Introduction to Machine Learning

Cédric Hassen-Khodja, Volker Baeker, Jean Bernard Fiche, Clément Benedetti

CNRS BioCampus MRI

2023-11-19

Outlines

- Introduction to Machine Learning
  - Definitions and Key Concepts
  - Types of Machine Learning
- Approaches and Applications
  - Data-Driven Approach
  - Image Classification
  - Classification Models and Algorithms
- Software Overview
  - CellProfiler & CellProfiler Analyst
  - QuPath

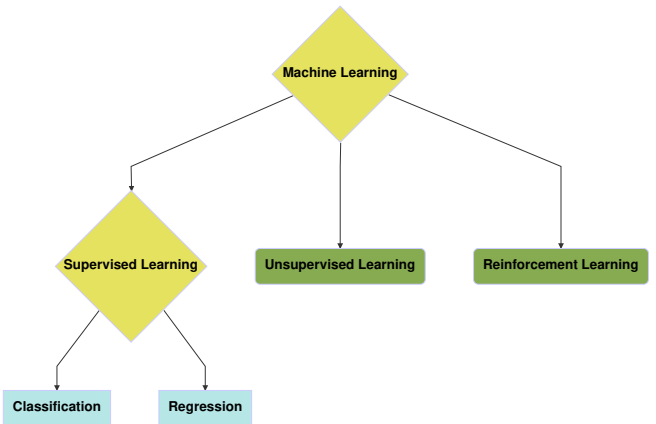
Machine Learning - Definitions

- **Machine Learning** is concerned with the technology that enables computer programs to improve their performance at a certain task by experience.
- In Machine Learning we want to infer (learn) some function `f` from data, capable of predicting the output `y` from an input (measurement) x:

$$y = f(x)$$

- The data is used to learn `f` is called **training set**.
- In this general formulation, there is no particular limitation as to the mathematical nature of `x` and `y` . In many cases `x` is a P-dimensional vector and `y` a categorical or continuous output variable, but there are other settings, where `x` and / or `y` are more complicated objects, such as images or graphs.

Types of Machine Learning

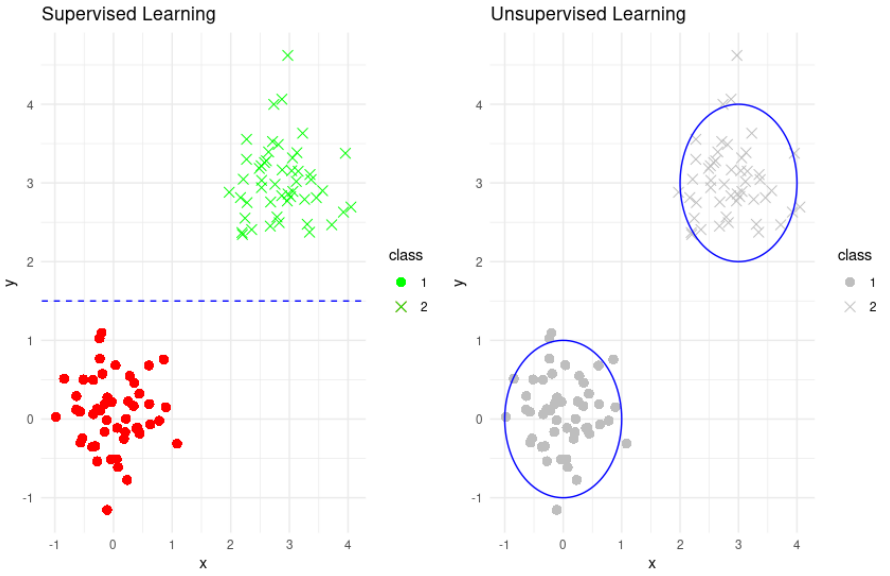


- In **supervised learning**, the training data contains both measurements x_i and the corresponding output variables y_i.
  Together, they build the training set T:

$$T = (x_i, y_i) i = 1, \ldots, N$$

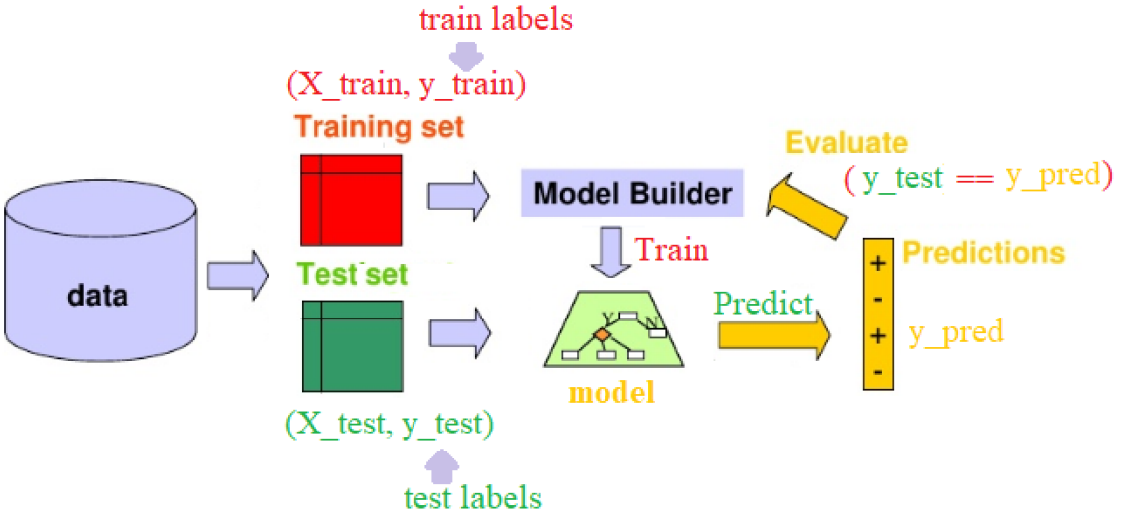
- In **unsupervised learning**, there are no annotations y_i. We aim at inferring **patterns** from the data.

Classification vs Clustering



- In **supervised learning**, we start from annotated data (here annotation is illustrated by the color), and we wish to learn a decision boundary that allows us to tell these classes apart.
- In **unsupervised learning**, we start with a point clouds and we wish to identify classes directly from their distribution.

Machine Learning: Data Driven Approach



1. Collect a dataset of images and labels.
2. The dataset will be splitted into **Training set** and **Test set**.
3. Use Machine Learning algorithm to train a **classifier** using the training dataset.
4. The classifier will be used to **predict** the labels of the images from the test dataset.
5. The predicted labels will be subsequently compared with the **ground-truth** labels to evaluate the performance of the classifiers.
6. Evaluate the classifier on **new images**.

