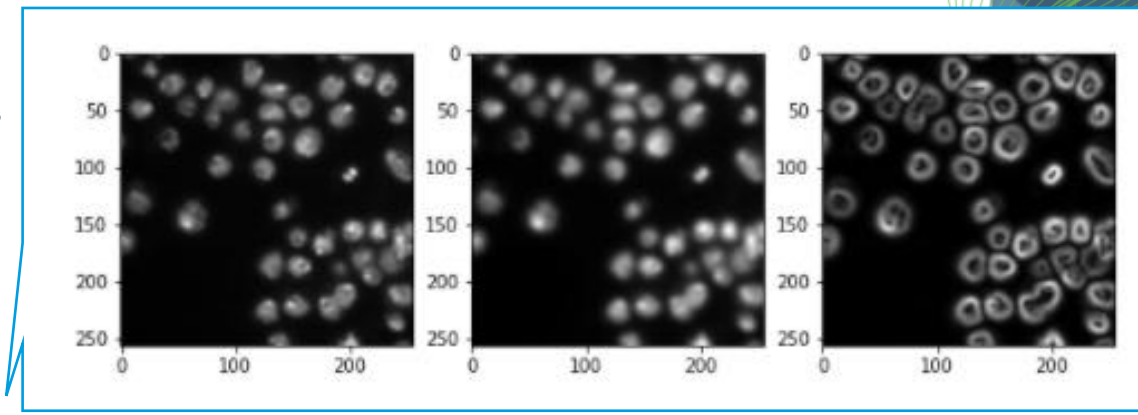
Image data

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

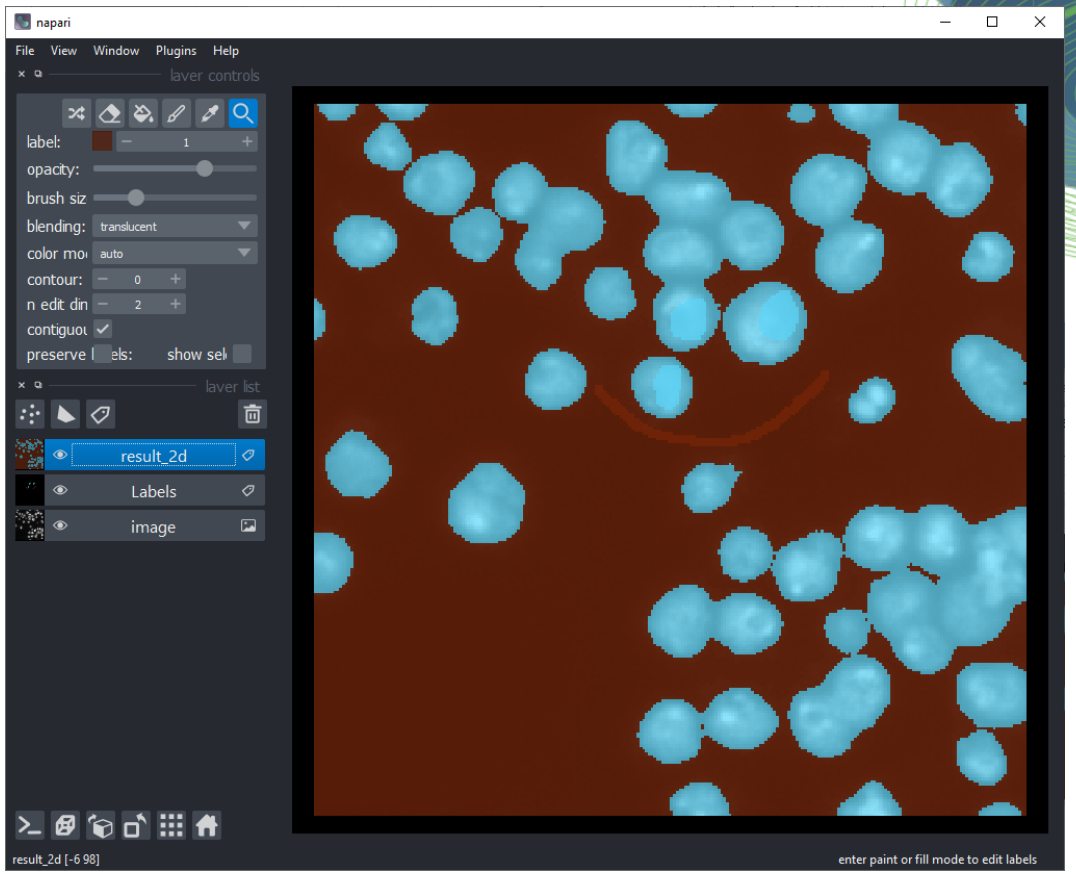


Image data source: BBBC038v1, available from the Broad Bioimage Benchmark Collection (Caicedo et al., Nature Methods, 2019)

Supervised and unsupervised ML in scikit-learn

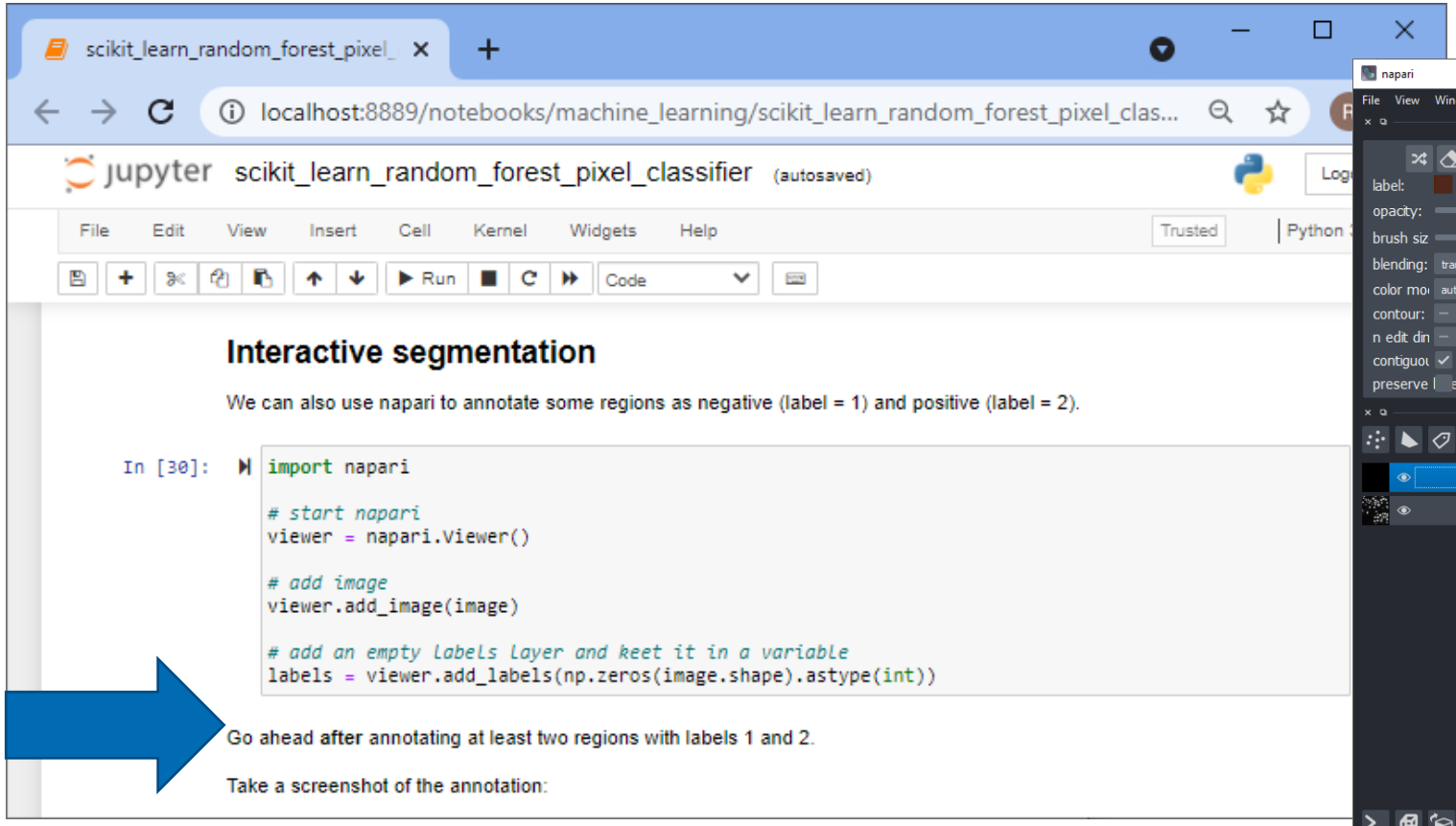
Common patterns

```
from sklearn.xyz import Model

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

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

y_pred = model.predict(x)
```

$$y = f(x)$$

Interactive pixel classification

Jupyter notebooks and napari side-by-side



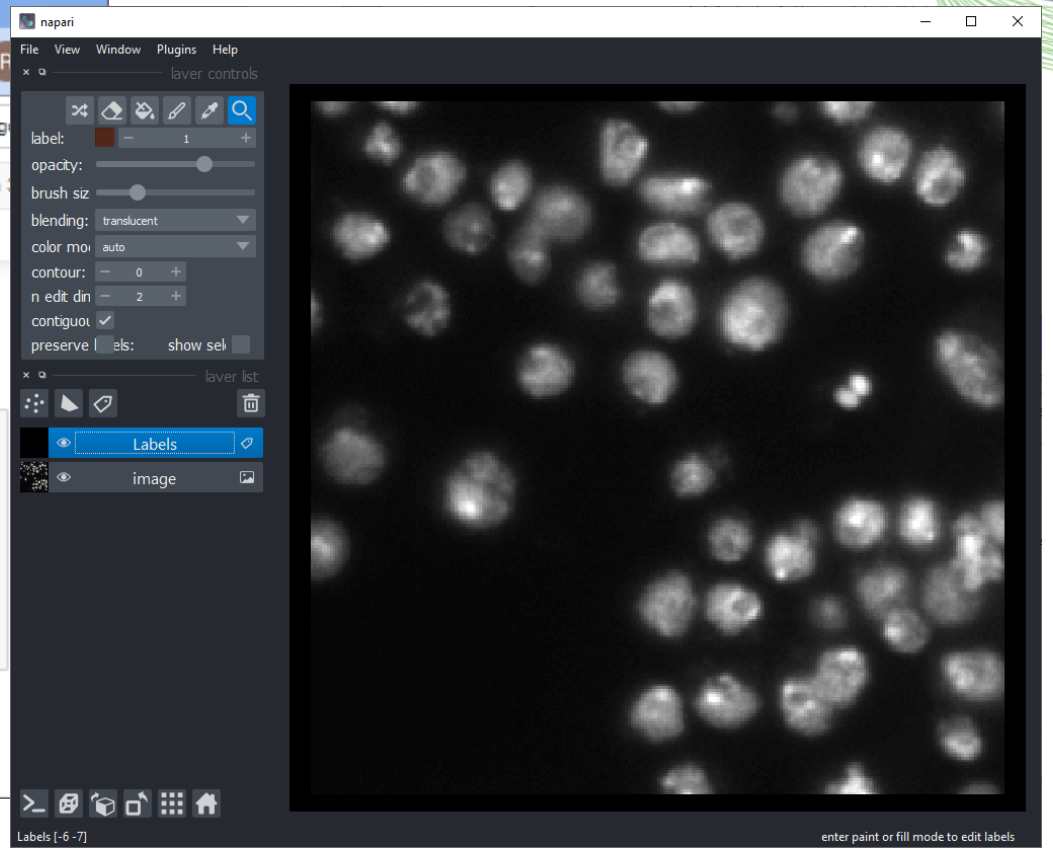


Image data source: BBBC038v1, available from the Broad Bioimage Benchmark Collection (Caicedo et al., Nature Methods, 2019)

