Untitled

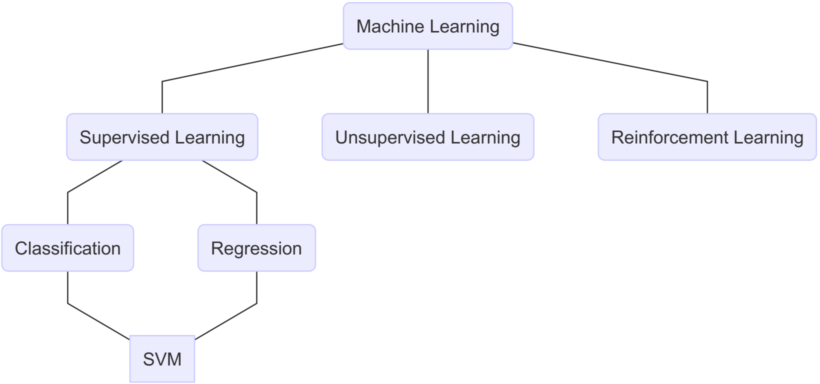
Supervised learning

Cédric Hassen-Khodja, Volker Baecker, Jean-Bernard Fiche, Francesco Pedaci

History of SVM

1. 1963: Linear classifier - Maximal Margin Classifier by Vapnik and Chervonenkis.
2. 1992: Nonlinear classification – Kernel trick by Bernhard E. Boser.
3. 1995: The Soft Margin Classifier by Corinna Cortes and Vapnik.

Types of Machine Learning



What is support vector machine ?

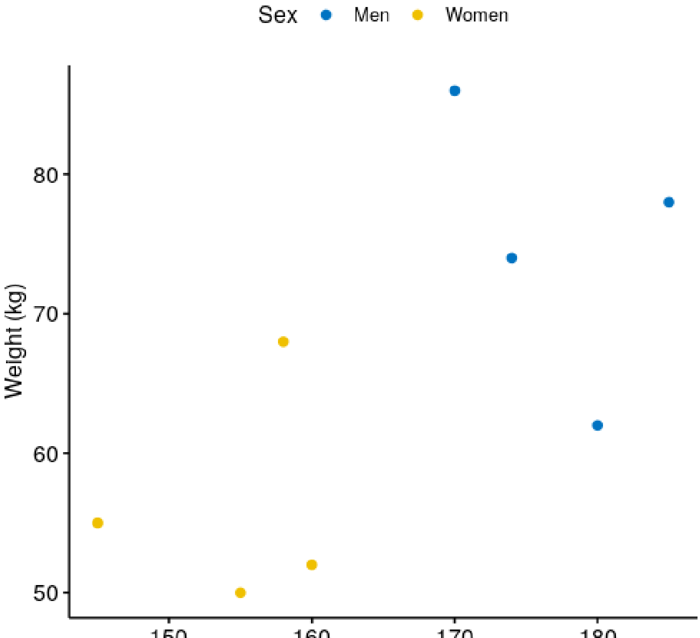
Support vector machines (SVMs) aim to find a decision hyperplane that separates data points of different classes with a maximal margin.

How does it work ?

We are given a set of people with different:

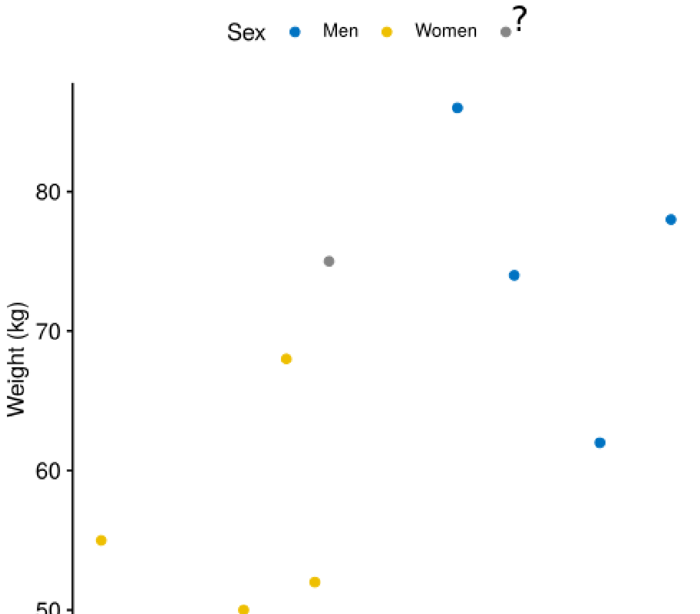
Height	Weight	Sex
145	55	Woman
155	50	Woman
160	52	Woman
158	68	Woman
174	74	Man
170	86	Man
180	62	Man
185	78	Man

How does it work ?



How does it work ?

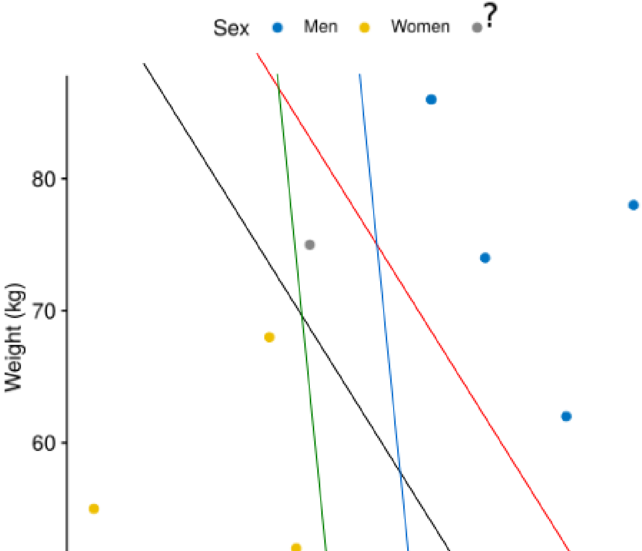
Let's add a new data point and figure out if it's a man or woman.