Image classification

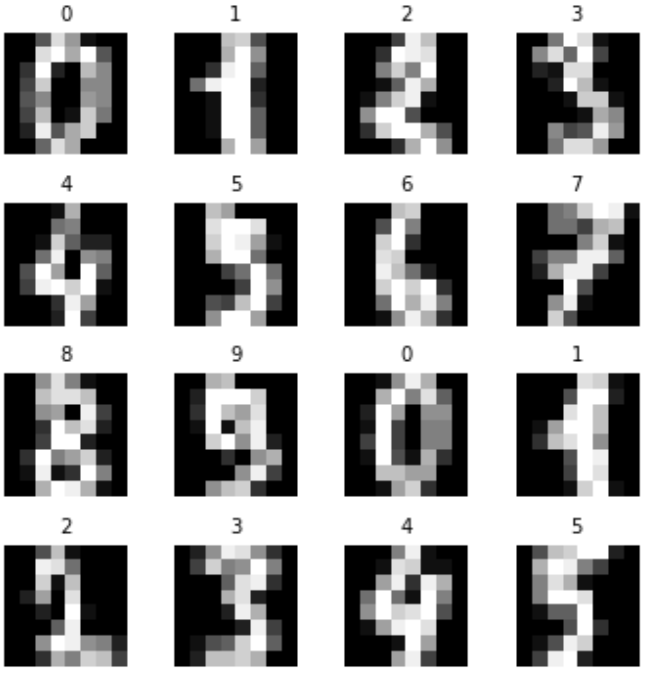
We are going to use a classic dataset in machine learning, https://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits, consisting of 8x8 grayscale images of handwritten digits.

Loading the MNIST (handwritten digits) dataset

```
# Import the digits dataset
from sklearn import datasets

# Load the digits dataset
digits = datasets.load_digits()
```

Visualizing the dataset

```
import numpy as np
import matplotlib.pyplot as plt

# Set the figure size
plt.figure(figsize=(5, 5))

# Display the first 16 images
for index, (image, label) in enu
    plt.subplot(4, 4, index + 1)
    plt.imshow(np.reshape(image,
    plt.title('%i' % label, font
    plt.axis('off')

plt.tight_layout()
plt.show()
```



Splitting the dataset

```
from sklearn.model_selection import train_test_split

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(digits.data,
digits.target, test_size=0.2, random_state=42)

# Display a summary of the data sets
print(f"Size of training data: {X_train.shape[0]}")
```

```
## Size of training data: 1437
```

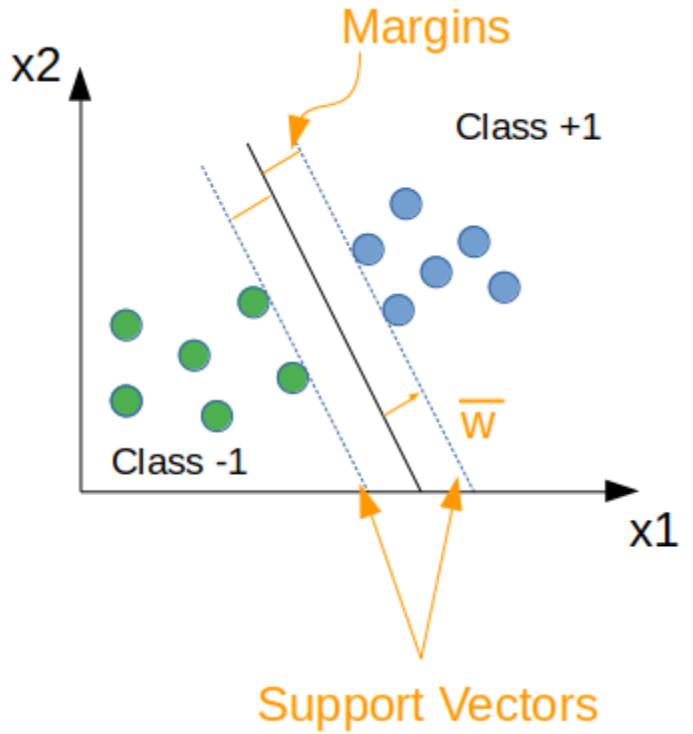
```
print(f"Size of test data: {X_test.shape[0]}")
```

```
## Size of test data: 360
```

SVM classifier

Support Vector Machine is a supervised learning algorithm which aims to separate the classes through a straight line, by maximizing the margins between the classes.

Hard Margin



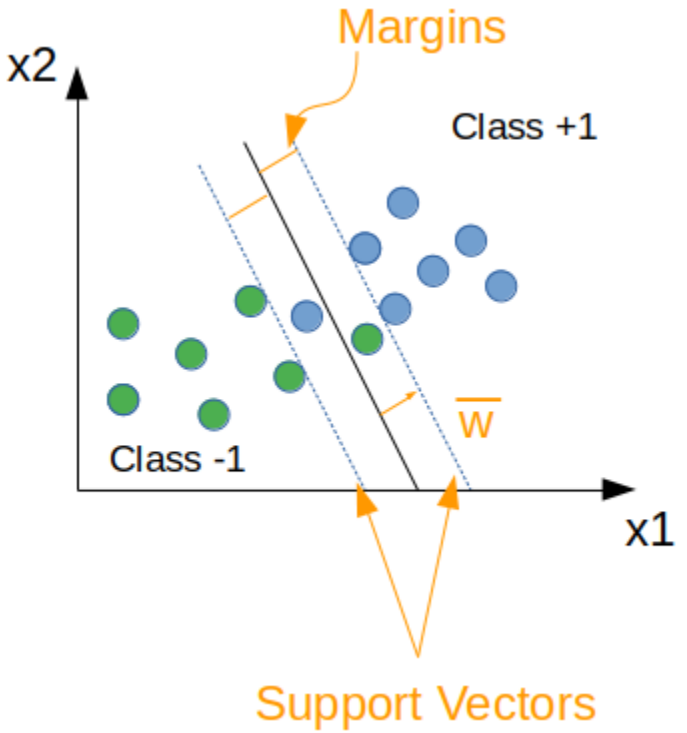
At inference time, what lies on the right side of the margin will be classified as "class 1" and vice versa.

Constraints..

$$y_i = +1 \mapsto \vec{w}\vec{x}_i + b \geq 1$$
$$y_i = -1 \mapsto \vec{w}\vec{x}_i + b \leq -1$$

SVM classifier

Soft Margin



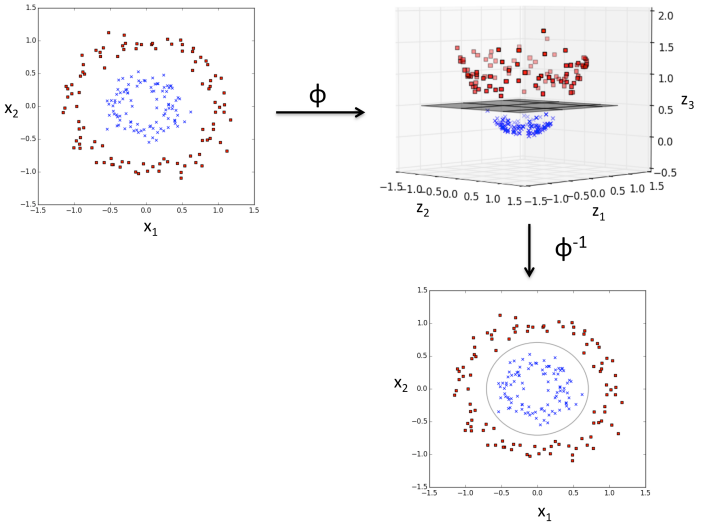
$$C \sum_i \varepsilon_i \; s.t. \; y_i(\vec{w}\vec{x}_i + b) \geq 1 - \varepsilon_i$$

$$\forall i = 1, \ldots, n$$

- if C is small, the margin is big
- If $C \mapsto \infty$, hard svm
- If $0 < \epsilon \leq 1$ (correct)
- If $\epsilon > 1$ (wrong)