Interactive pixel classification

Jupyter notebooks and napari side-by-side

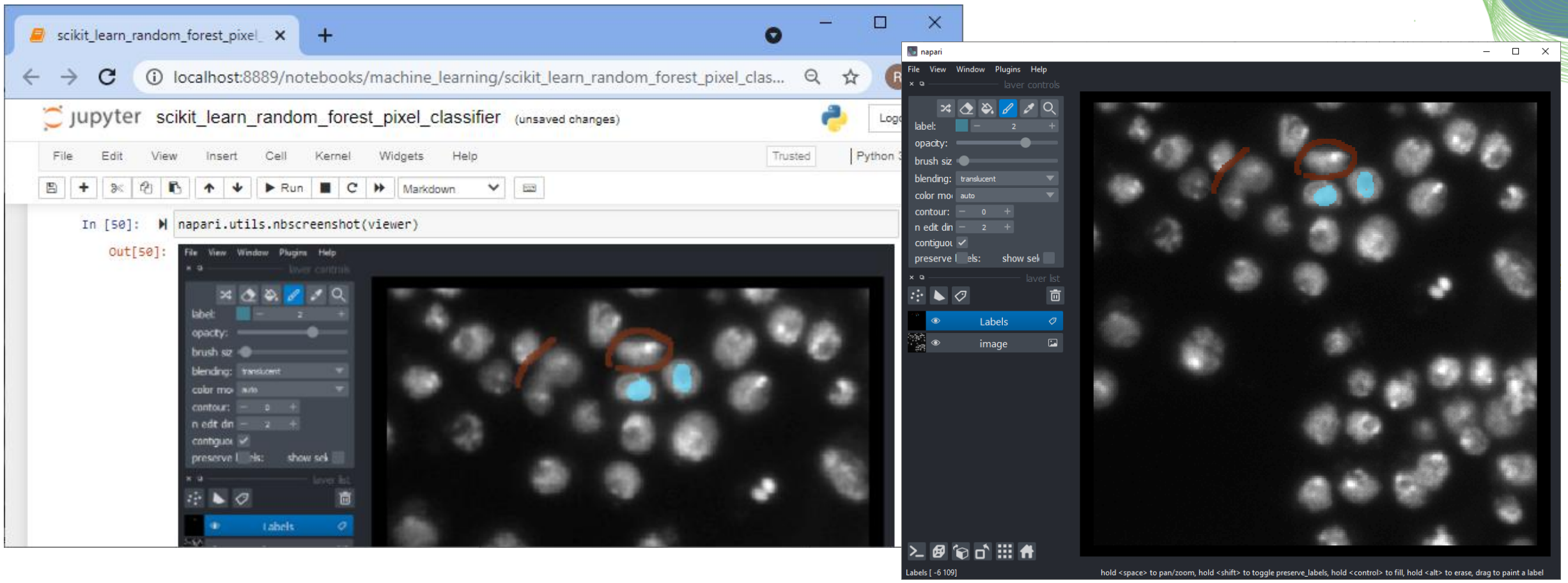
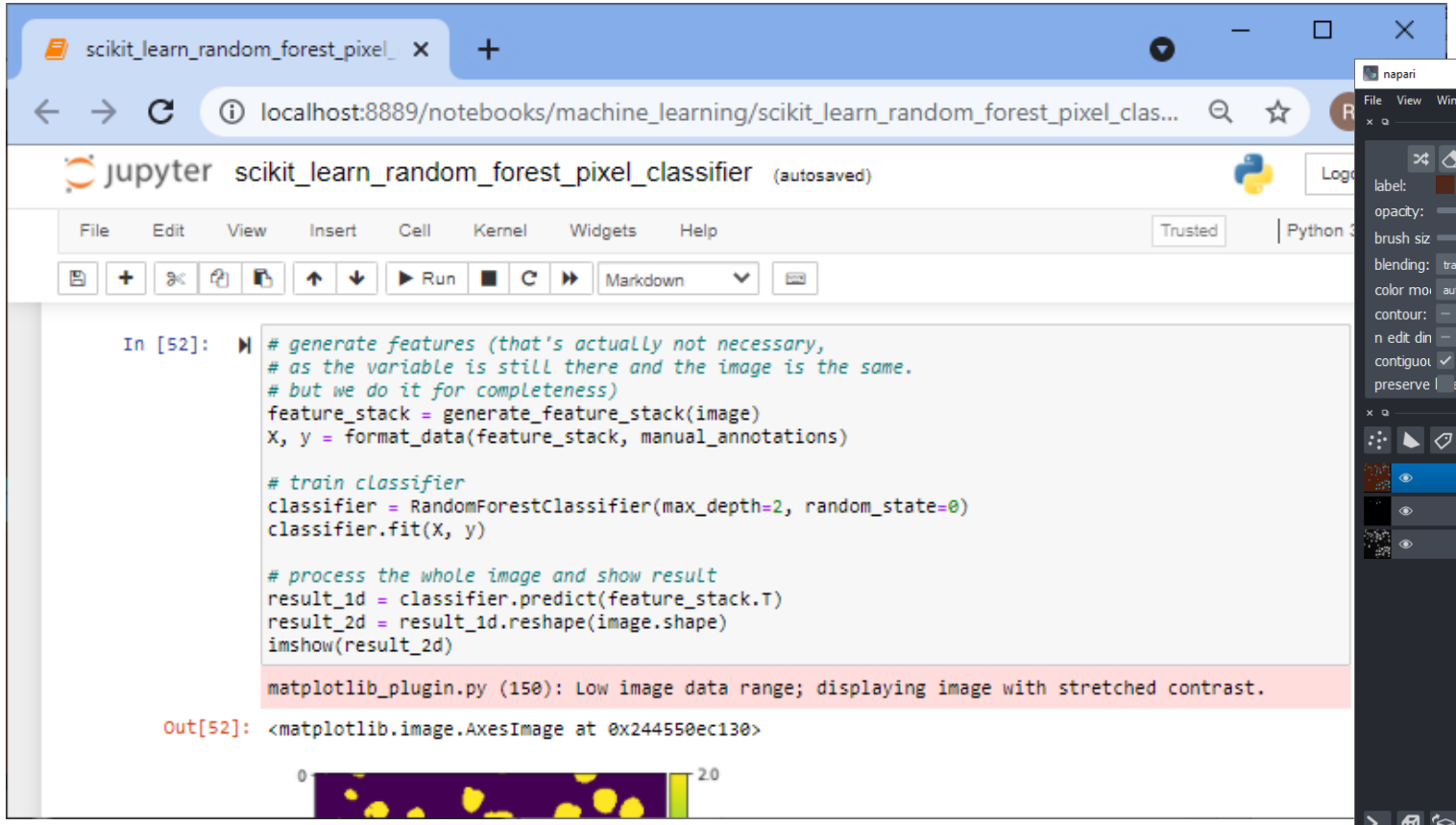


Image data source: BBBC038v1, available from the Broad Bioimage Benchmark Collection (Caicedo et al., Nature Methods, 2019)

Interactive pixel classification

Jupyter notebooks and napari side-by-side



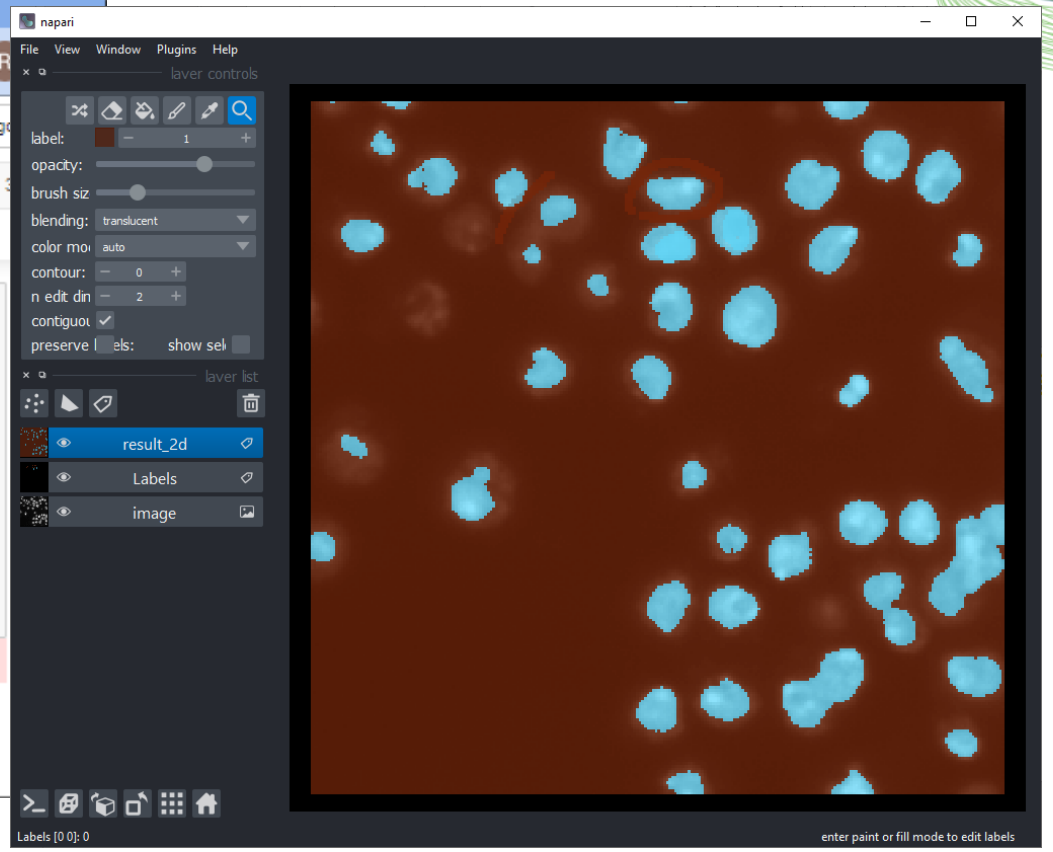


Image data source: BBBC038v1, available from the Broad Bioimage Benchmark Collection (Caicedo et al., Nature Methods, 2019)

ScaDS.AI
DRESDEN LEIPZIG

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

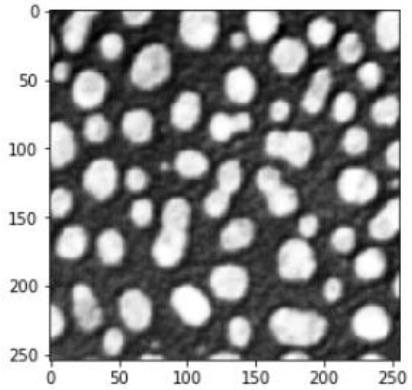
SACHSEN

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 Žigutytė (TU Dresden), Ryan Savill (MPI-CBG), Matthias Täschner (ScaDS.AI/Uni Leipzig) and the Scikit-learn community.

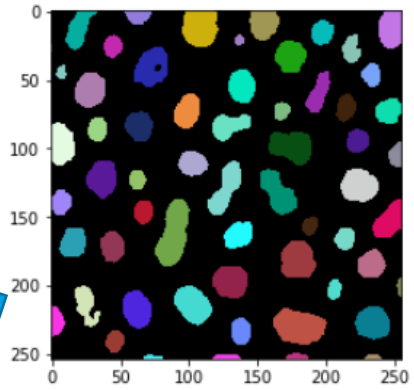
Accelerated pixel and object classification

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



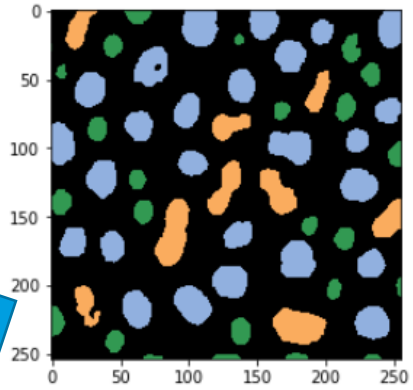
Raw image

Object segmentation

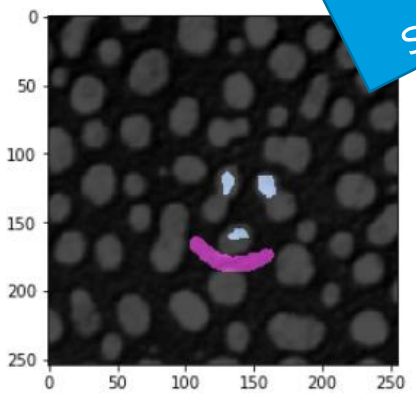


Object label image

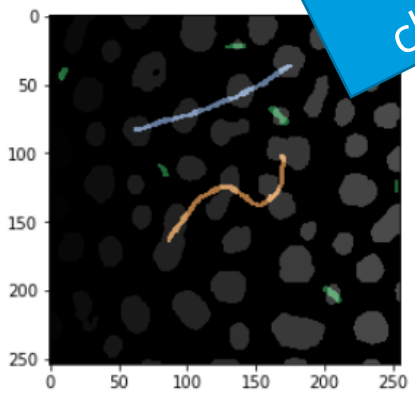
Object classification



Class label image



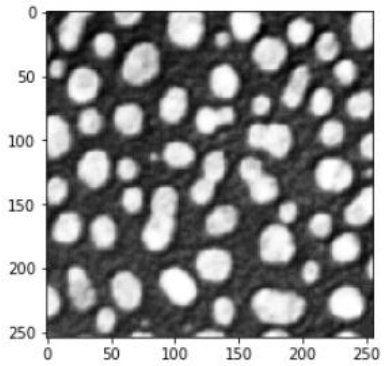
Pixel annotation



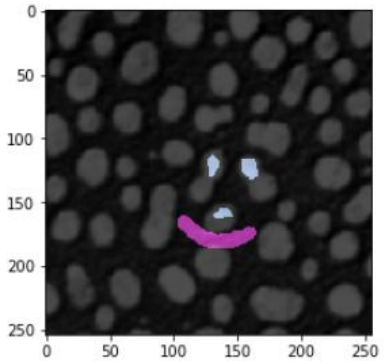
Object annotation

Object segmentation

Pixel classification + connected component labeling



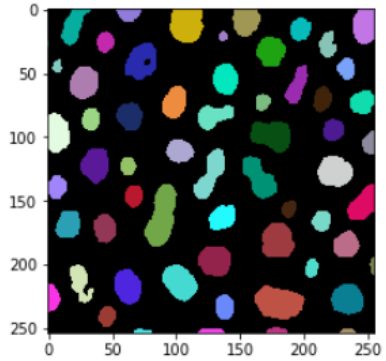
Raw image



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 segmentation

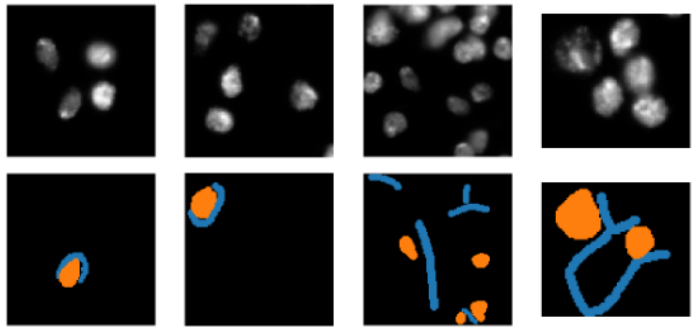


Object label image

Training on folders of annotated images

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

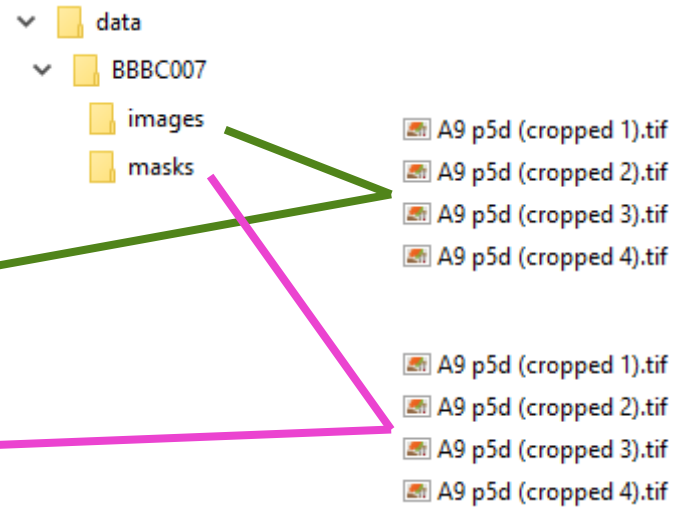
```
[3]: file_list = os.listdir(image_folder)

     # 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)
```

