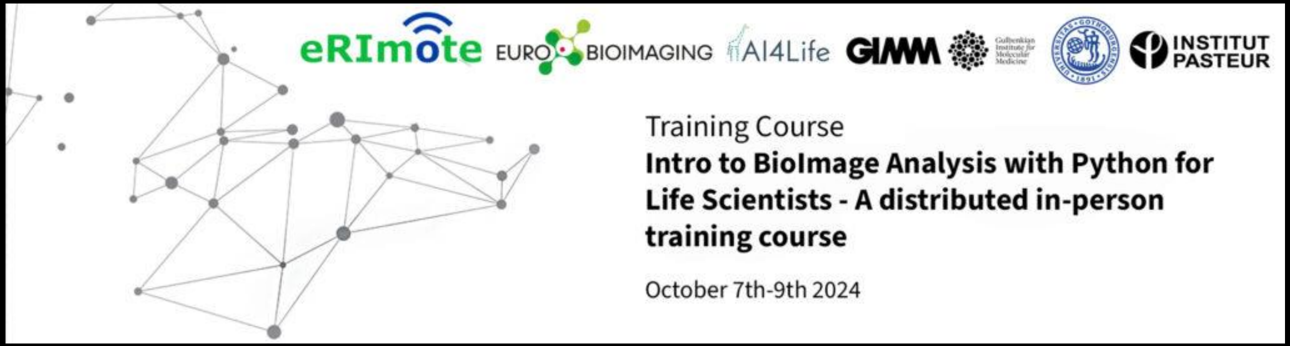


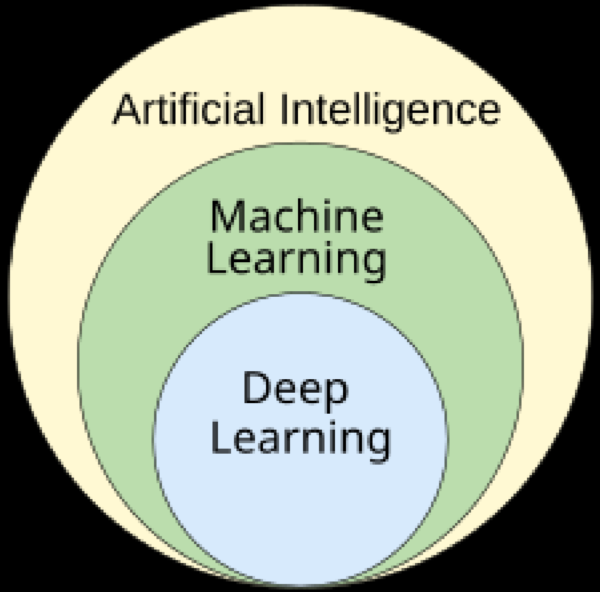
Day 3: Introduction to classical Machine Learning using scikit-learn

GIMM Gulbenkian Institute *for* Molecular Medicine

Bruno Saraiva | Postdoc Researcher
Ricardo Henriques' lab

9th October 2024

What is machine learning?

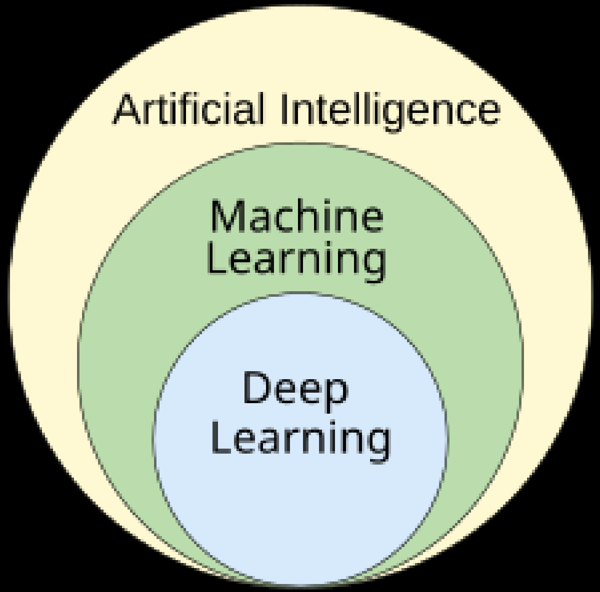


https://en.wikipedia.org/wiki/Machine_learning

In very simple terms, machine learning is a computational approach, where we use data to try to make predictions or decisions around new data or events that were not explicitly programmed

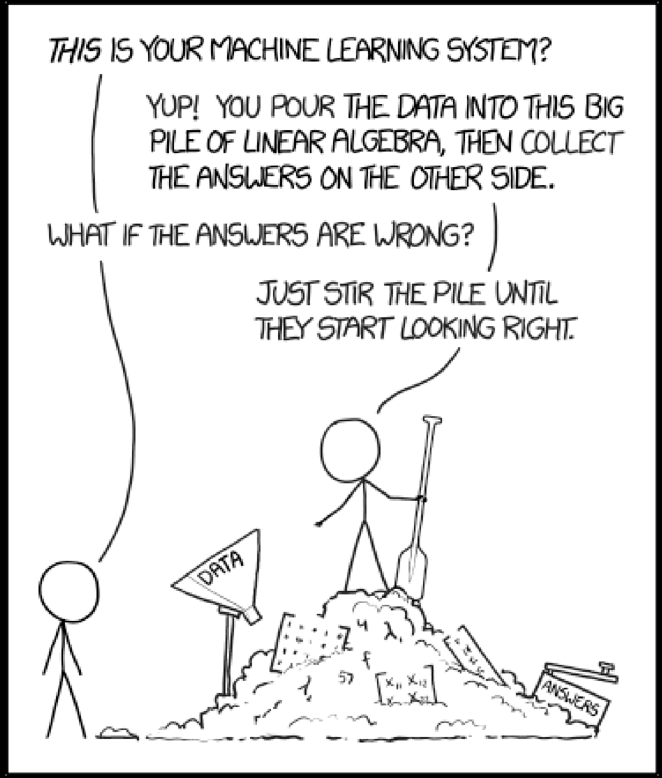
GIMM Gulbenkian Institute *for* Molecular Medicine

What is machine learning?