SVM classifier

Kernel trick

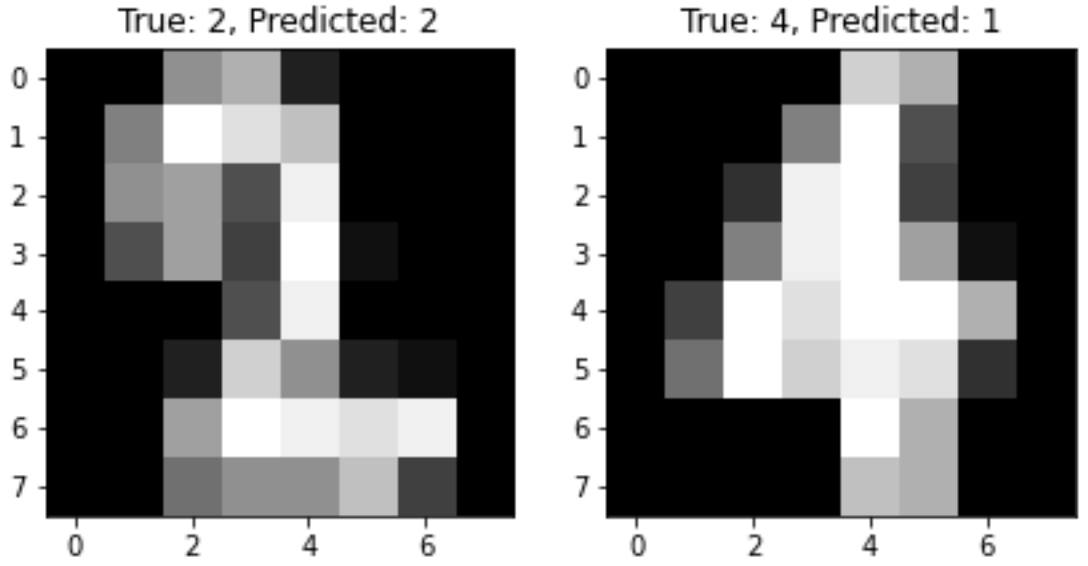


Common Kernels

- Sigmoid kernel:
  $K(x_i, x_j) = tanh(\alpha x_i^T x_j + c)$
- Polynomial kernel:
  $K(x_i, x_j) = (x_i . x_j + 1)^p$
- Radial Basis Function:
  $K(x_i, x_j) = exp(-\gamma(x_i - x_j)^2)$

Train, predict the SVM classifier using the MNIST dataset

```
from sklearn.svm import SVC

# Create an SVM classifier
svm = SVC(kernel='linear')

# Train the classifier
svm.fit(X_train, y_train);

# Predict using the classifier
y_pred = svm.predict(X_test)
```



Evaluate the SVM classifier

Confusion Matrix

	Predicted "non-spam"	**Predicted "spam"**
Actual "non-spam"	50	10
Actual "spam"	5	35

- **True Positives (TP)**: Emails that are actually spam and the algorithm correctly classified as such. In our example, there are 35 TPs.
- **True Negatives (TN)**: Emails that aren't spam and the algorithm correctly classified as non-spam. In our example, there are 50 TNs.
- **False Positives (FP)**: Emails that aren't spam but the algorithm incorrectly classified as spam. In our example, there are 10 FPs.
- **False Negatives (FN)**: Emails that are actually spam but the algorithm incorrectly classified as non-spam. In our example, there are 5 FNs.

Confusion matrix is useful to calculate precision, recall, and accuracy!!

Evaluate the SVM classifier (continued)

Precision

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

Out of all items predicted as positive by the model, how many were actually positive?

Recall

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

Out of all items that are actually positive, how many did the model correctly predict as positive?

Evaluate the SVM classifier (continued)

F1-Score

It's defined as the harmonic mean of precision and recall:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

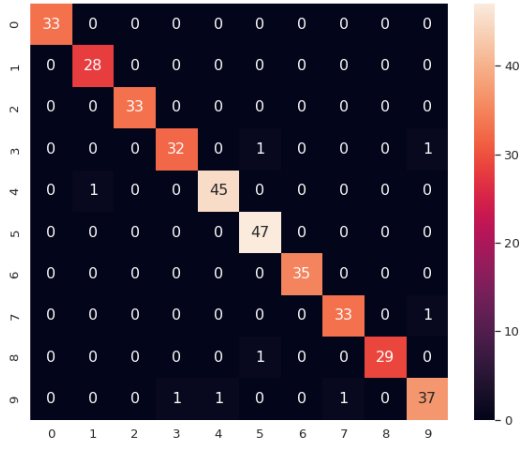
Accuracy

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Out of all predictions made by the model, how many were correct?

Evaluate the SVM classifier using the MNIST dataset

```
from sklearn import metrics
import pandas as pd
import seaborn as sn

# Confusion matrix
cm = metrics.confusion_matrix(y_test, y_pred)
df_cm = pd.DataFrame(cm, range(10), range(10))
sn.set(font_scale=1.2)
plt.figure(figsize=(10, 8))
sn.heatmap(df_cm, annot=True, annot_kws={"size":16}, fmt="g")
plt.show()
```



Evaluate the SVM classifier using the MNIST dataset (continued)

```
# Print the classification report
print(
    f"Classification report for classifier {svm}:\n"
    f"{metrics.classification_report(y_test, y_pred)}\n"
)
```

```
## Classification report for classifier SVC(kernel='linear'):
##               precision    recall  f1-score   support
##
##            0       1.00      1.00      1.00        33
##            1       0.97      1.00      0.98        28
##            2       1.00      1.00      1.00        33
##            3       0.97      0.94      0.96        34
##            4       0.98      0.98      0.98        46
##            5       0.96      1.00      0.98        47
##            6       1.00      1.00      1.00        35
##            7       0.97      0.97      0.97        34
##            8       1.00      0.97      0.98        30
##            9       0.95      0.93      0.94        40
##
##     accuracy                           0.98       360
##    macro avg       0.98      0.98      0.98       360
## weighted avg       0.98      0.98      0.98       360
```

Tuning the hyper-parameters of an classifier

GridSearchCV

- It helps to loop through predefined hyperparameters and fit your estimator (model) on your training set.
- You can select the best parameters from the listed hyperparameters.