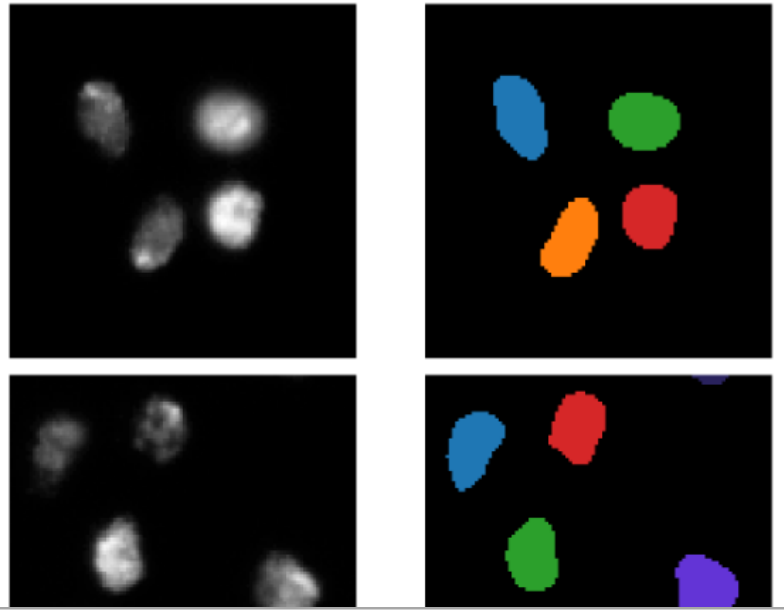


Prediction

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

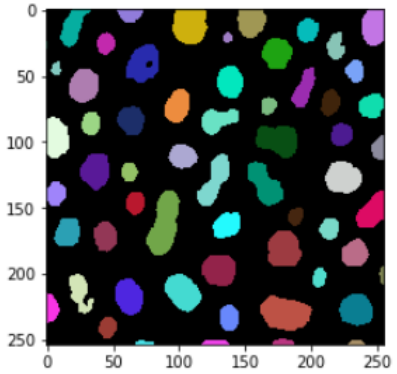
```
[5]: segmenter = apoc.ObjectSegmenter(opencl_filename="data/object_segmenter_trained_on_f
```