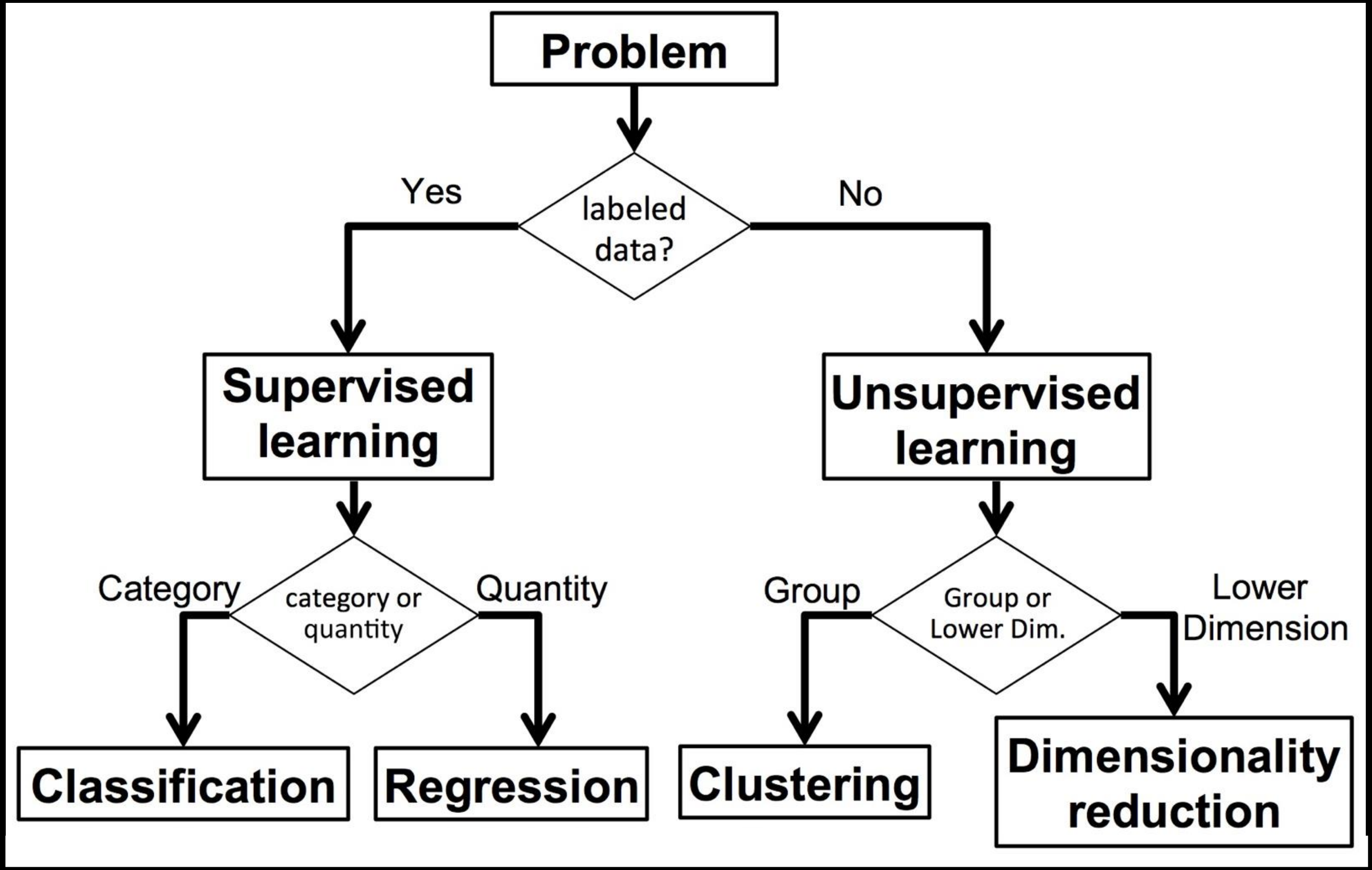
https://en.wikipedia.org/wiki/Machine_learning

In very simple terms, machine learning is a computational approach where we use data to try to make predictions or decisions around new data or events that were not explicitly programmed



https://xkcd.com/1838/

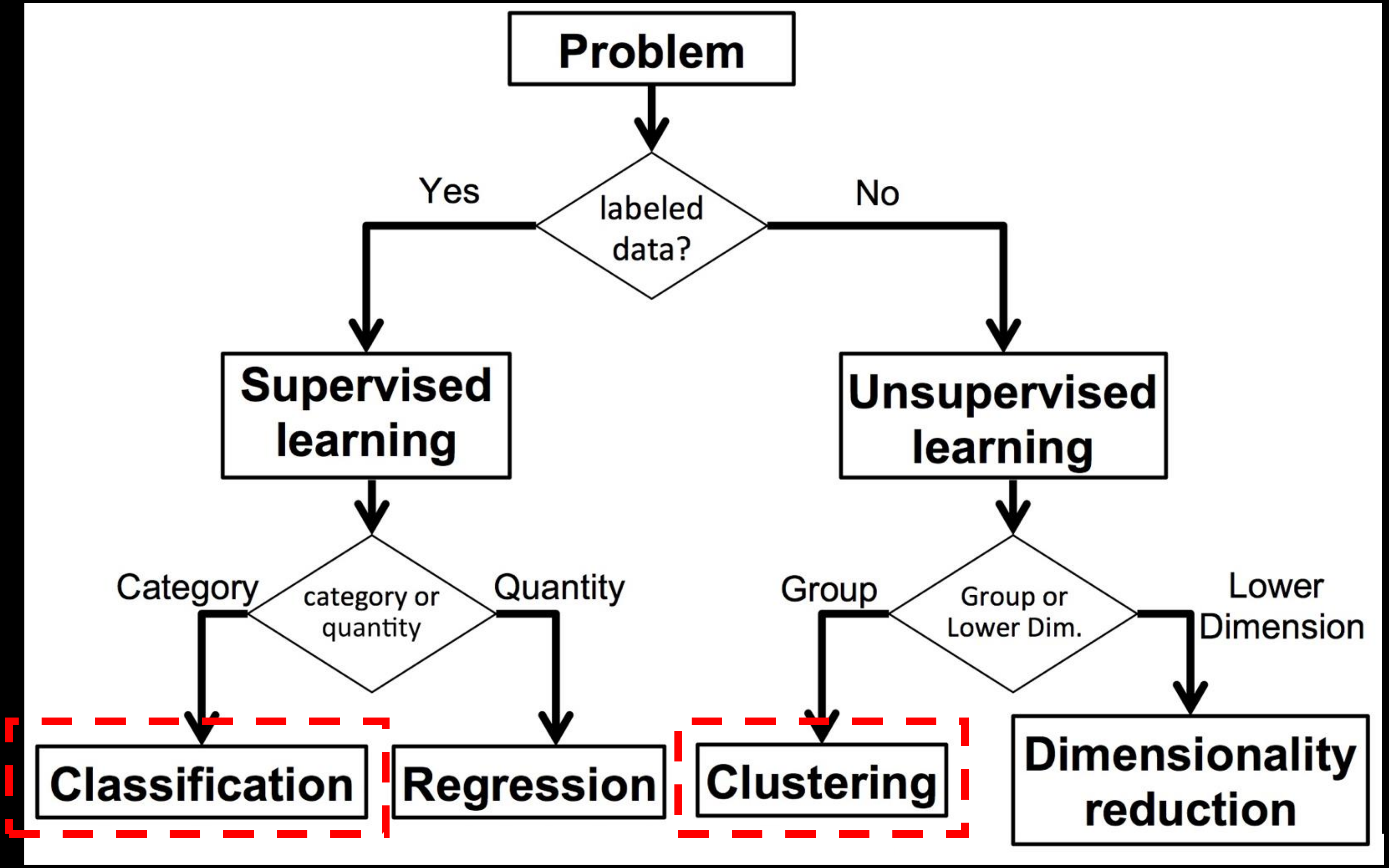
Different types of machine learning algorithms