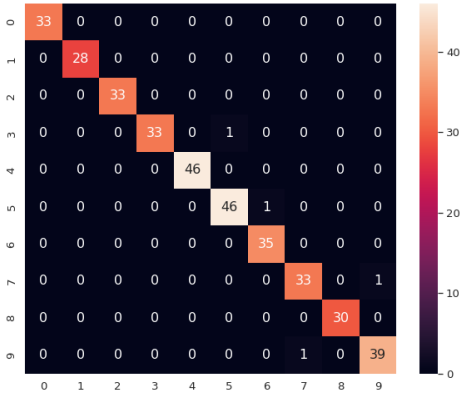
Tuning the hyper-parameters of the SVM classifier using the MNIST dataset

```
from sklearn.model_selection import GridSearchCV

# Defining the hyperparameter grid
param_grid = {
    'C': [0.1, 1, 10, 100],
    'gamma': [0.0001, 0.001, 0.1, 1],
    'kernel': ['linear', 'rbf', 'poly']
}

# Initializing the SVM classifier
svc = SVC()

# Setting up GridSearchCV
model = GridSearchCV(svc, param_grid, n_jobs=10)

# Fitting the model with the training data
model.fit(X_train, y_train);

# After the grid search, you can get the best parameters like this:
print("Best parameters found:", model.best_params_)
```

```
## Best parameters found: {'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}
```

Predict and Evaluate the "Best Model"

```
# Predict using the best model
y_pred = model.best_estimator_.predict(X_test)

# Compute the confusion matrix
cm = metrics.confusion_matrix(y_test, y_pred)

# Visualize the confusion matrix using seaborn
df_cm = pd.DataFrame(cm, range(10), range(10))
sn.set(font_scale=1.2)
plt.figure(figsize=(10, 8))
sn.heatmap(df_cm, annot=True, annot_kws={"size":16}, fmt="g")
plt.show()
```



Evaluate the best classifier

```
# Print the classification report
print(
    f"Classification report for classifier {model.best_estimator_}:\n
    f"{metrics.classification_report(y_test, y_pred)}\n"
)
```

```
## Classification report for classifier SVC(C=10, gamma=0.001):
##               precision    recall  f1-score   support
##
##            0       1.00      1.00      1.00        33
##            1       1.00      1.00      1.00        28
##            2       1.00      1.00      1.00        33
##            3       1.00      0.97      0.99        34
##            4       1.00      1.00      1.00        46
##            5       0.98      0.98      0.98        47
##            6       0.97      1.00      0.99        35
##            7       0.97      0.97      0.97        34
##            8       1.00      1.00      1.00        30
##            9       0.97      0.97      0.97        40
##
##     accuracy                           0.99       360
##    macro avg       0.99      0.99      0.99       360
## weighted avg       0.99      0.99      0.99       360
```

From SVM to Random Forest: Addressing Weaknesses

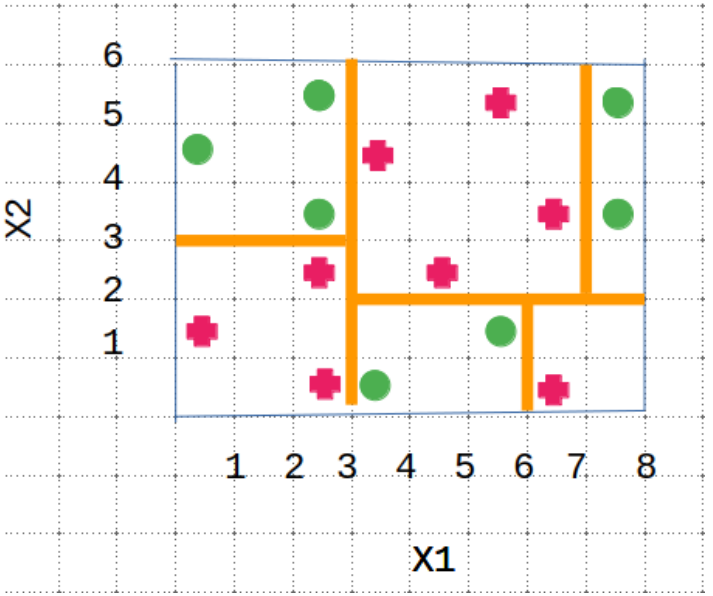
Weaknesses of SVM

- **Sensitivity to High Dimensionality**: SVMs can become less effective as the dimensionality and size of the dataset increase.
- **Kernel Selection and Tuning**: Choosing and tuning the appropriate kernel can be complex and non-intuitive.
- **Less Efficient on Large Data**: SVMs require longer computation time for large datasets.
- **Difficulty in Interpretation**: The decision boundaries in SVMs, especially with non-linear kernels, are often hard to interpret.

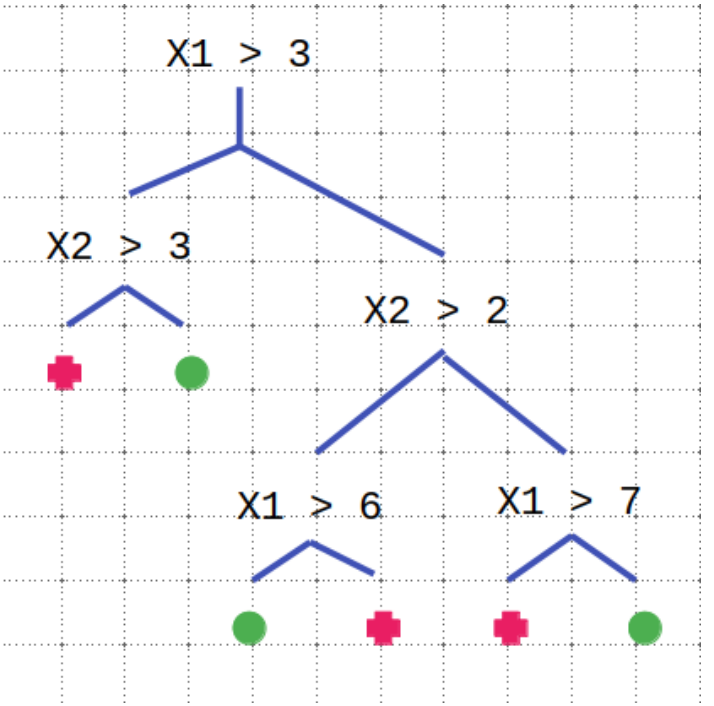
Advantages of Random Forest

- **Effective Handling of High Dimensionality**: Performs well even with a large number of features and data.
- **Robustness to Noisy Data**: Tolerates noise and outliers well.
- **Interpretability**: Provides more intuitive understanding through the aggregation of individual tree decisions.
- **Flexibility and Versatility**: Effective for a wide range of classification and regression tasks, with less risk of overfitting.