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

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

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

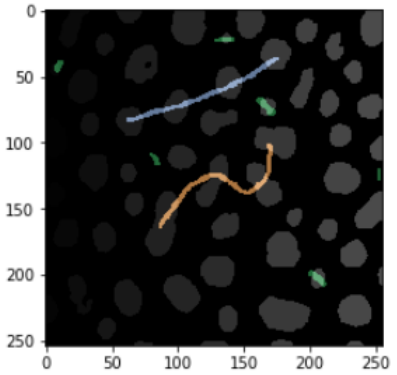


Object classification

Feature extraction + tabular classification



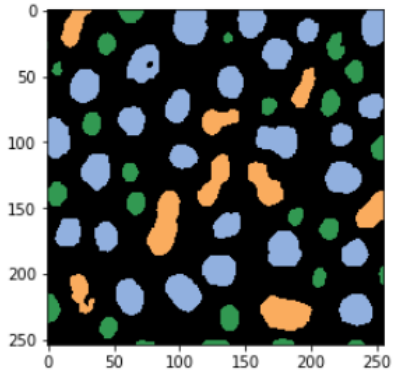
Object label image



Object annotation

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

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

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

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

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

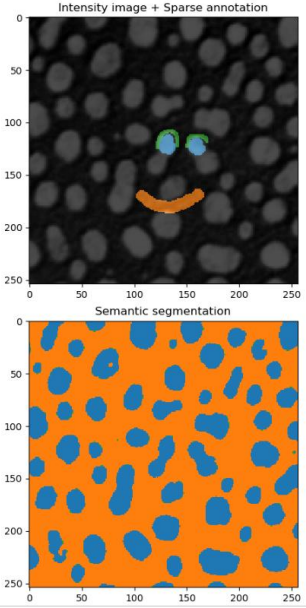
Object classification



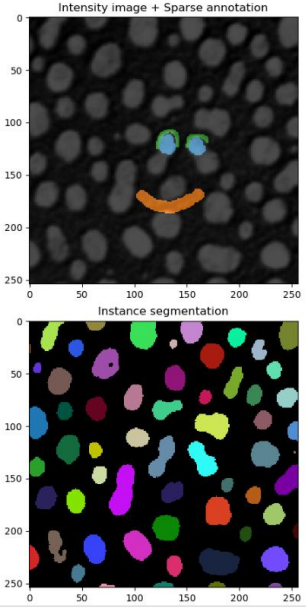
Class label image

Other classification / regression tasks

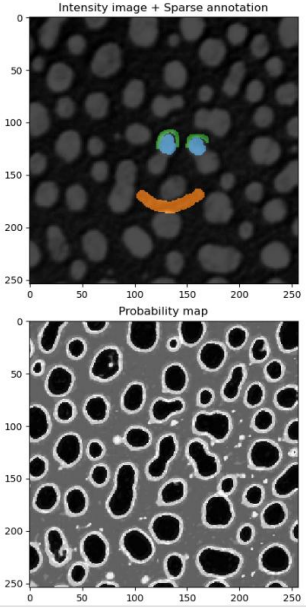
Pixel-Classifier



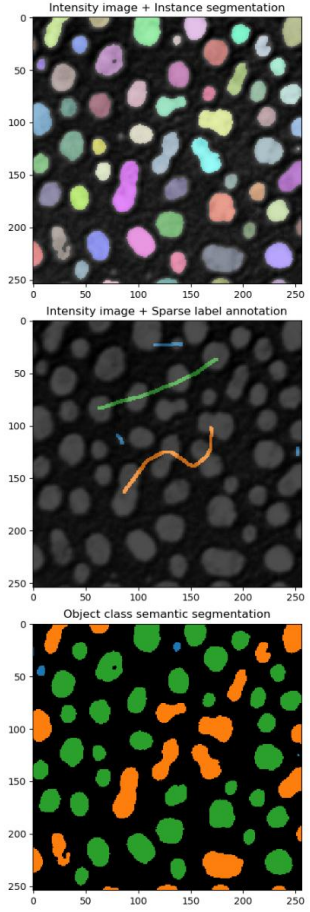
Object-Segmenter



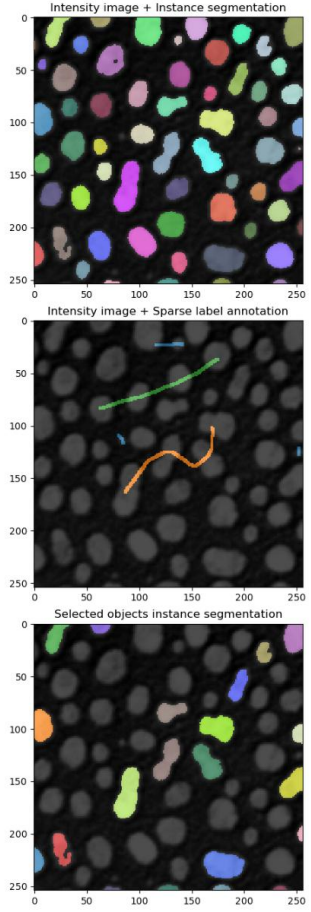
Probability-Mapper



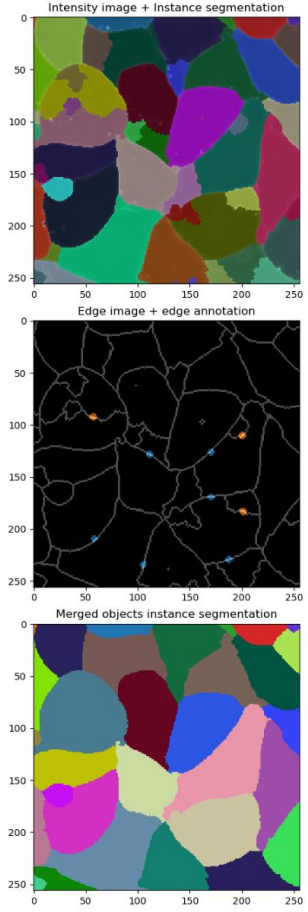
Object-Classifier



Object-Selector

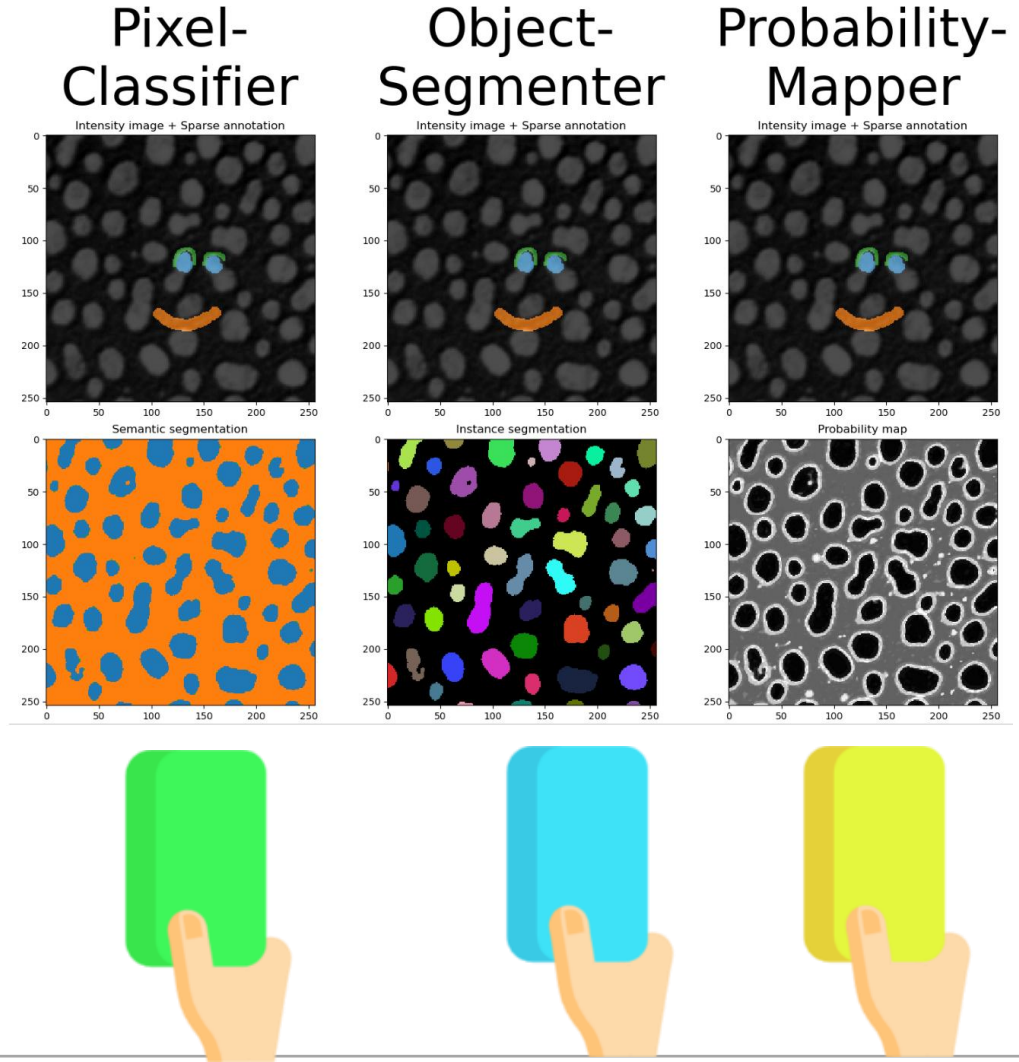


Object-Merger



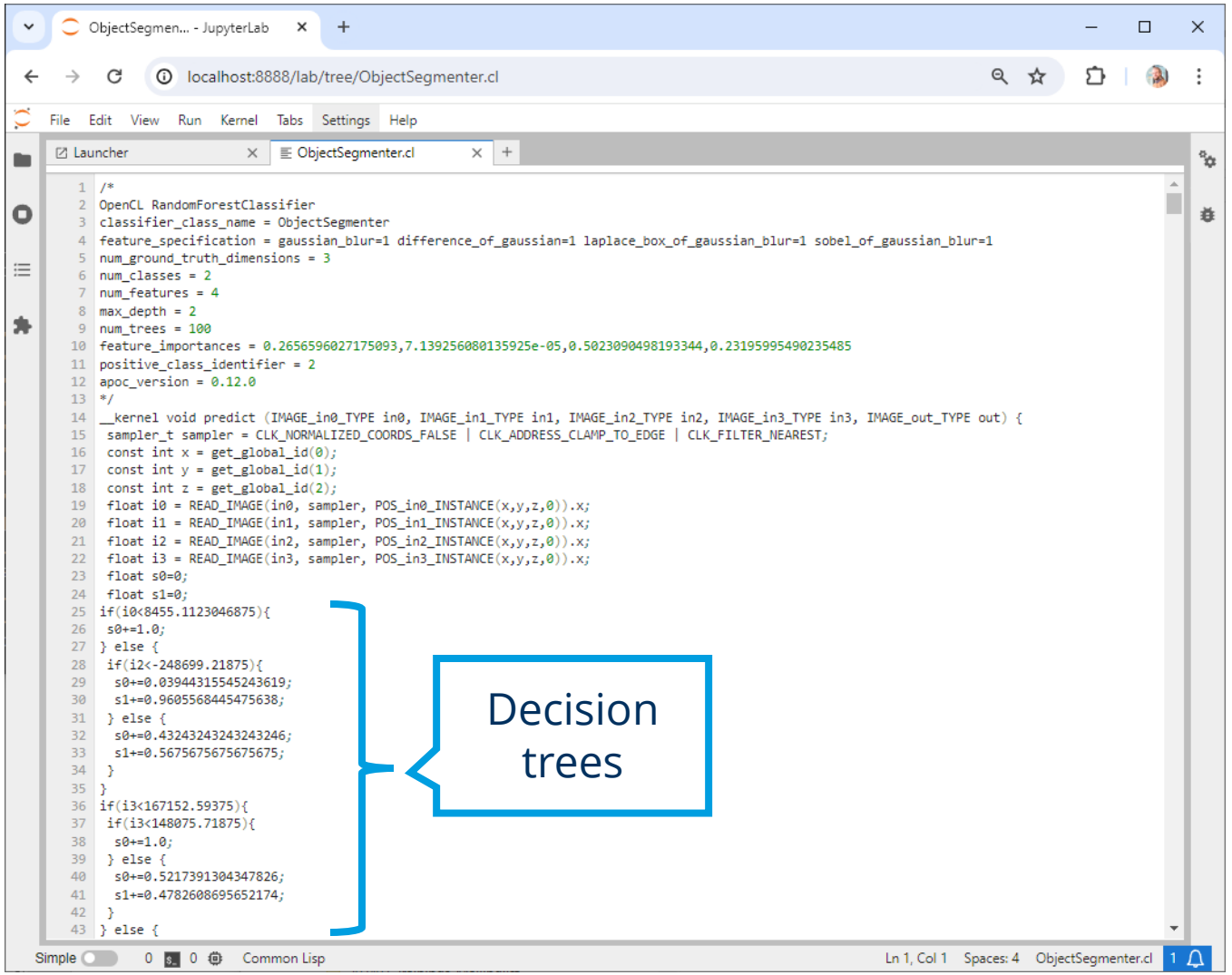
Quiz: Classification versus Regression

Which of these three solves a regression task?



Under the hood: clesperanto / OpenCL

classifier.cl files
can be *read*



ScaDS.AI
DRESDEN LEIPZIG
CENTER FOR SCALABLE DATA ANALYTICS
AND ARTIFICIAL INTELLIGENCE

Explainable Machine Learning

Robert Haase

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.

Explainable Artificial Intelligence (XAI)

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

  Wikipedia [1]

Relevant Aspects:

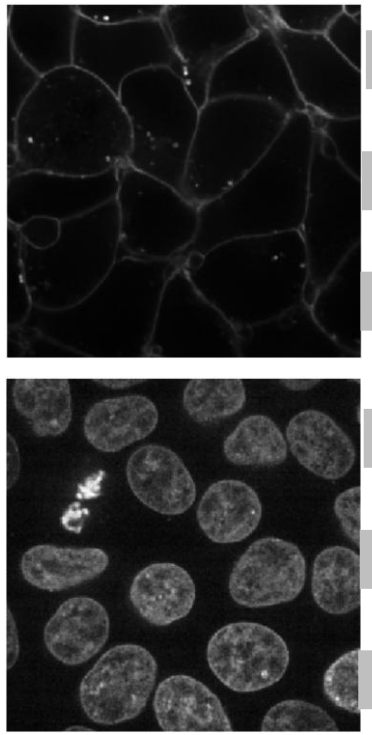
- Explainability vs. Interpretability of AI-algorithms
- We seek to enable humans to
  - predict results of AI Systems,
  - trust AI-Systems and
  - using AI-Systems effectively.

Explanation of Random Forest Classifiers

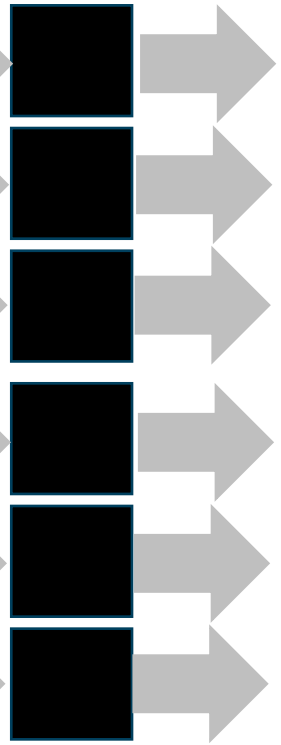
… by reading codde

… is quite useless

Input images



Feature images



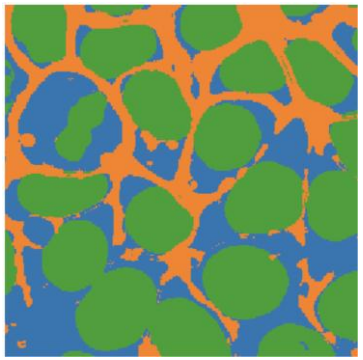
Decision trees (OpenCL)

```
if(i3<6093.5){
 if(i1<3335.03
  s0+=0.994917
  s1+=0.001270
  s2+=0.003811
 } else {
  s1+=1.0;
 }
} else {
 if(i4<8042.86
  s0+=0.818181
  s1+=0.181818
 } else {
  s0+=0.000885
  s1+=0.001180
  s2+=0.997933
 }
}
```

```
if(i3<6615.0){
 if(i1<3277.4873
  s0+=0.98595146
  s1+=0.00127713
  s2+=0.01277139
 } else {
  s1+=1.0;
 }
} else {
 if(i1<3244.2104
  s0+=0.00118835
  s2+=0.99881164
 } else {
  s1+=1.0;
 }
}
```

```
if(i0<2118.0){
 if(i4<8134.84912109375){
  s0+=1.0;
 } else {
  s0+=0.0012077294685990338;
  s2+=0.998792270531401;
 }
} else {
 if(i1<3335.035888671875){
  s0+=0.8279569892473119;
  s1+=0.005376344086021506;
  s2+=0.16666666666666666;
 } else {
  s1+=1.0;
 }
}
```

Classification result images



Explainability

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

Intrinsically explainable AI-algorithms

- Example: Linear Regression

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

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

Model explainable

Results predictable

Explainability

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

Intrinsically explainable AI-algorithms

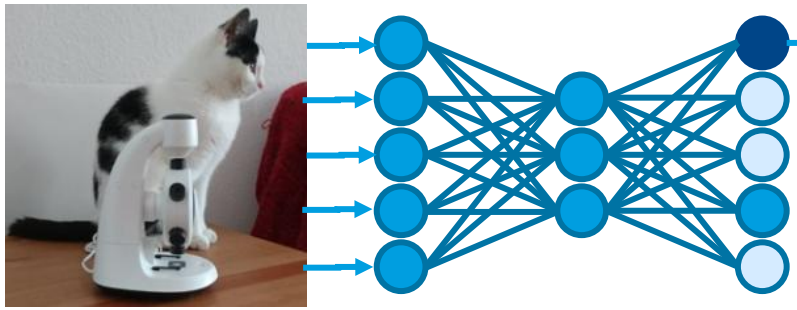
- Example: Linear Regression

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

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

Black-Box AI-algorithms

- Example: Deep Neural Networks (DNN)



"sewing machine" [1]

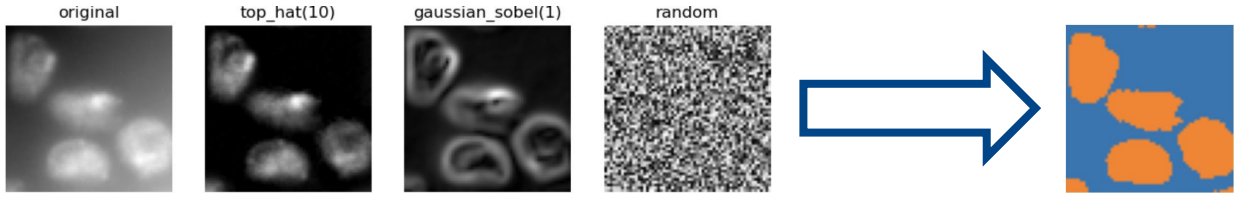
Not easily explainable and predictable

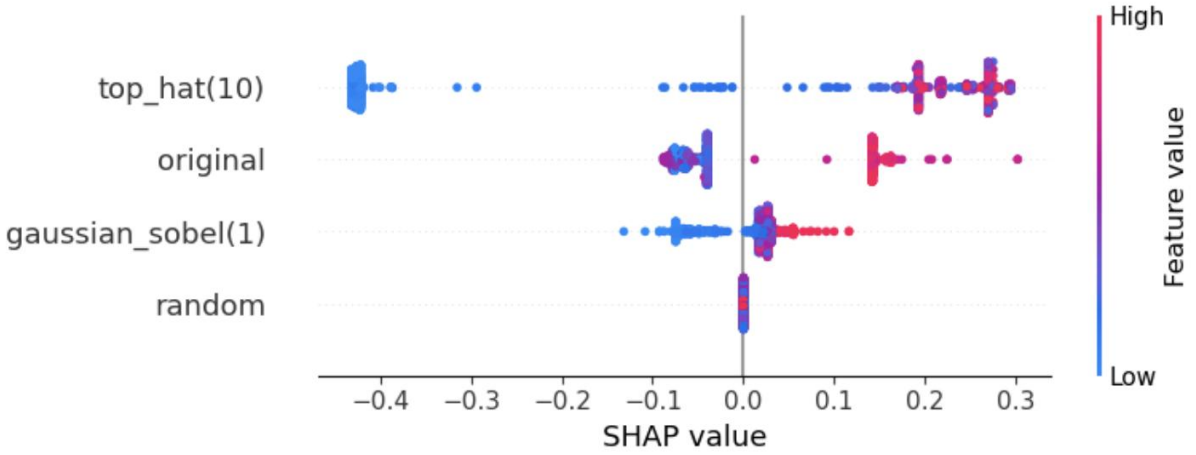
Interpretability

Visualization of intermediate results and their influence on results

Model-agnostic methods

Example: Shapley's Additive exPlanations (SHAP)