https://link.springer.com/chapter/10.1007/978-981-99-3955-8_12

Different types of machine learning algorithms

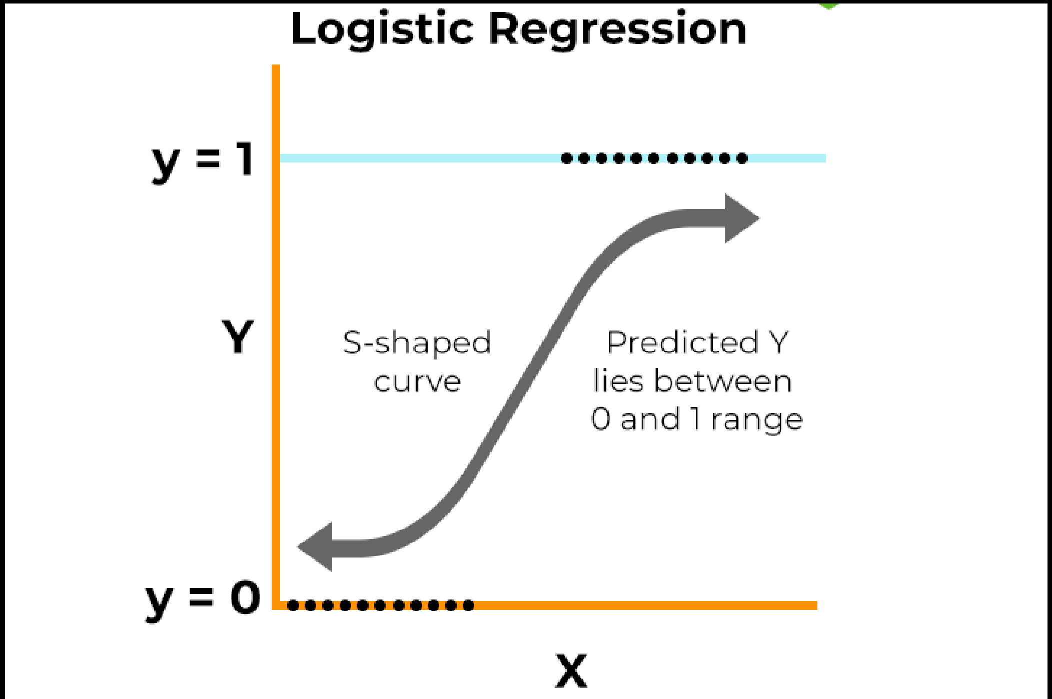
On the practical session we will be using examples of classification and clustering algorithms



https://link.springer.com/chapter/10.1007/978-981-99-3955-8_12

Common supervised machine learning algorithms

Logistic regression for classification tasks

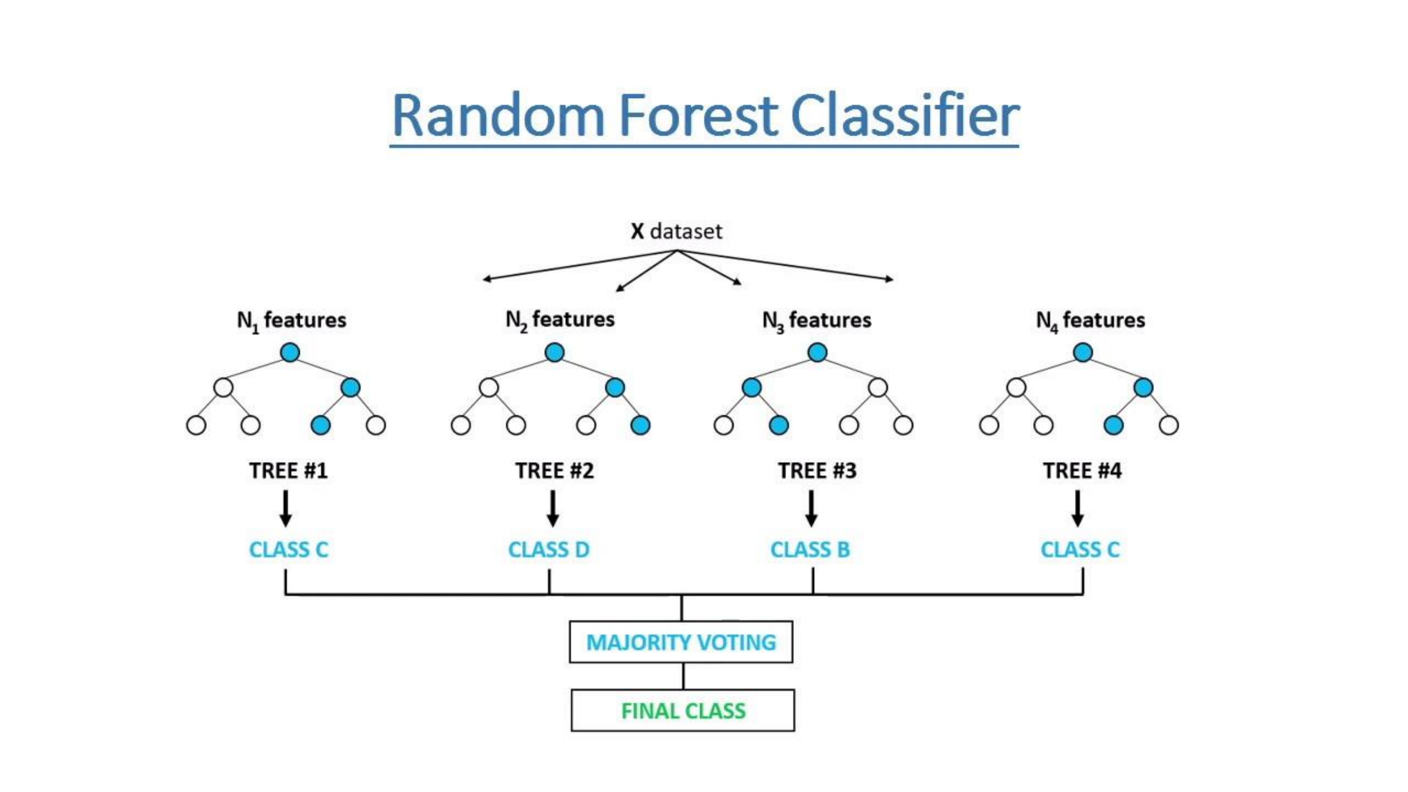


https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/

Logistic regression classification predicts the probability that a given data point belongs to a certain class by modeling the relationship between input features and the class label using a sigmoid function to output values between 0 and 1.