Decision tree - Algorithm



The (binary) decision tree



Decision tree - Node splitting

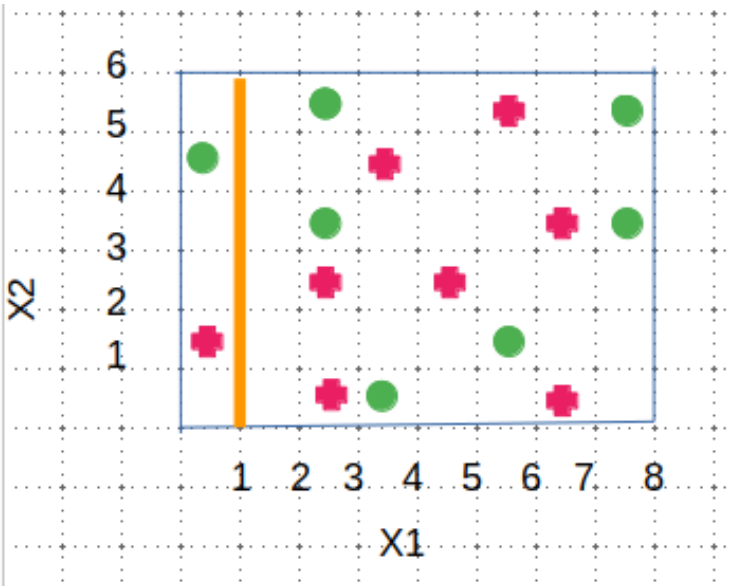
- **Gini impurity**: Select the one having lowest Gini Impurity

$$GiniImpurity = 1 - Gini$$
$$Gini = \sum_i (p_i^2)$$

- **Information Gain**: Select the one having highest IG or lowest Entropy

$$IG = 1 - Entropy$$
$$Entropy = -\sum_i (p_i log_2 p_i)$$

Example



	#O1	#X1	#O2	#X2	Gini1	Gini2	Gini
X1 > 1	1	1	6	7	0.5	0.5	1
X1 > 3	3	3	4	5	0.5	0.49	0.99
X1 > 4	4	4	3	4	0.5	0.49	0.99
X1 > 5	4	5	3	3	0.49	0.5	0.99
X1 > 6	5	5	2	3	0.49	0.5	0.99
X1 > 7	5	8	2	0	0.47	0	0.47
X2 > 1	1	2	6	6	0.44	0.5	0.94
X2 > 2	2	3	5	5	0.48	0.5	0.98
X2 > 3	2	5	5	3	0.41	0.47	0.88
X2 > 4	4	6	3	2	0.48	0.48	0.96
X2 > 5	5	7	2	1	0.49	0.44	0.93

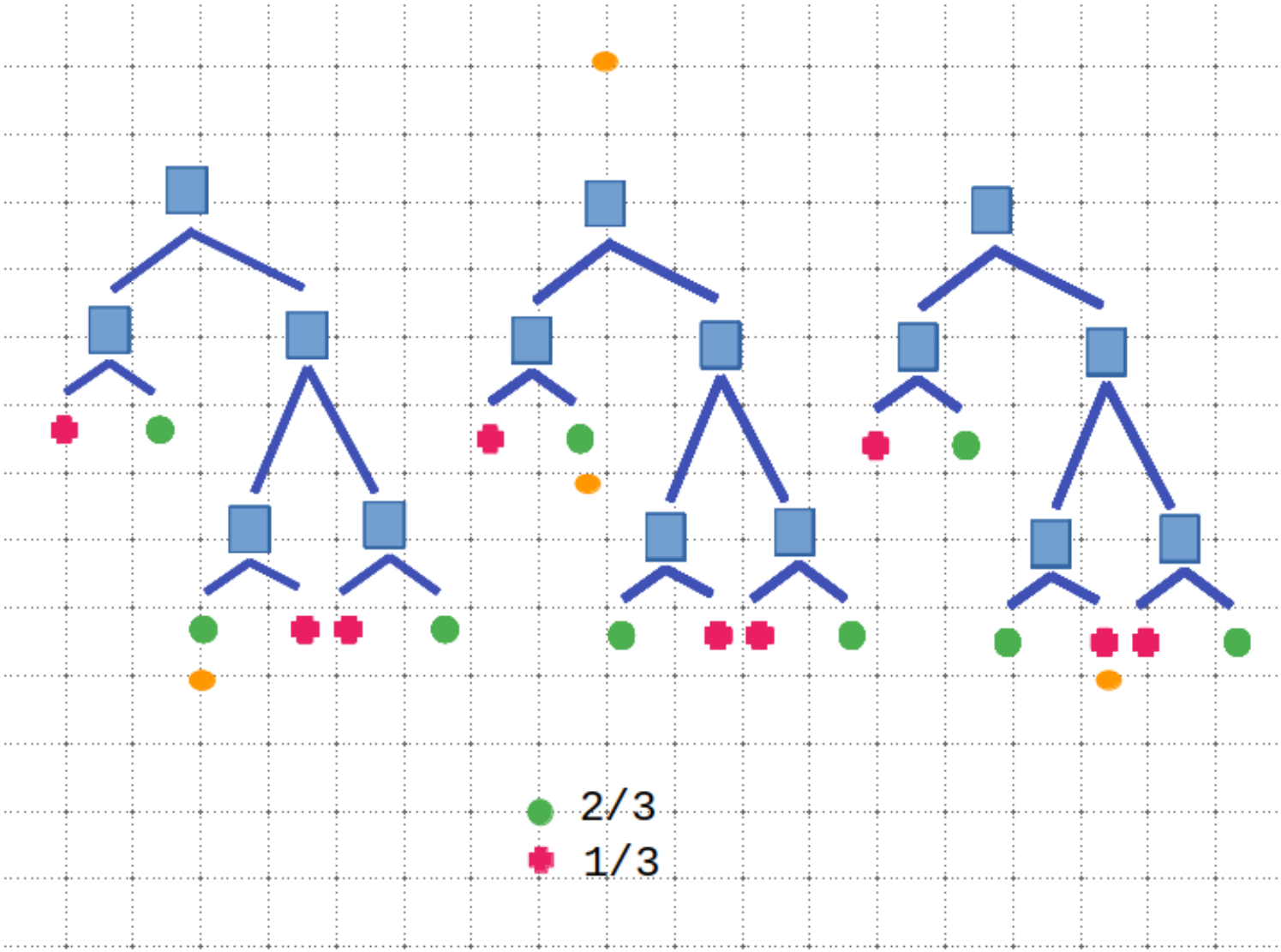
Pros and cons of decision trees

- Advantages
  - Simple to interpret.
  - Can handle numerical and categorical data.
  - Little data preparation.
  - Fast.
- Disadvanages
  - Susceptible to noise.
  - Greedy algorithm is locally optimal.
  - Chance of overfitting.

Random Forest

- Random forest is a **collection of decision trees**.
- Each tree is trained on a **bootstrap sample** of the training data (random sampling with replacement)
- Each node considers a **random subset** of dimensions.
- As a consequence, each DT is different.
- At runtine:
  - The new observation is **classified by all N trees**.
  - **Majority vote**.
  - Additionally provides **uncertainty** information.