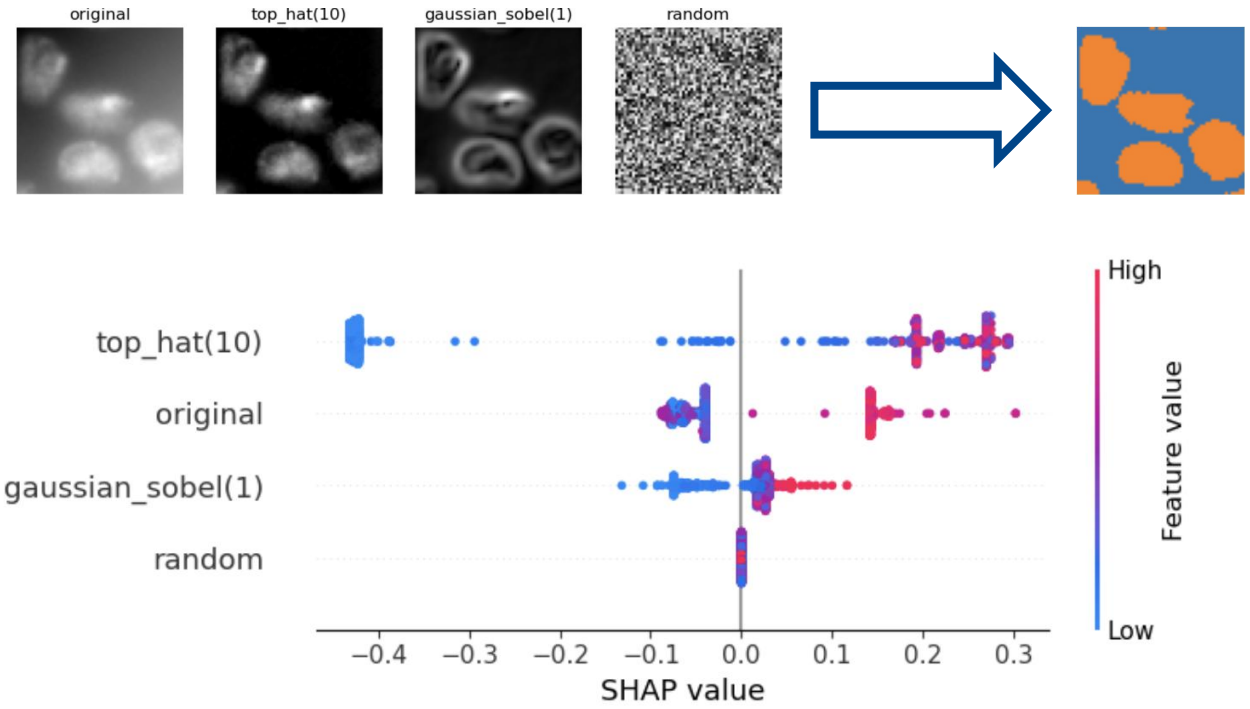


Interpretability

Visualization of intermediate results and their influence on results

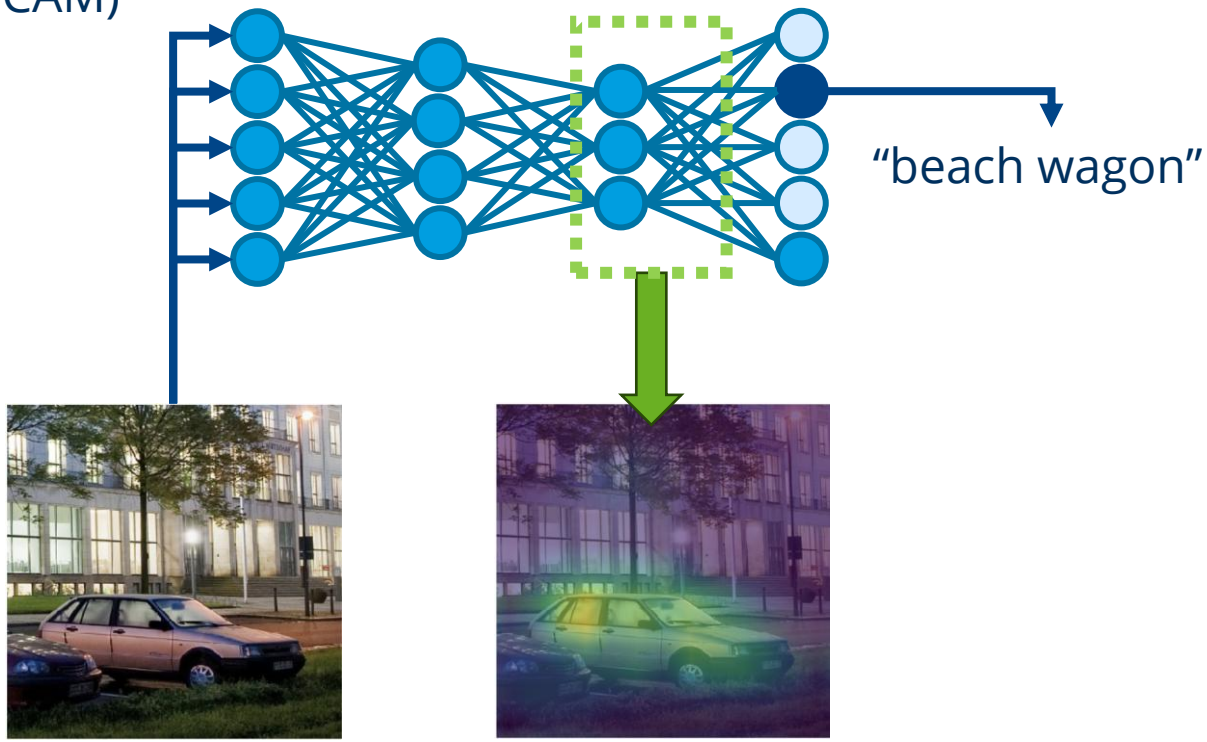
Model-agnostic methods

Example: Shapley's Additive exPlanations (SHAP)



Model-specific methods

Example: Gradient Class Activation Maps (Grad-CAM)

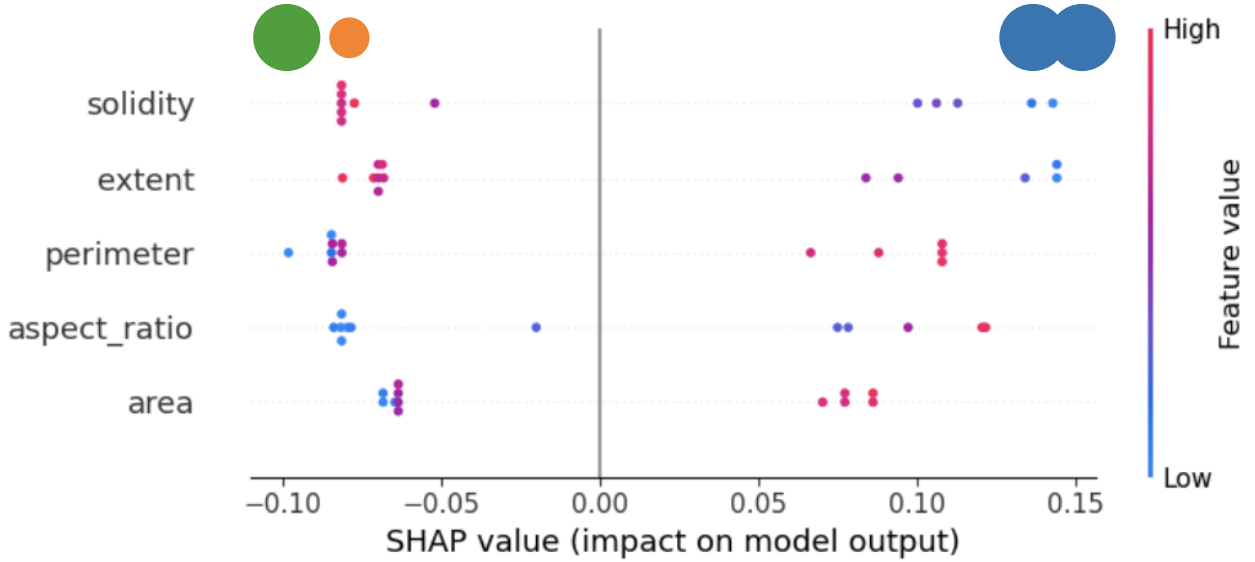


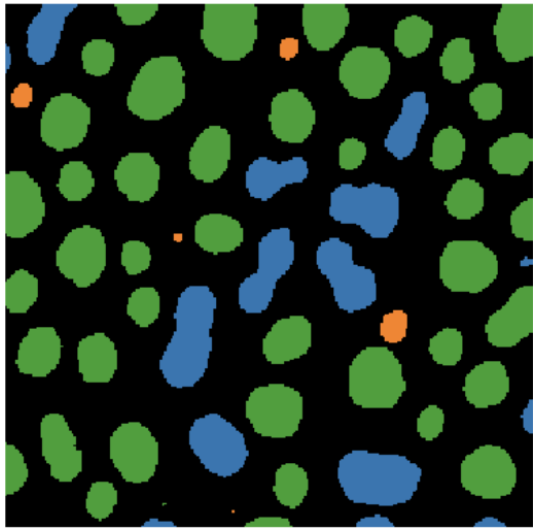
https://haesleinhuepf.github.io/xai/30_shap/pixel_classifier.html
https://haesleinhuepf.github.io/xai/60_grad-cam/classification_resnet.html

Explainable AI

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

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





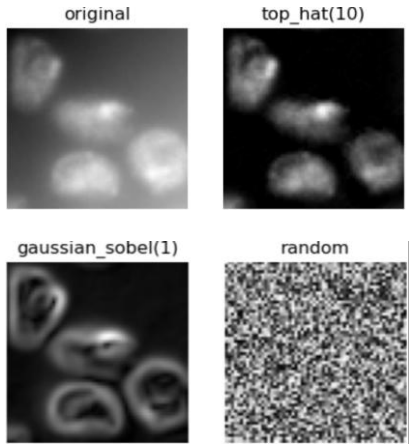
Recap: Feature selection

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

- Amplitude
- Energy
- Duration
- ...

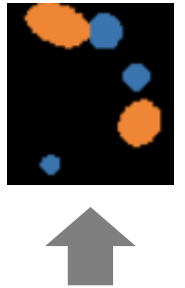
- Noise
- Tourists jumping on a sensor
- Earthquake approaching

Signal classification



Pixel classification

- Area
- Perimeter
- Aspect ratio
- ...



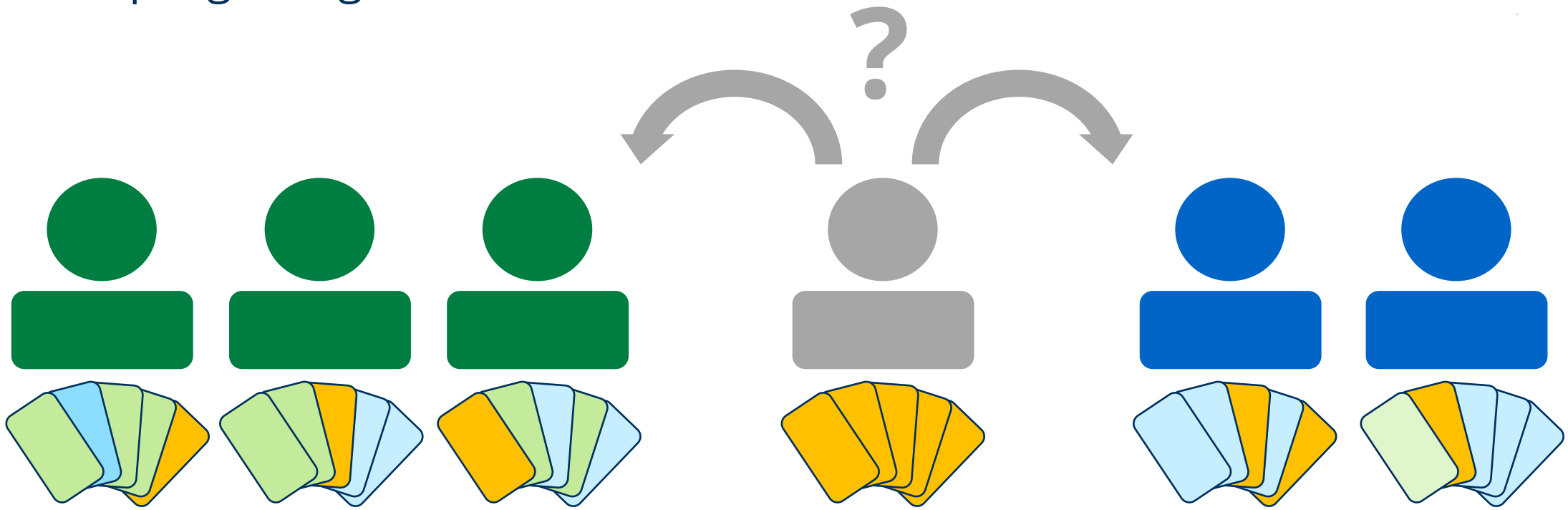
- Round
- Elongated

Object classification

Collaborative game theory

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

Example game goal: maximize cards of the same colour.

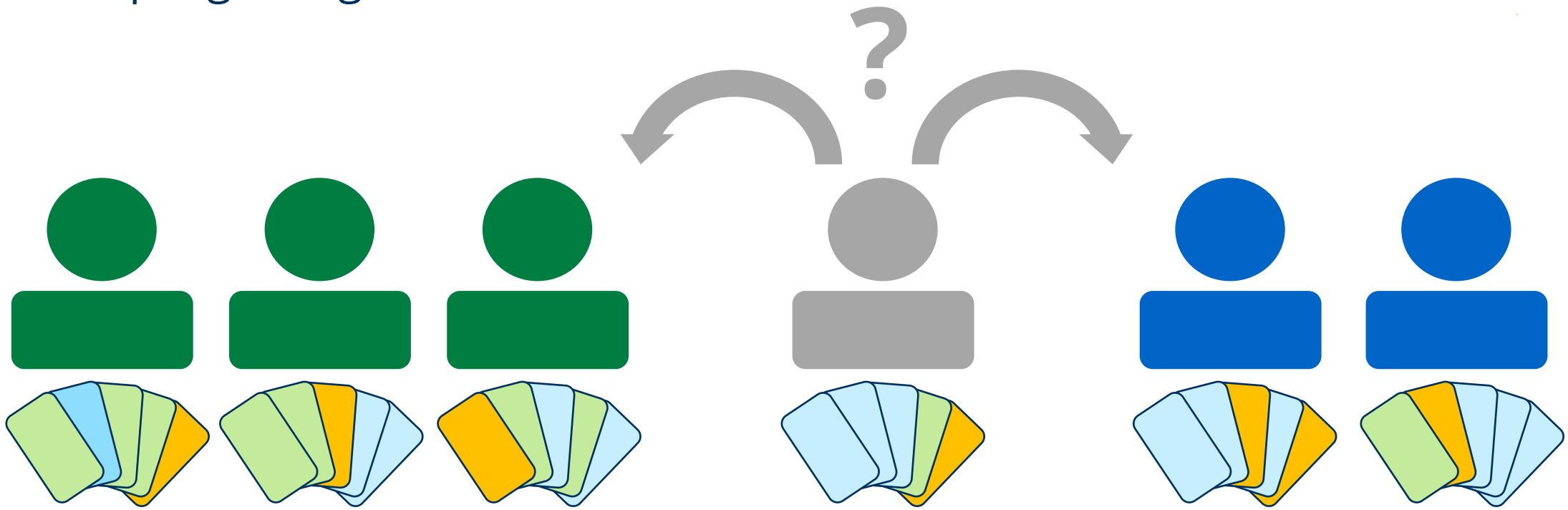


Value for both teams: 0

Collaborative game theory

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

Example game goal: maximize cards of the same colour.