Common <u>supervised</u> machine learning algorithms

Random Forest classifier

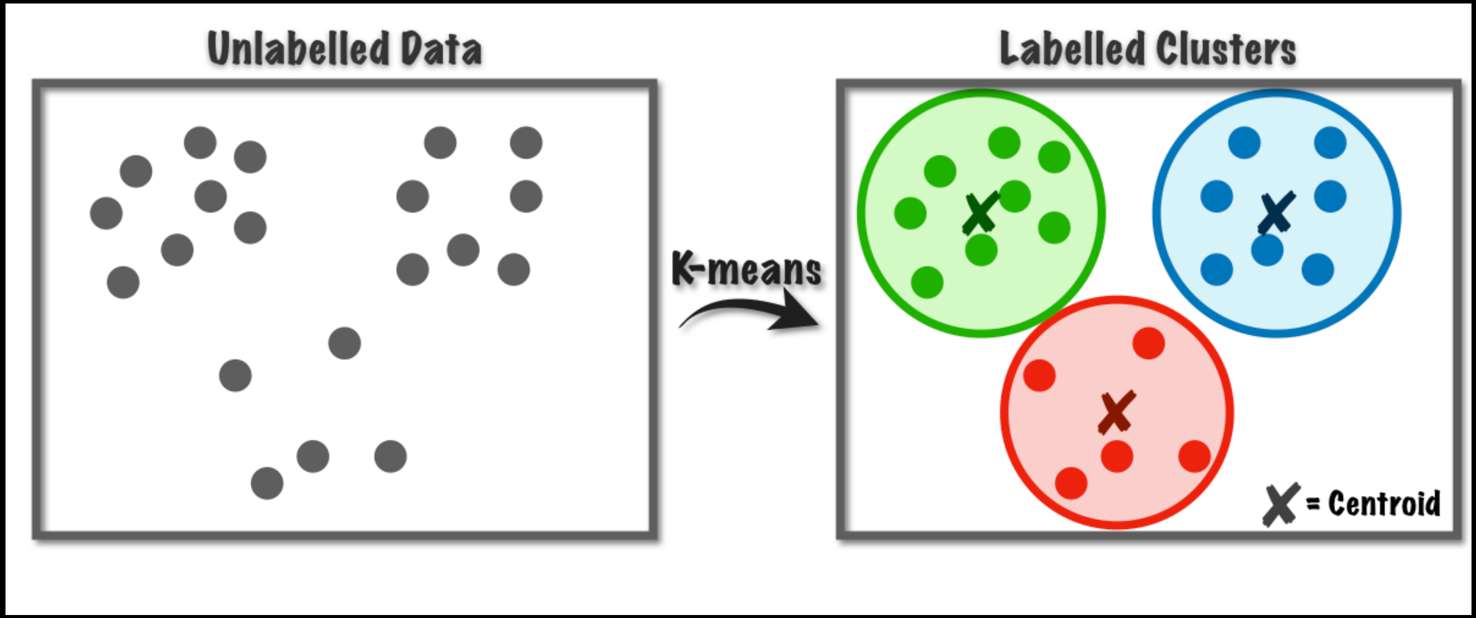


https://medium.com/@mrmaster907/introduction-random-forest-classification-by-example-6983d95c7b91

Random forest classification is an ensemble learning method that combines multiple decision trees. Each tree in a random forest uses a random subset of data features to split the data at each node, which helps reduce correlation between trees and improve overall model accuracy.

Common unsupervised machine learning algorithms

K-means clustering

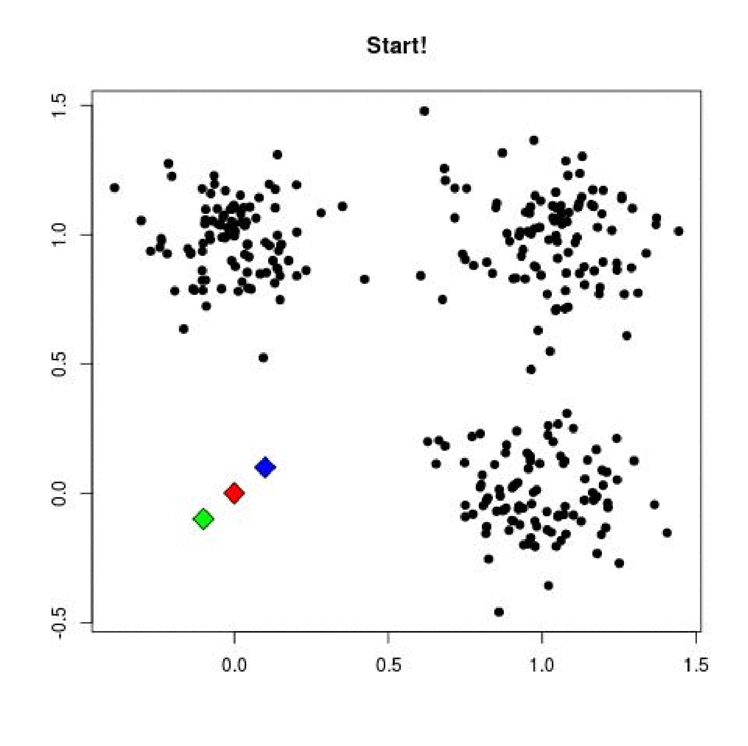


https://medium.com/@JenniferPuspita/k-means-clustering-c0a645715231

K-means clustering is an unsupervised learning algorithm that groups data points into a predefined number of clusters (k) by assigning each point to the nearest cluster center (centroid) and iteratively updating the centroids until the assignments stabilize, minimizing the variance within each cluster.

How does a machine learning algorithm learn?