Classification using random forest



Apply the Random Forest classifier on MNIST dataset

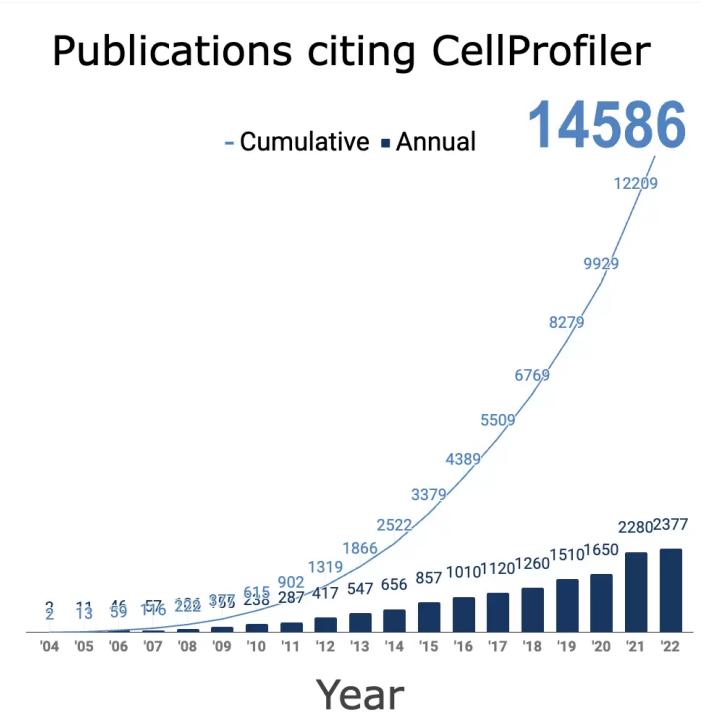
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Load MNIST dataset
digits = datasets.load_digits()

# Split the data into a training set and a test set
X_train, X_test, y_train, y_test = train_test_split(digits.data, dig

# Initialize the classifier
clf = RandomForestClassifier(n_estimators=100)

# Train the classifier
clf.fit(X_train, y_train);

# Make predictions on the test data
y_pred = clf.predict(X_test)

# Calculate and print the accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy of Random Forest Classifier: {accuracy:.4f}")
```