Value for team green: 0.1
Value for team blue: 0.3

SHAP

SHapley's Additive exPlanations

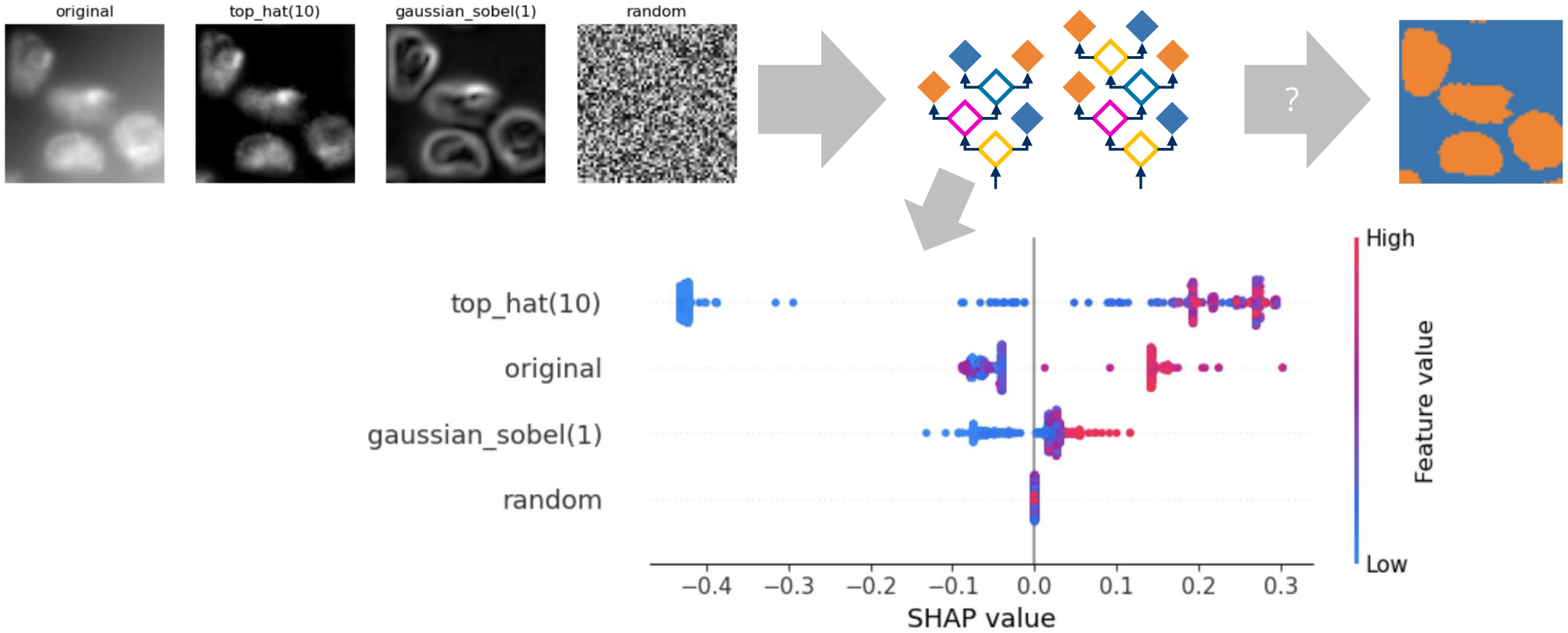
Analogously, this can be done with data points instead of features.

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

	SHAP value of feature i	Sum over all Subsets of Features not including i	Weight related to number of used features in relation all players	Quality of classifier using feature i	Quality of classifier *not* using feature i
Game theory	SHAP value of player i	Sum over all Subsets of Players not including i	Weight related to number of players in a coalition in relation to undecided players and all players	Chance to win game of coalition without player i	Chance to win game of coalition *including* player i

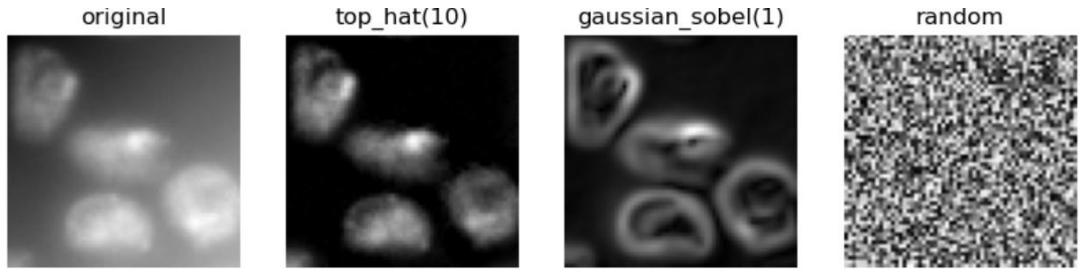
SHAP

Allows interpreting [pixel] classification results

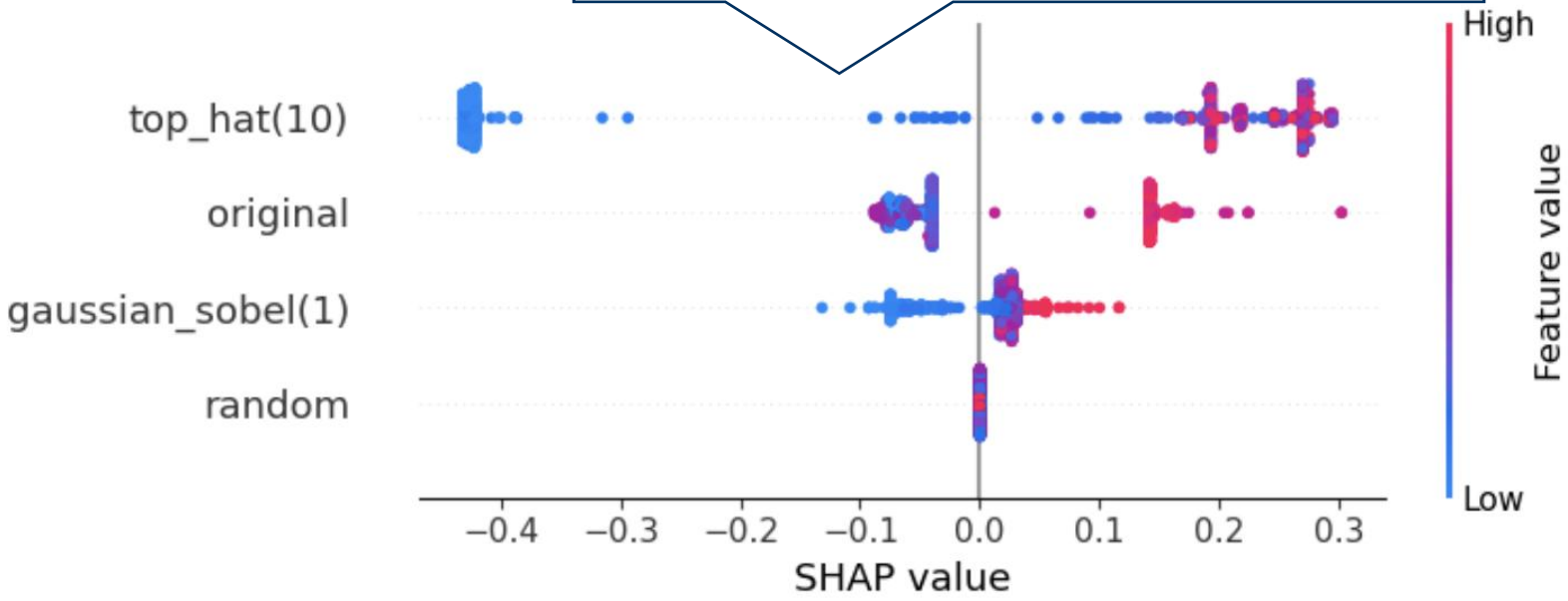


SHAP

Allows interpreting [pixel] classificatio

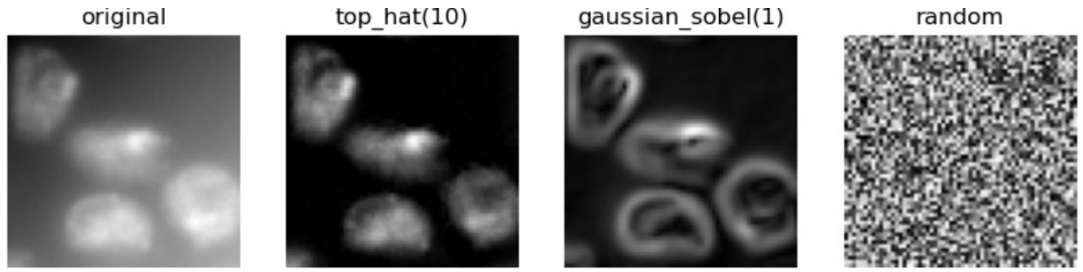


"If intensity in the top-hat image is high, the classifier tends to select the positive class (orange)."

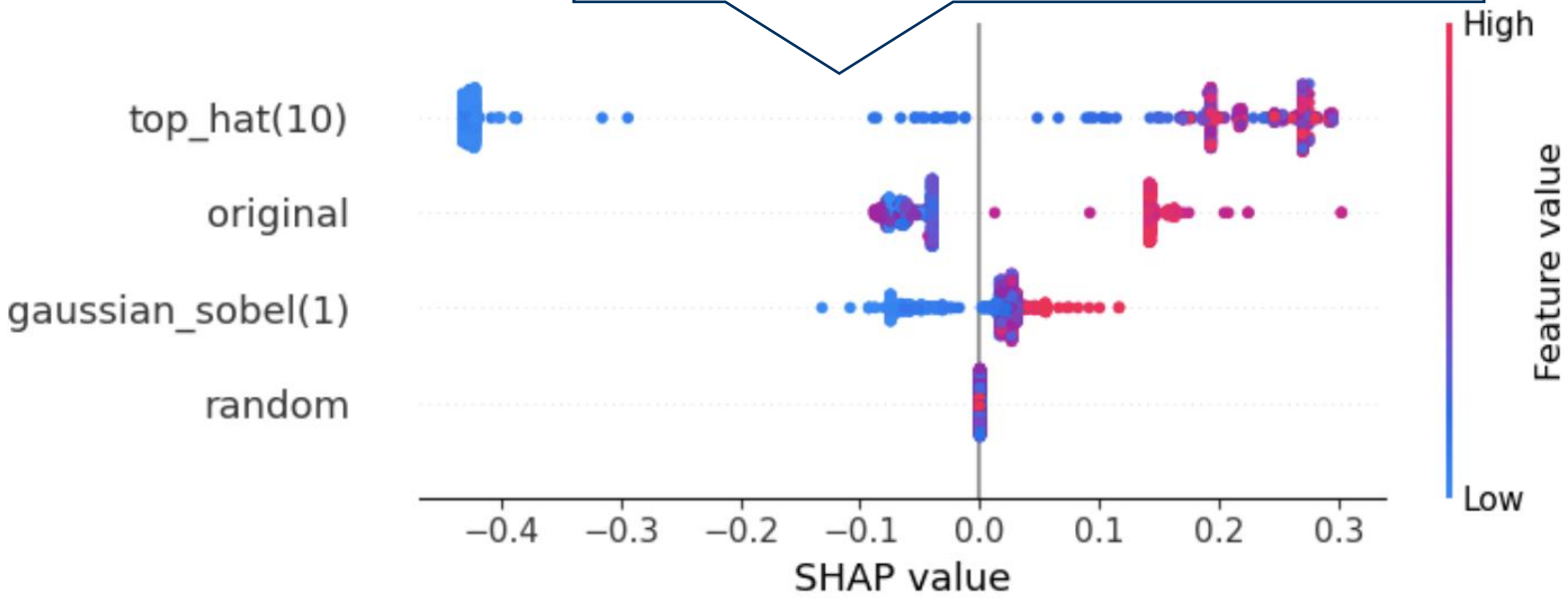


SHAP

Allows interpreting [pixel] classificatio

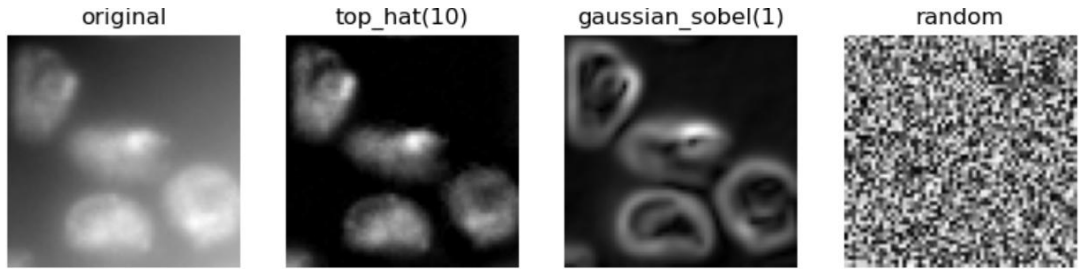


"If intensity in the top-hat image is low, the classifier needs to take other features into account."