An example with k-means clustering



https://mubaris.com/posts/kmeans-clustering/

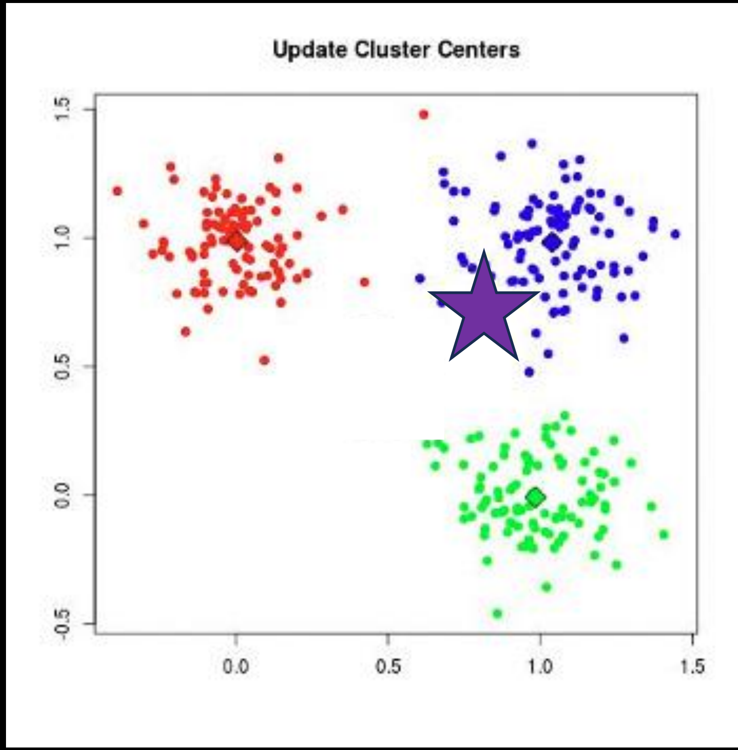
On each iteration k-means will define k number of centroids and assign each datapoint to the closest centroid

It continues doing so until it finds the position for each centroid that minimizes the variance withing each group

The same concept is exploited by every other machine learning algorithm where it’s trying to optimize according to a specific metric (group variance for k-means clustering).

How does it predict?

New datapoints are projected in the same dimensional space

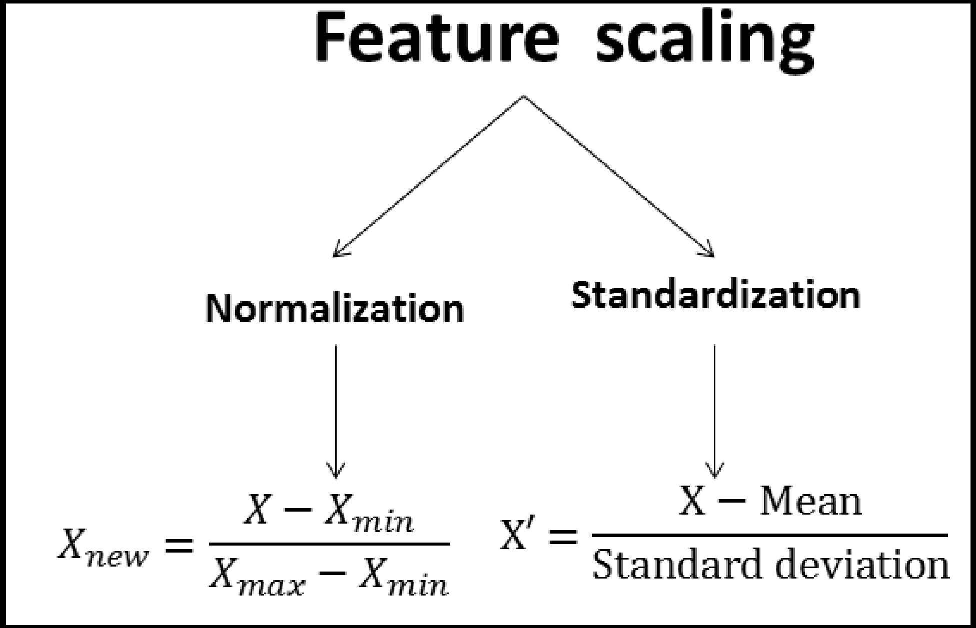


https://mubaris.com/posts/kmeans-clustering/

The new datapoint is closer to the “blue” centroid so it belongs to the ”blue” class

Preparing data for training

One important step on preparing your training data is normalization.



https://medium.com/aimonks/normalizing-scaling-and-standardizing-data-a-crucial-prelude-to-machine-learning-success-a71307f7c1f0

Especially important for ensuring training data and ”prediction” data are comparable, and the model can understand it.

It also ensures that different data “features” are comparable and have the same “weight”.

Preparing data for training

Effect of data standardization:

	Area	Perimeter	Length	Width	Eccentricity	Irregularity
	182	57	16.000000	13.000000	0.582961	4.225121
	211	64	16.552945	15.231546	0.391516	4.405942
	237	67	18.027756	14.866069	0.565685	4.352118
	227	66	17.000000	16.000000	0.337915	4.380574
	204	60	15.000000	15.000000	0.000000	4.200840
	217	66	17.204651	15.000000	0.489760	4.480372
	239	66	18.027756	16.155494	0.443760	4.269185
	239	66	17.691806	15.556349	0.476274	4.269185
	263	69	20.615528	16.401219	0.605854	4.254722
	263	72	18.027756	17.088007	0.318651	4.439710