```
## Accuracy of Random Forest Classifier: 0.9622
```

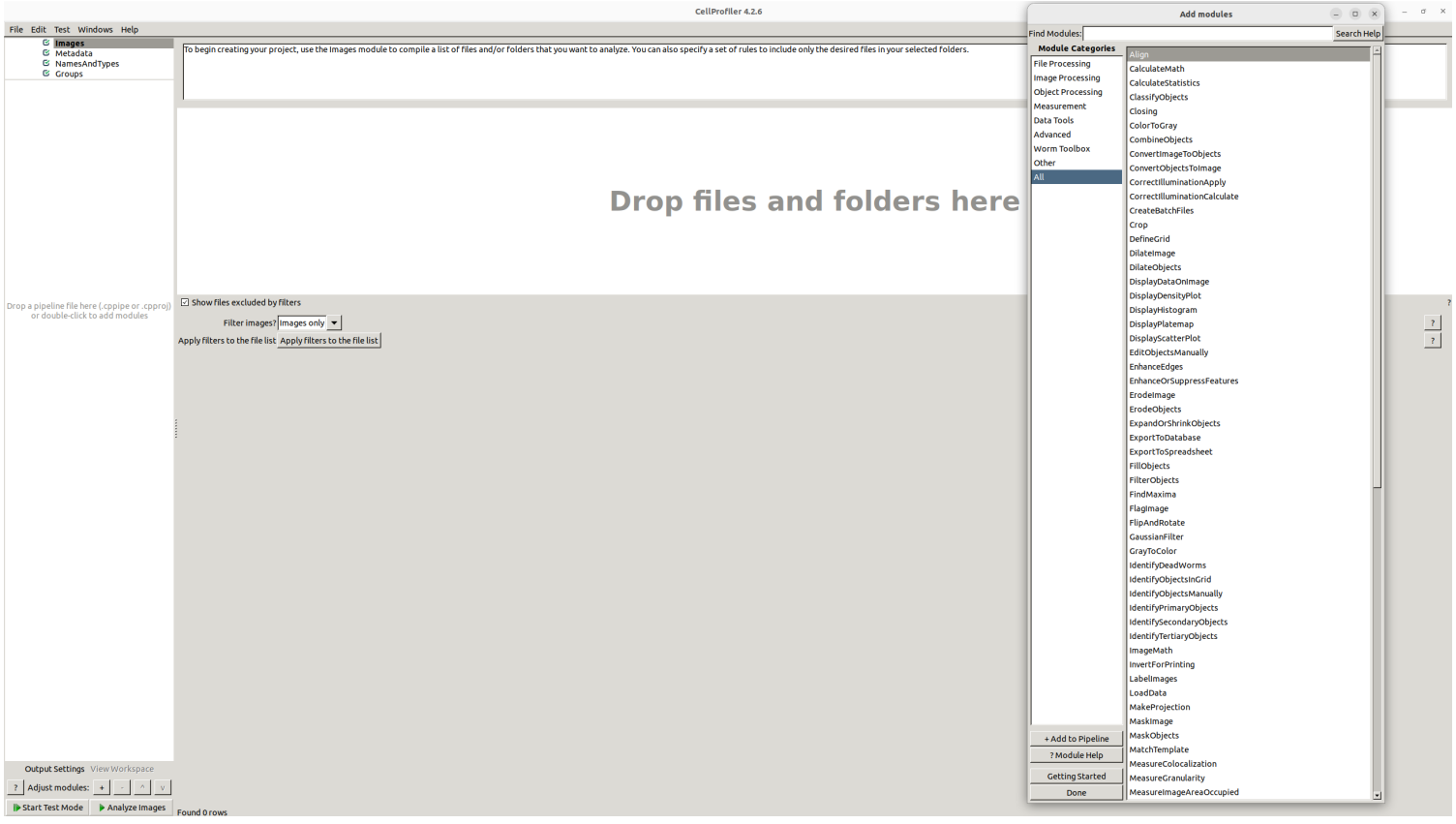
Introduction to CellProfiler

Open-source software for image analysis

- Free and open-source; Windows, Mac, Linux.
- Image analysis & **quantification**.
- Cited in **1000**+ papers per year.



Main CellProfiler window

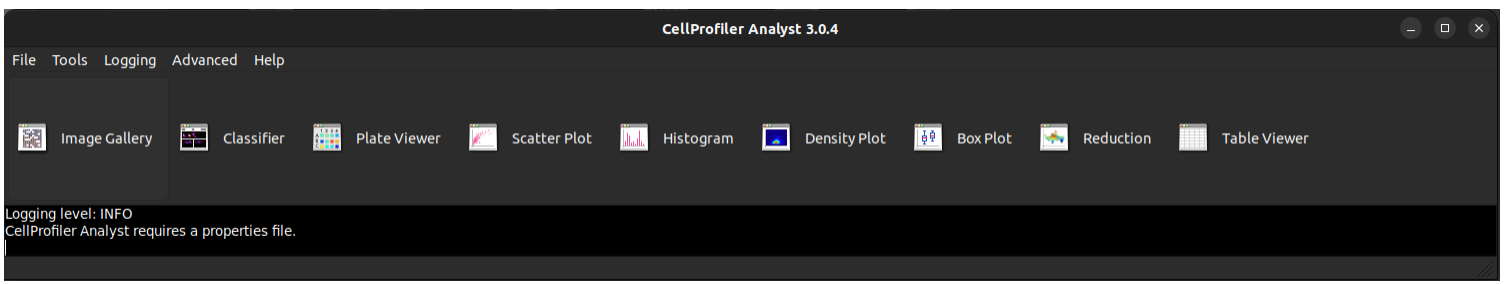


Introduction to CellProfiler Analyst

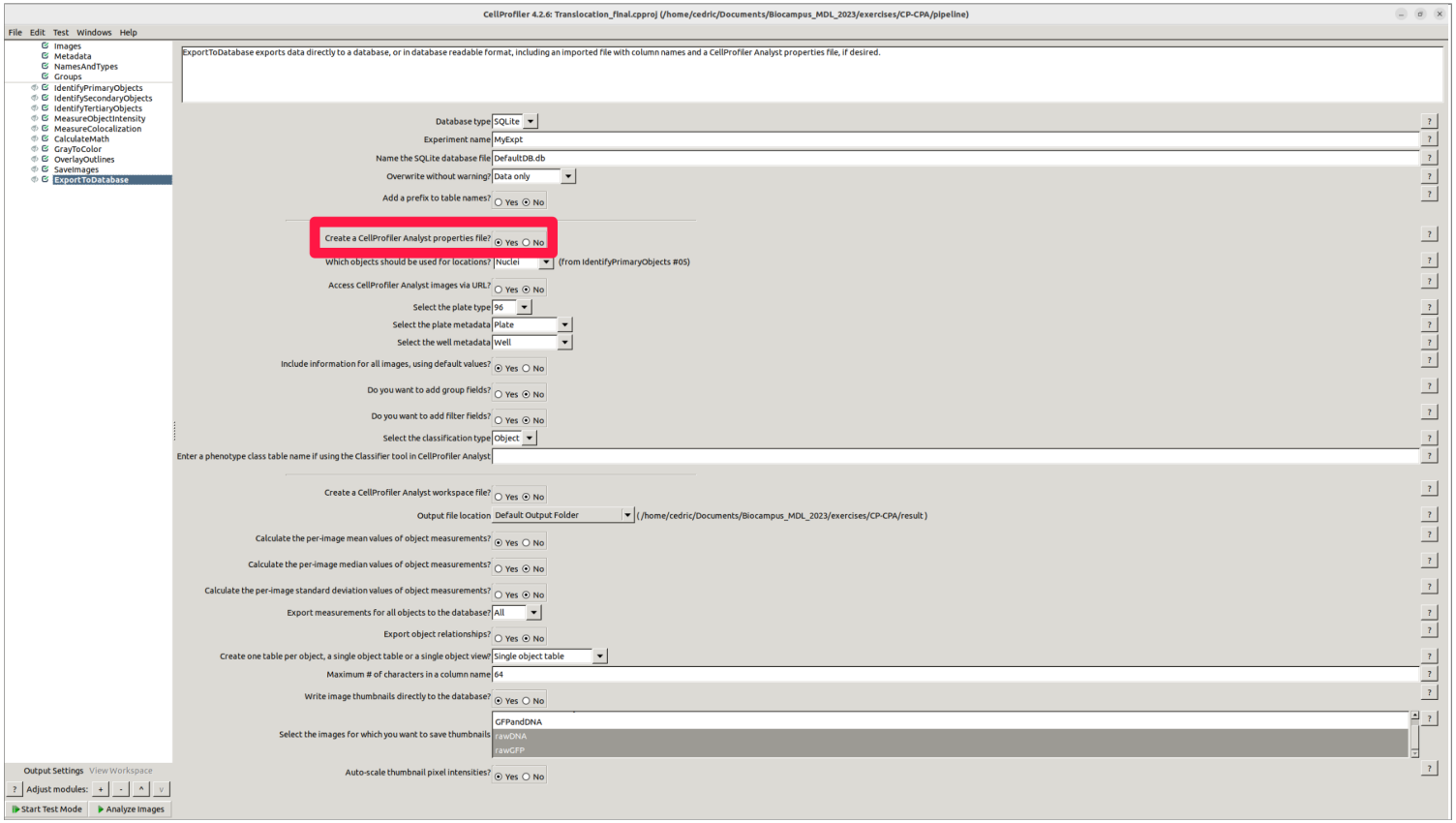
Open-source software for data exploration

- Free and open-source; Windows, Mac, Linux.
- Image-centric data analysis & machine learning.

Main CellProfiler Analyst window