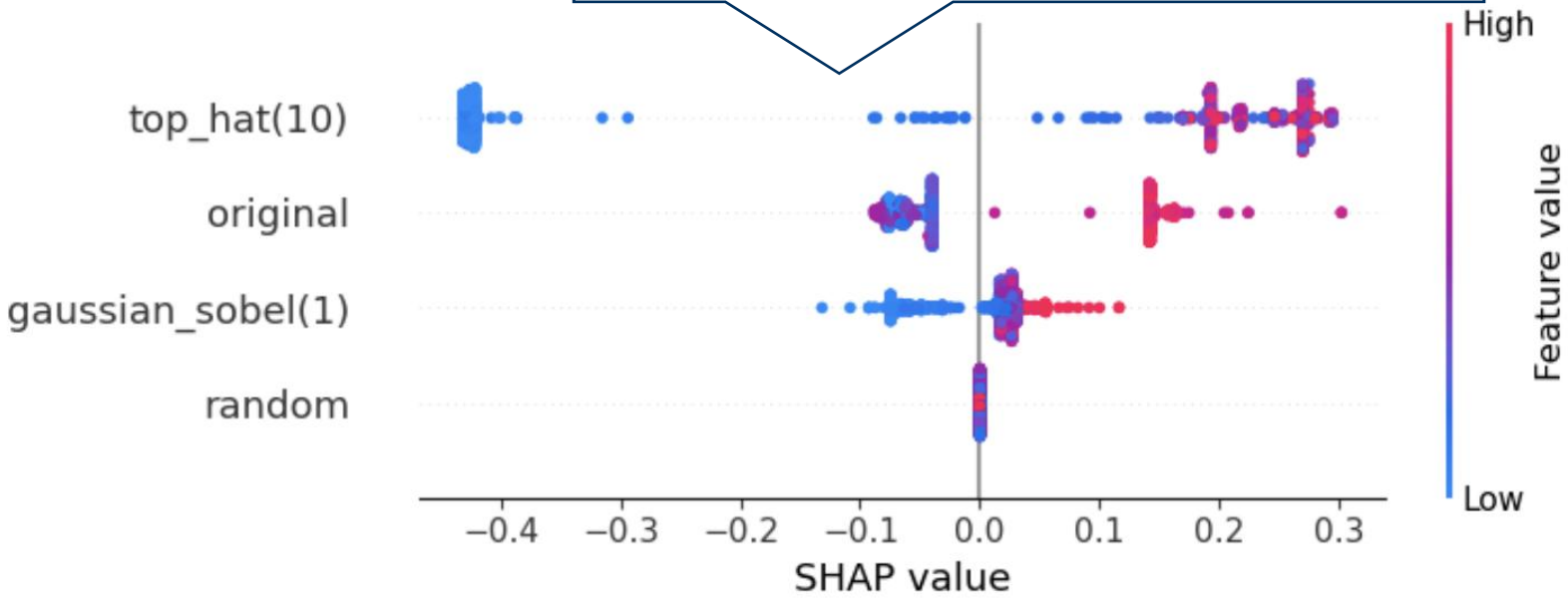


SHAP

Allows interpreting [pixel] classificatio

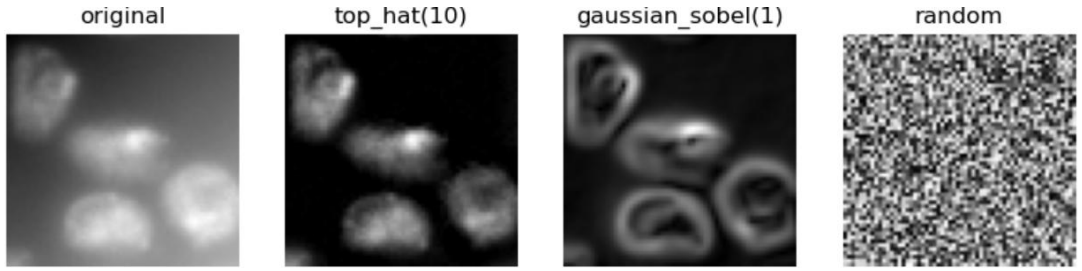


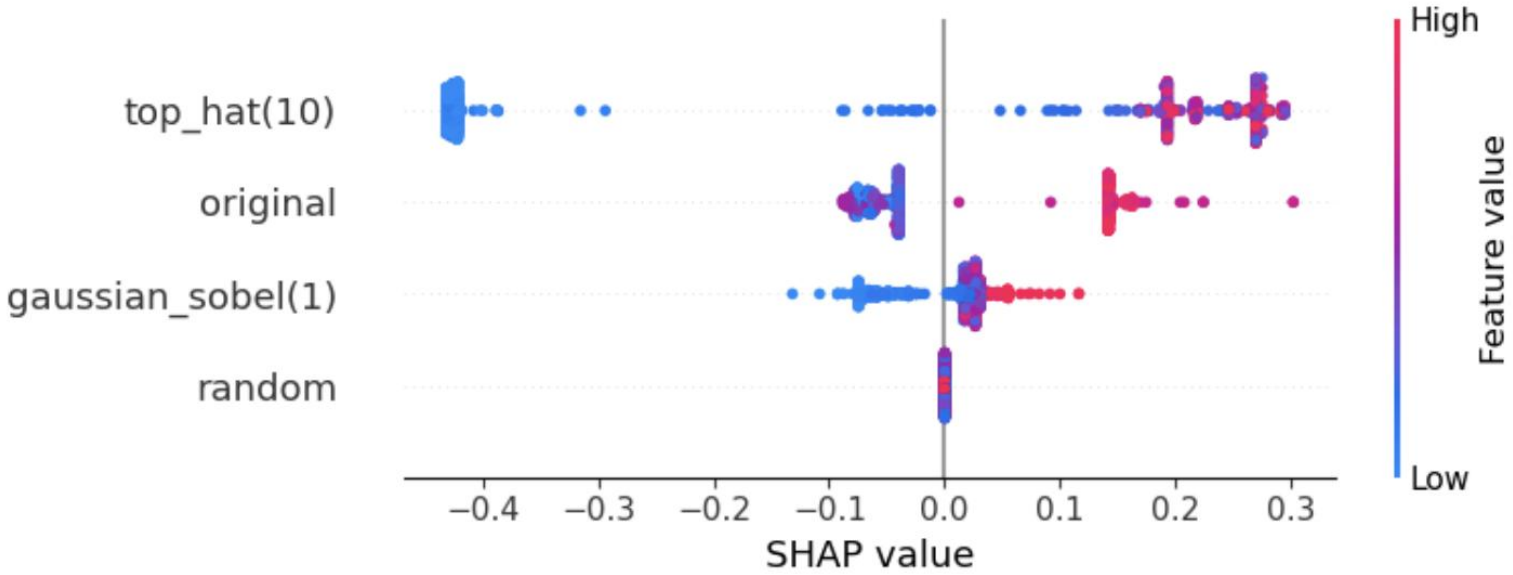
"The random feature has no value for classification."



Pitfall: Correlation

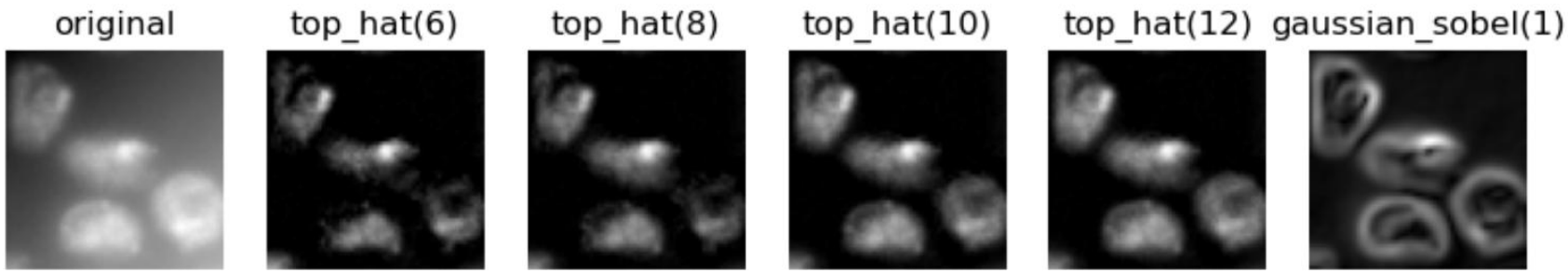
Correlated features may harm interpretability

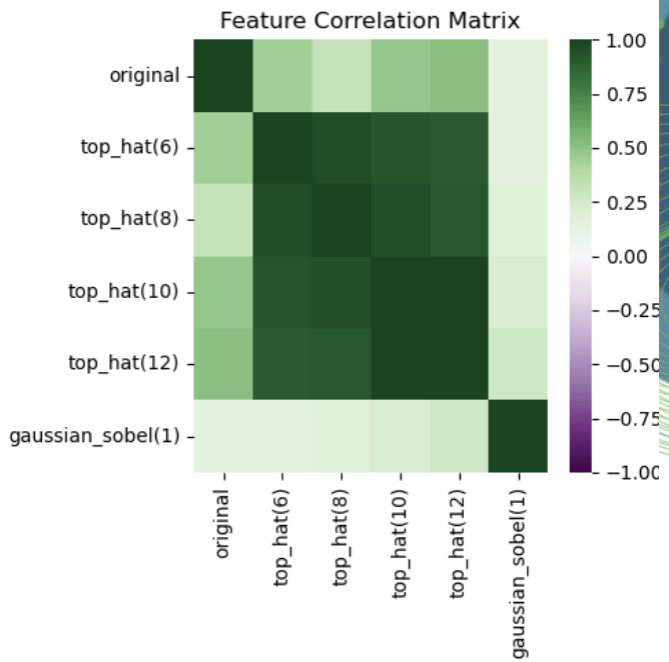




Pitfall: Correlation

Correlated features may harm interpretability

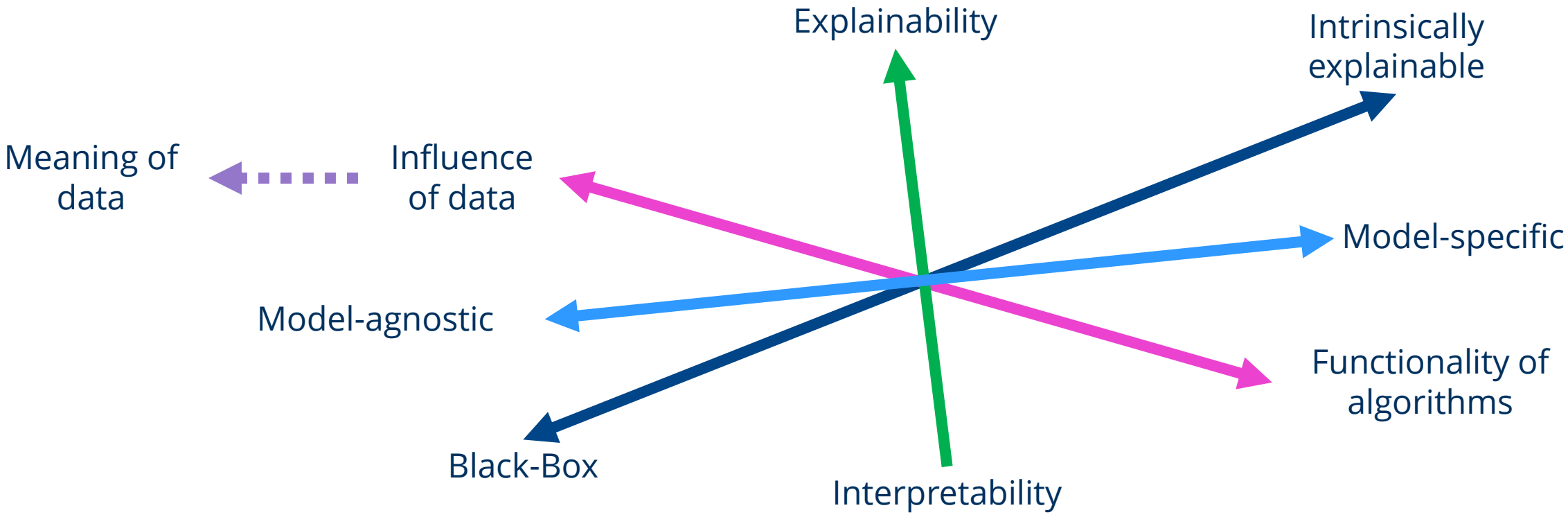




Features may appear less valuable.

Summary: Explainable AI

Methods of XAI can be classified on different scales



ScaDS.AI
DRESDEN LEIPZIG

CENTER FOR SCALABLE DATA ANALYTICS AND ARTIFICIAL INTELLIGENCE

Exercises

Robert Haase

Funded by

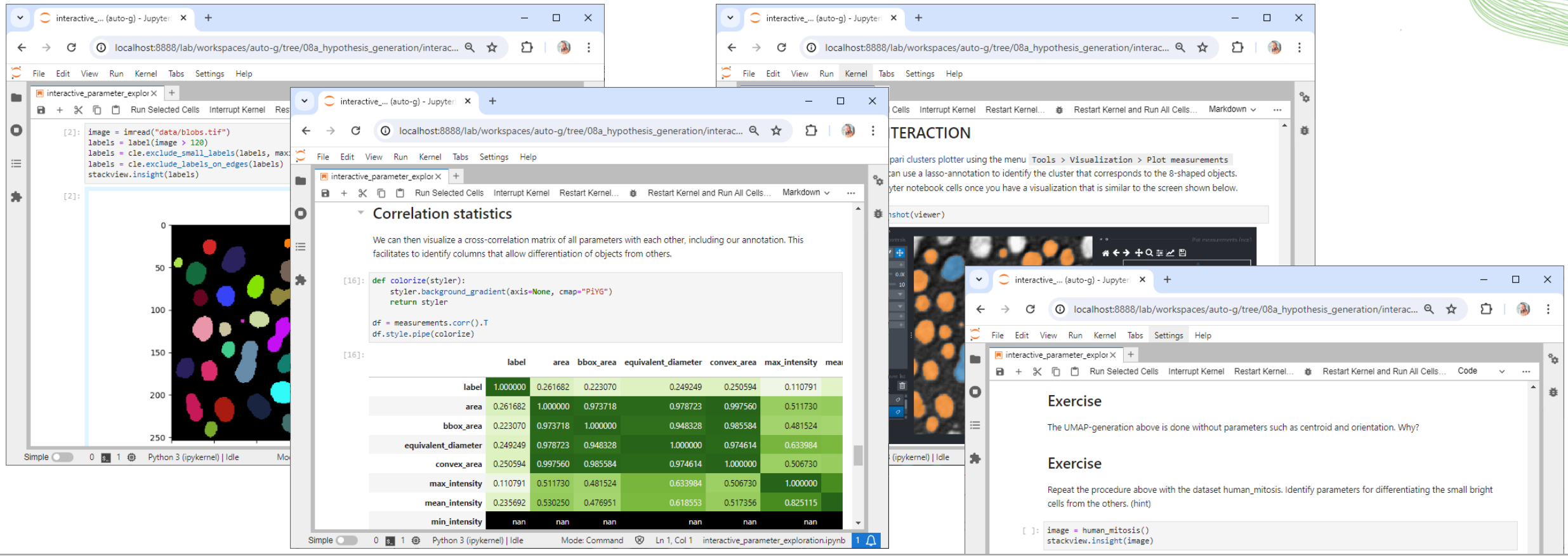
Bundesministerium für Bildung und Forschung