→

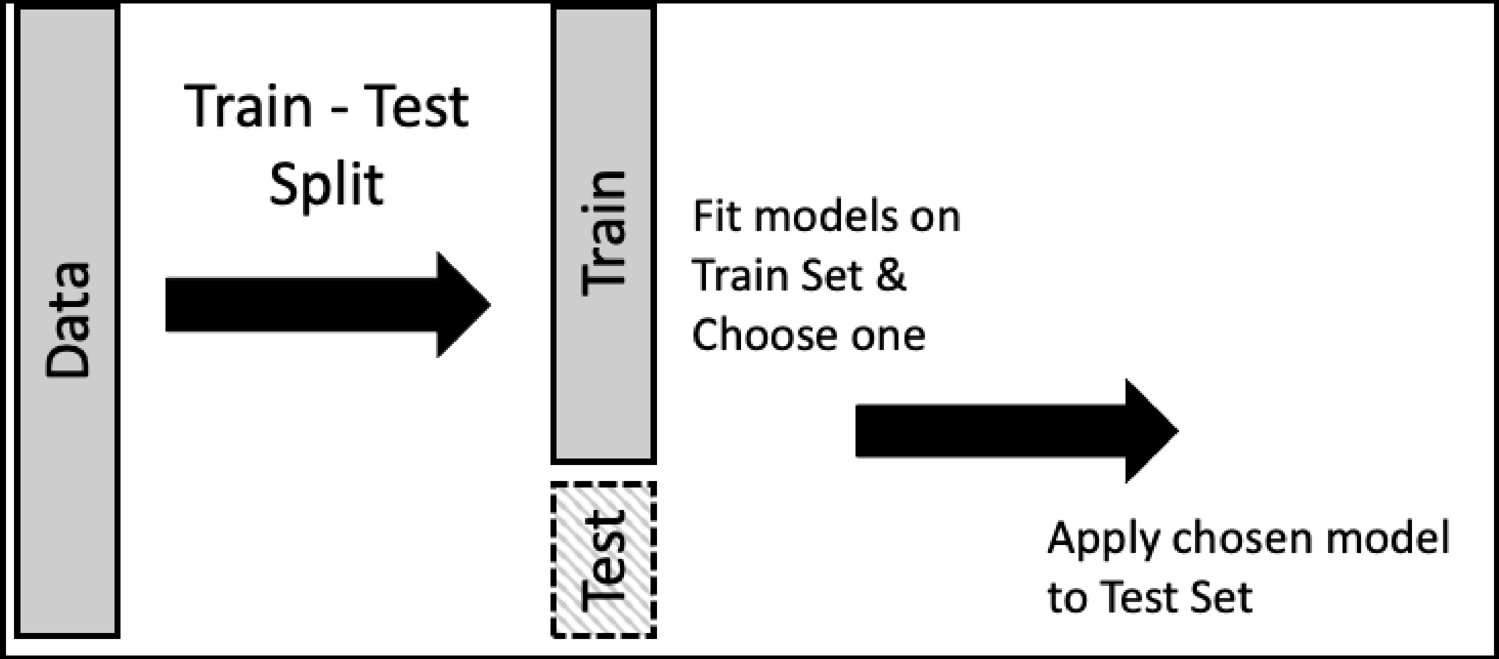
Area	Perimeter	Length	Width	Eccentricity	Irregularity
0.792537	1.690458	2.550826	-0.829119	2.268749	5.387692
1.645258	0.905877	0.670494	1.599536	-0.460539	-0.547352
-0.605365	-0.488935	-0.511774	0.022080	-0.700971	-0.096735
-0.102120	-0.140232	-0.386271	0.371875	-1.023854	-0.218632
0.163481	0.295647	0.135748	-0.185042	0.671941	0.181751
-0.381701	-0.401759	-0.256490	0.022080	-0.124583	-0.286012
1.477509	0.993052	0.670494	1.599536	-0.460539	-0.245740
-0.074162	-0.227408	-0.452346	0.261102	-0.999771	-0.415462
-1.304316	-0.488935	-0.564328	-1.077332	0.667521	1.254488
-0.689239	-0.576111	-0.883446	-0.411495	-0.945751	-0.126124

All values are now scaled so that each feature has mean=0 and std=1

Preparing data for training

To avoid our model just optimizing for the training data (overfitting), we also need to split data into train and test datasets

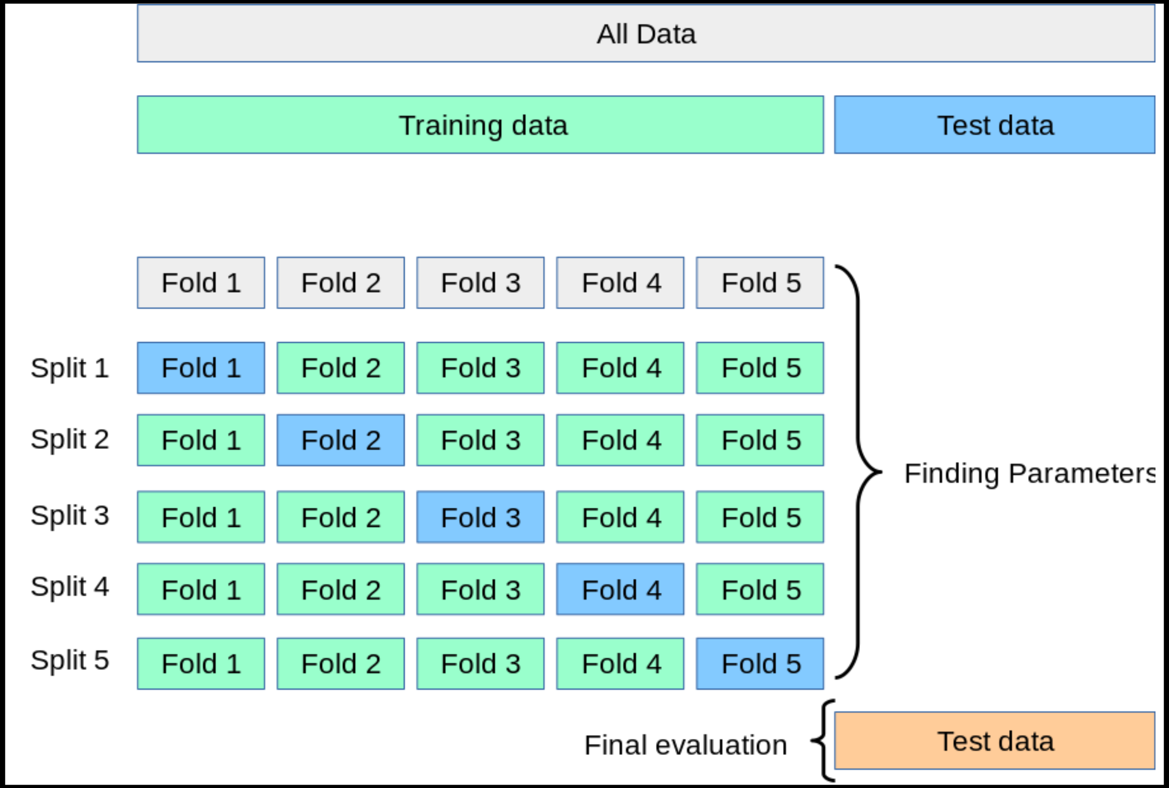
A common split is:
- 80% for training
- 20% for testing



https://learningds.org/ch/16/ms_train_test.html

K-fold cross validation

A better way to ensure model generalization



https://scikit-learn.org/stable/modules/cross_validation.html

Besides splitting the data into training and test data, we now also split the training data into multiple fold so that we can train the model on different subsets of the training data

We find the best parameters by choosing the ones that provide the best average evaluation across all different subsets

Finally, we obtain the final evaluation using the test dataset

What metrics can we use for evaluating our models?