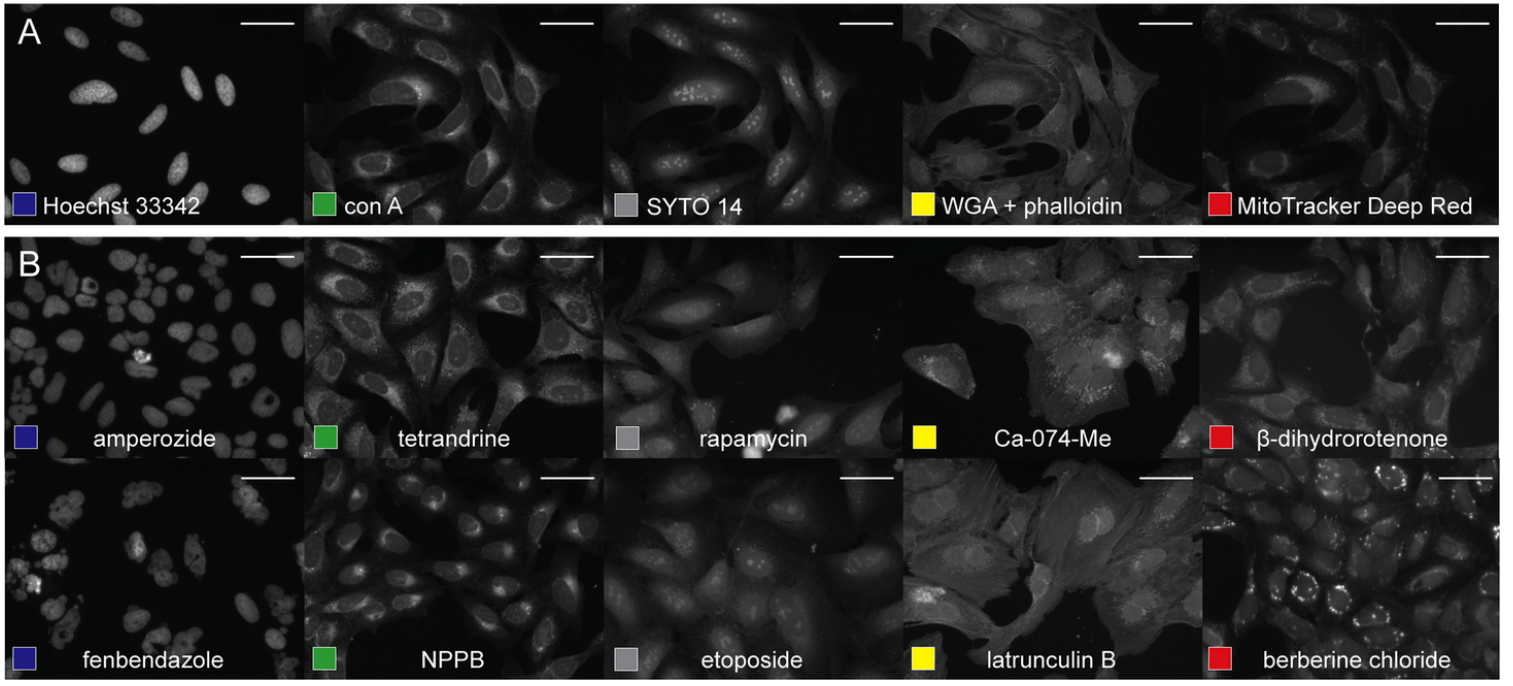


Export a properties file from CellProfiler



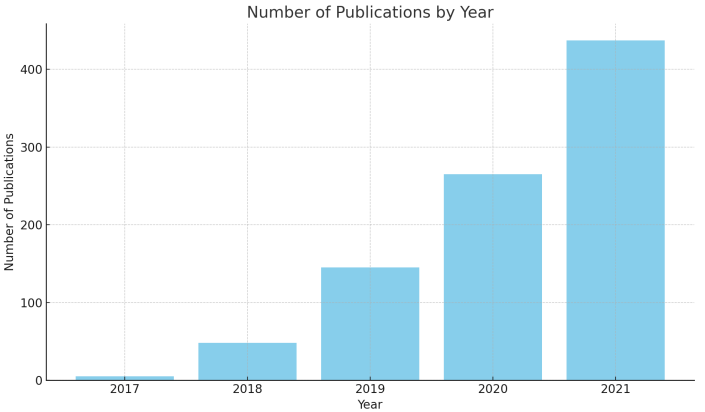
CellProfiler & CellProfiler Analyst

DEMO/Exercise: Using machine learning to perform cell painting assay

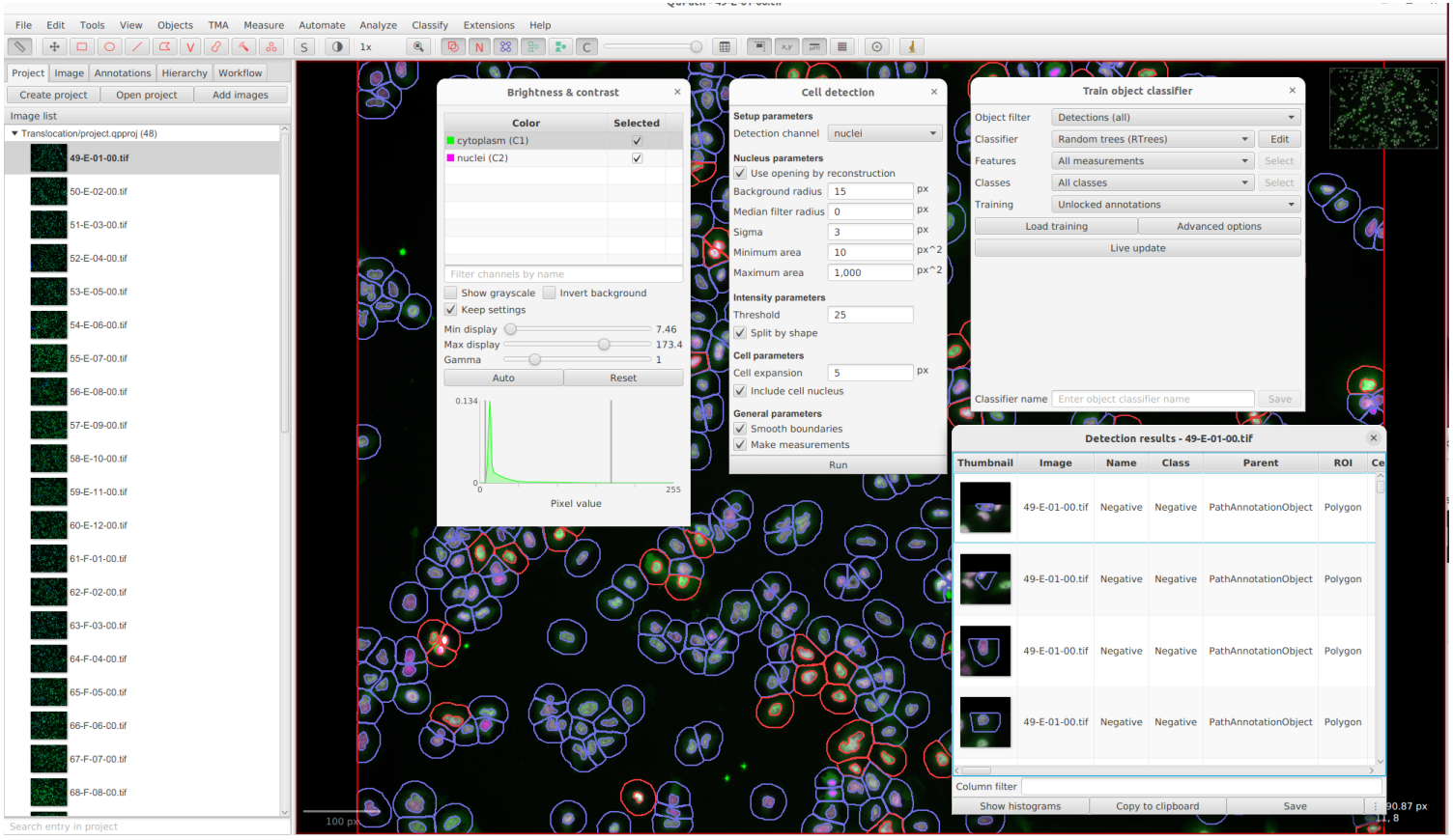


Introduction to QuPath

- Freely available and open-source; Windows, Mac, Linux.
- Dedicated to histopathology.
- Integration of machine learning.
- Growing academic recognition.



Main QuPath window



QuPath

DEMO/Exercise: Introduction

Hypothesis: treatment causes GFP translocation from cytoplasm to nucleus