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

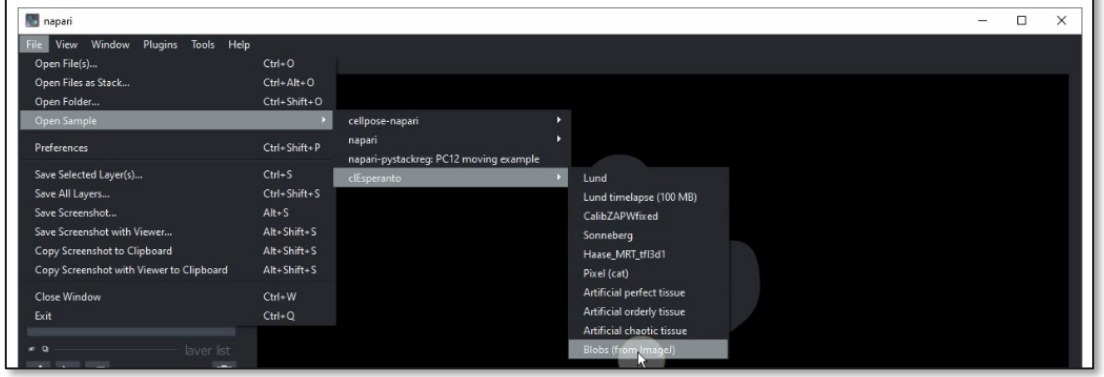
Open a terminal window and activate your conda environment:

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

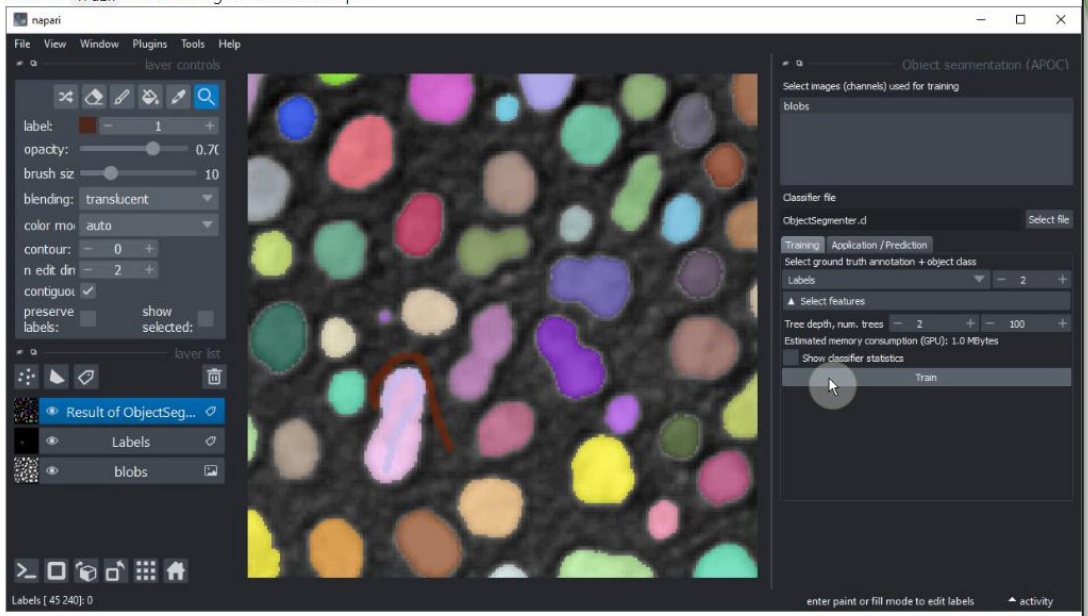
Afterwards, start up Napari:

```
napari
```

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



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:

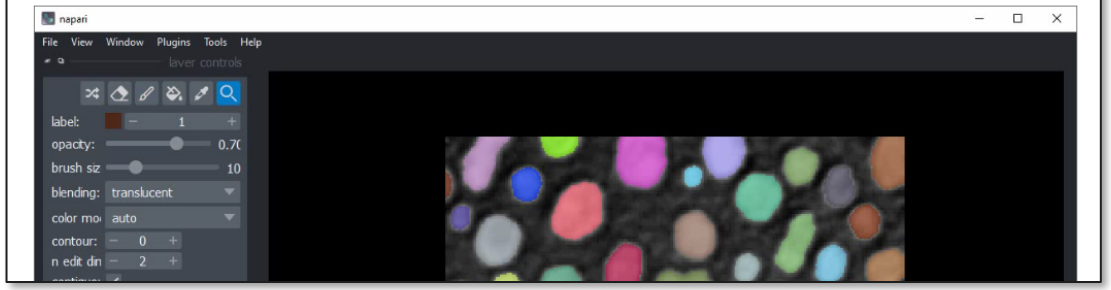
```
napari
```

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

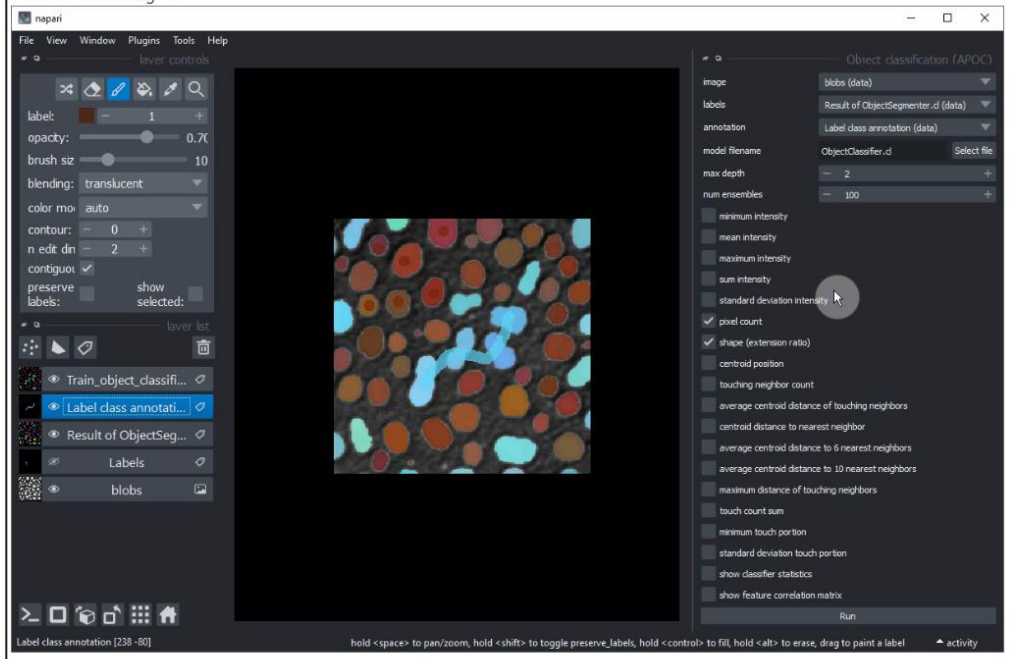
We furthermore need a label image. You can create it using the pixel classifier trained earlier or using the menu `Tools > Segmentation / labeling > Gauss-Otsu Labeling (clesperanto)`.

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.

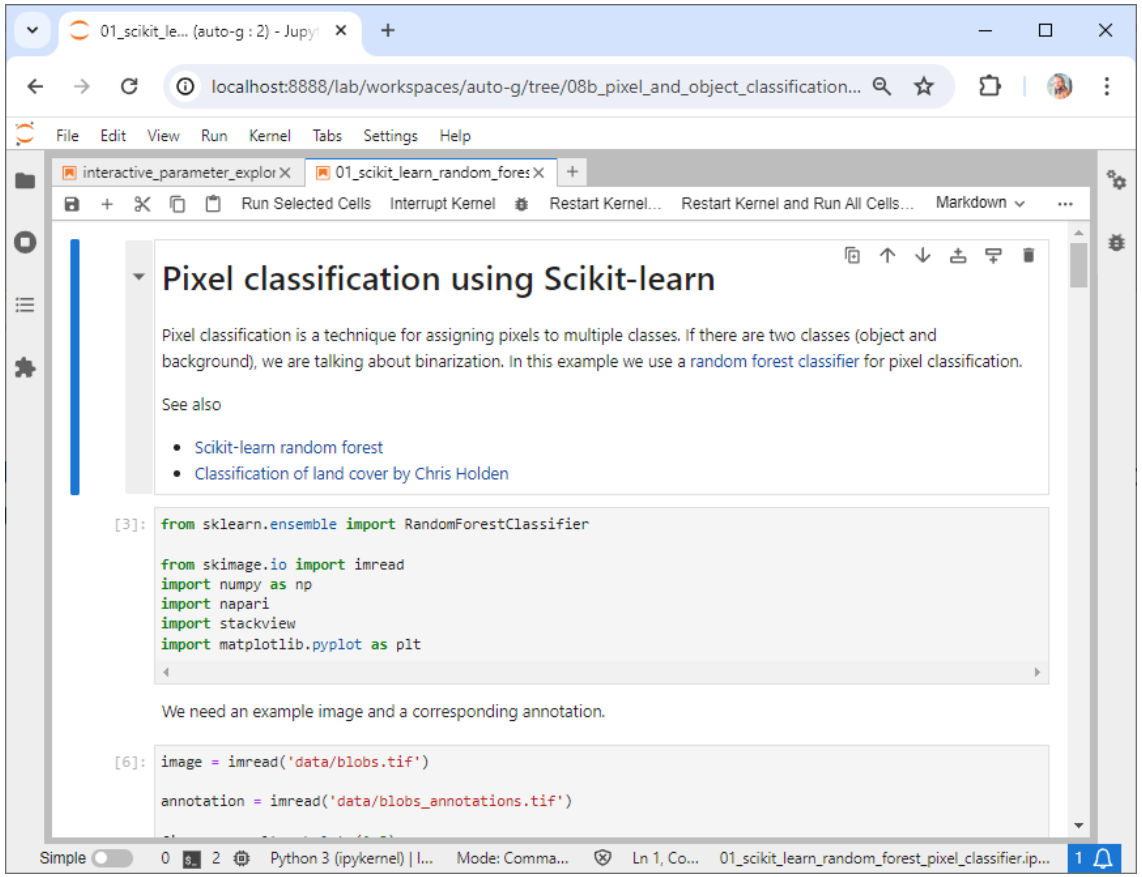
Extra exercise

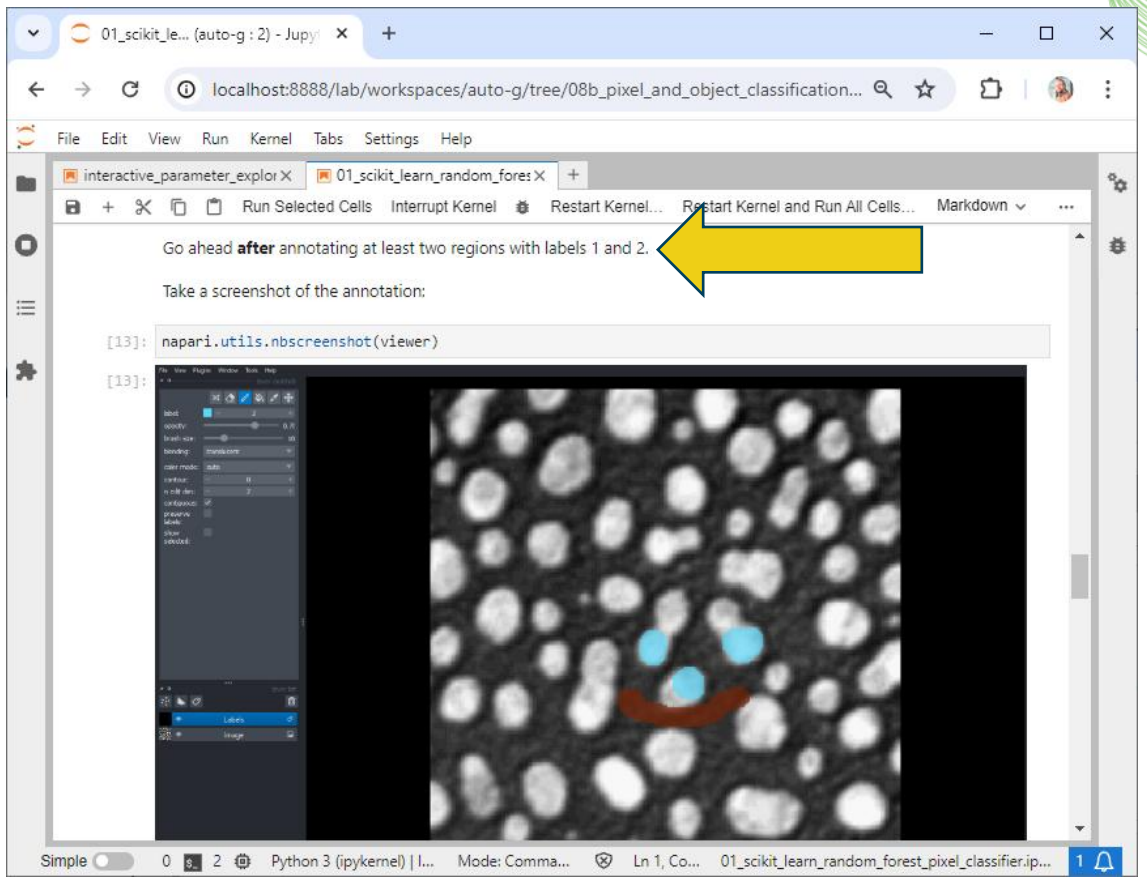
Retrain the classifier so that it can differentiate three different classes:

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

Supervised machine learning using Python

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





SHAP Analysis in Python

Use the opportunity and explain SHAP plots like this one!