For classification tasks the simplest measurement is accuracy:

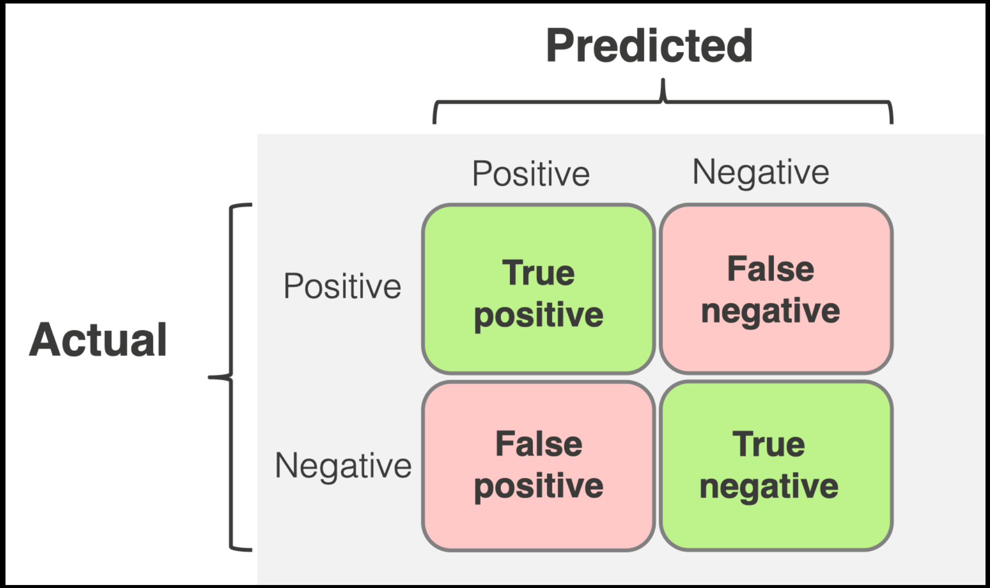
$$Accuracy = \frac{Correct\ predictions}{All\ predictions}$$

But it is not a very good metric…

It doesn’t tell us the full story.
For example, are we getting a good accuracy because the model is very good at predicting the most represented class?

What metrics can we use for evaluating our models?

Using a confusion matrix to better understand our model



With a confusion matrix we are able to understand how the model behaves for each class:

- Is the accuracy the same for all classes?
- Is the model always picking the same class?
- …

What metrics can we use for evaluating our models?

Using a confusion matrix to better understand our model

		Predicted: Positive	Predicted: Negative
Actual	Positive	**True positive**	**False negative**
	Negative	**False positive**	**True negative**

With it we can calculate better metrics than accuracy:

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

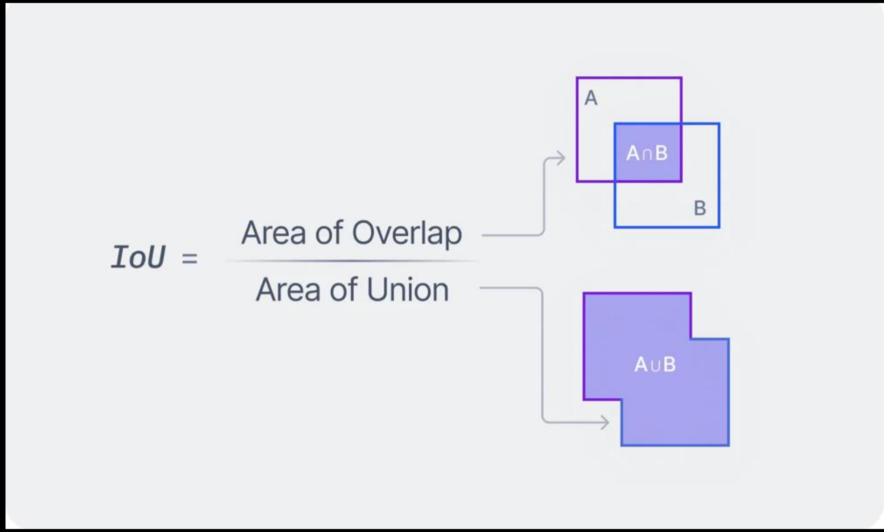
Tell us how many positives truly positive?

$$Recall = \frac{TP}{TP + FN}$$

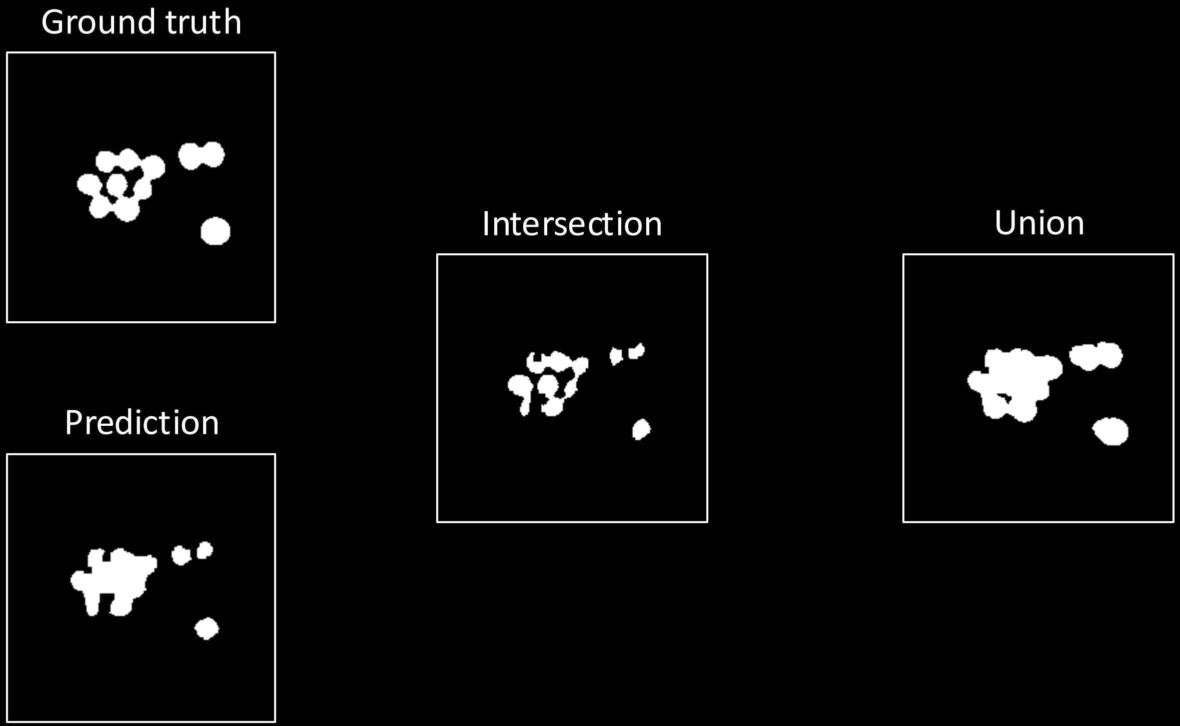
Tell us how many of all positives did we get right?

Evaluating the performance of segmentation tasks

Most used metric is Intersection over Union (IoU) between the ground truth and the predicted segmentation

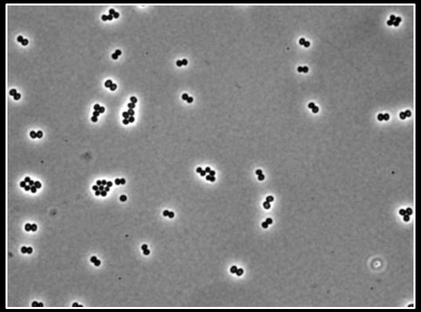


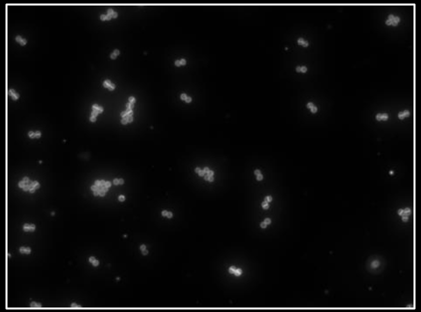
https://www.v7labs.com/blog/intersection-over-union-guide

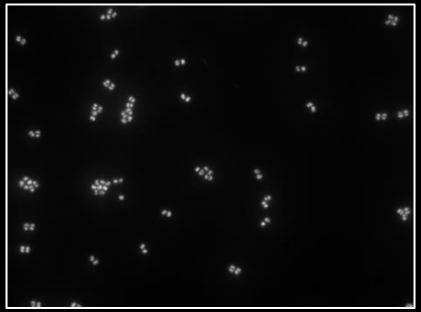


Agenda for the morning

Notebook #1- Image segmentation using classical machine learning







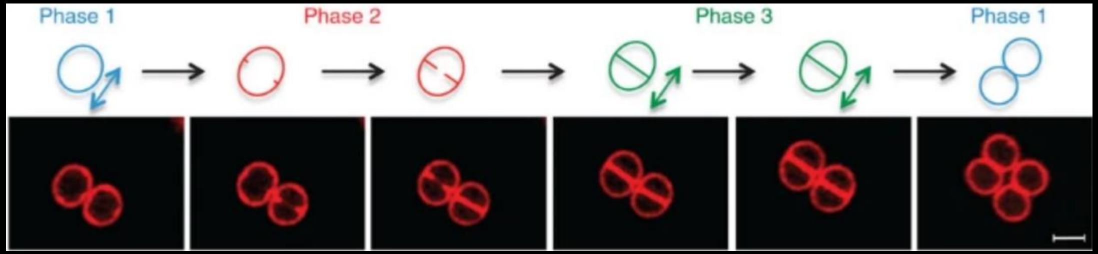
Semantic segmentation

K-means clustering
Random-forest classifier
K-neighbours classification

Segmentation quality measurement through IoU

Agenda for the morning

Notebook #2 – Automatic Cell Cycle classification of *S. aureus* cells



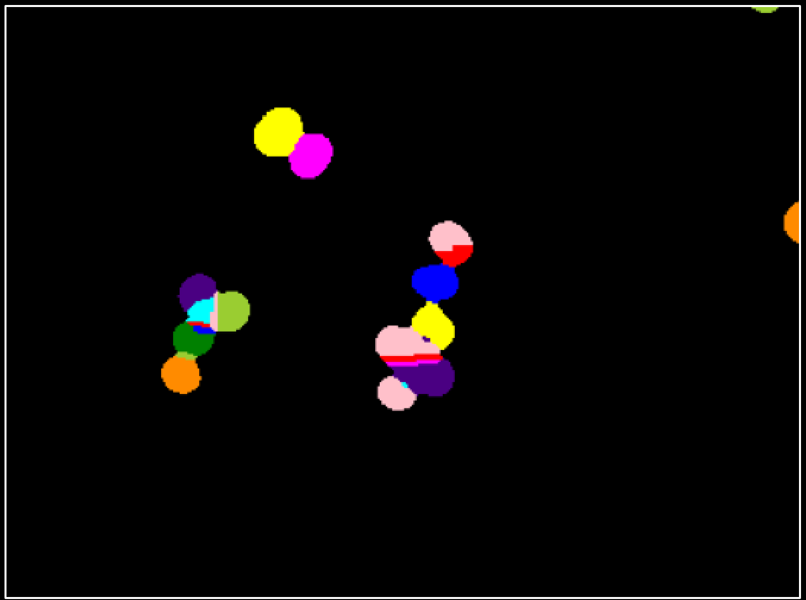
	Cell ID	Area	Perimeter	Length	Width	Eccentricity	Irregularity	Cell Cycle Phase
0	1	182	57	16.000000	13.000000	0.582961	4.225121	1
1	2	211	64	16.552945	15.231546	0.391516	4.405942	1
2	3	237	67	18.027756	14.866069	0.565685	4.352118	1
3	4	227	66	17.000000	16.000000	0.337915	4.380574	1
4	5	204	60	15.000000	15.000000	0.000000	4.200840	1
...	...	...	...	...	...	...	...	...
529	530	389	85	24.351591	20.248457	0.555518	4.309671	1
530	531	420	87	23.000000	21.000000	0.407862	4.245165	2
531	532	450	91	25.961510	20.615528	0.607813	4.289781	3
532	533	424	89	25.495098	21.213203	0.554700	4.322222	3
533	534	421	91	25.317978	20.223748	0.601610	4.435069	3

1. Loading datasets using Pandas
2. Exploring the data
3. Data formatting and normalization
4. Training a simple Logistic Regression model for the automatic classification task
5. Evaluation of the model using the confusion matrix
6. k-fold cross-validation
7. Fine-tuning a model with parameter sweep

Gulbenkian Institute *for* Molecular Medicine

Agenda for the morning

Notebook #3 (Optional) – Using machine learning to filter out badly segmented cells



	Cell ID	Area	Perimeter	Length	Width	Eccentricity	Well Segmented
0	1	209	71	18.601075	15.000000	0.591364	1
1	2	224	79	19.416488	15.811388	0.580405	1
2	3	263	87	24.166092	15.652476	0.761892	1
3	4	204	76	25.455844	12.041595	0.881042	1
4	5	229	81	20.615528	15.524175	0.657983	1

1. Instance segmentation from a binary image
2. Measuring morphological features
3. Training a classifier on manually annotated data of well and badly segmented cells
4. Filter out badly segmented cells using the trained classifier

Gulbenkian Institute *for* Molecular Medicine

Agenda for the morning

All notebooks, images and datasets are available in: https//github.com/brunomsaraiva/BioimageCourseGIMM

Instructions

1. Clone this repository to the BAND machine

Open a terminal and run:

```
git clone https://github.com/brunomsaraiva/BioimageCourseGIMM.git
```

2. Run Jupyter Lab and Open Notebook #1:

- Using BAND's Jupyter Lab -> Applications -> Programming -> Jupyter Lab
- Using previoysly created environment:

```
conda activate env_name
jupyter lab
```