How does it work ?

Maximize the margin

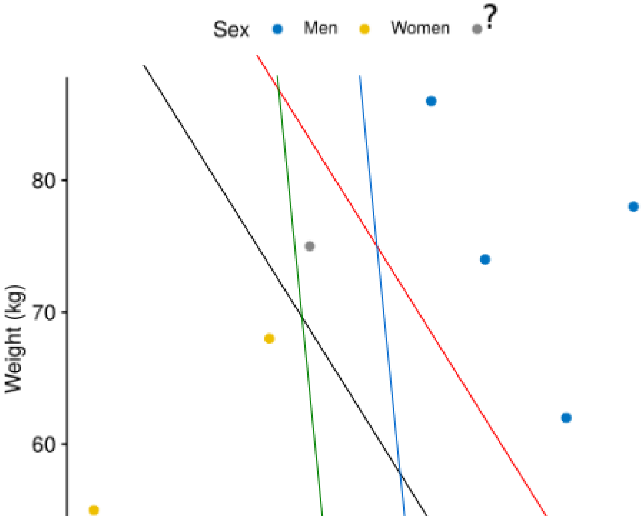
We can split our data by choosing any of these lines.



How does it work ?

Maximize the margin

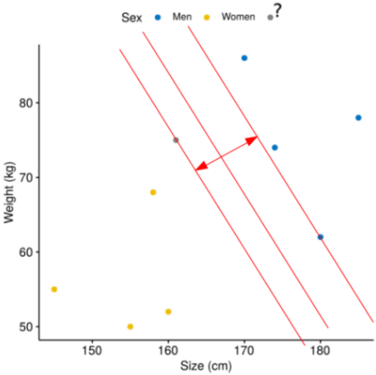
To predict the gender of a new data point we should split the data in the best possible way.

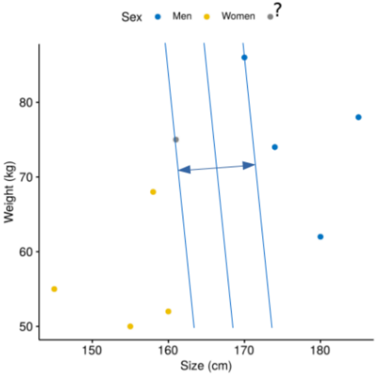


How does it work ?

Maximize the margin

This red / blue line has the maximum space that separates the two classes.

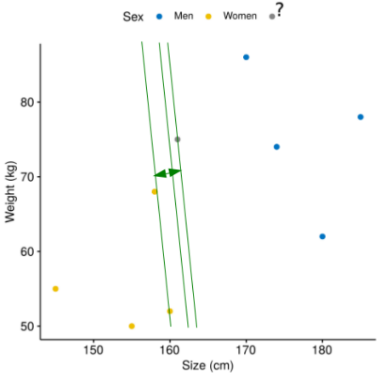


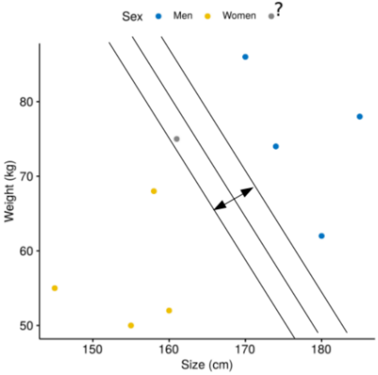


How does it work ?

Maximize the margin

While the others lines (black / green) doesn't have the maximum space that separates the two classes.

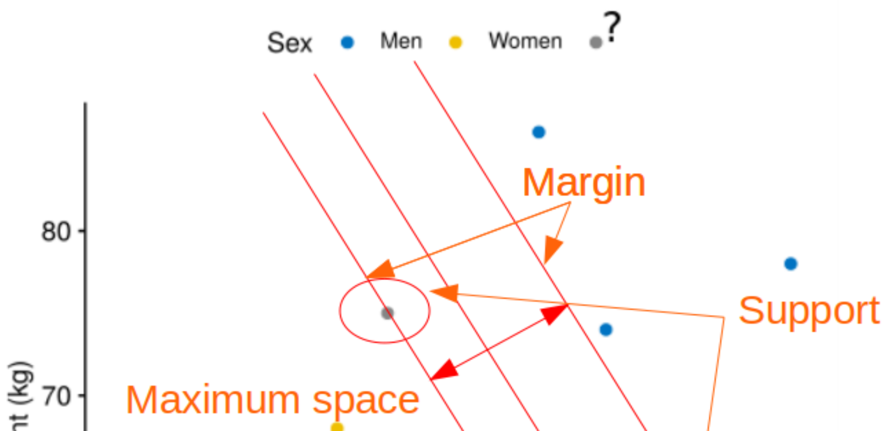




How does it work ?

Maximize the margin

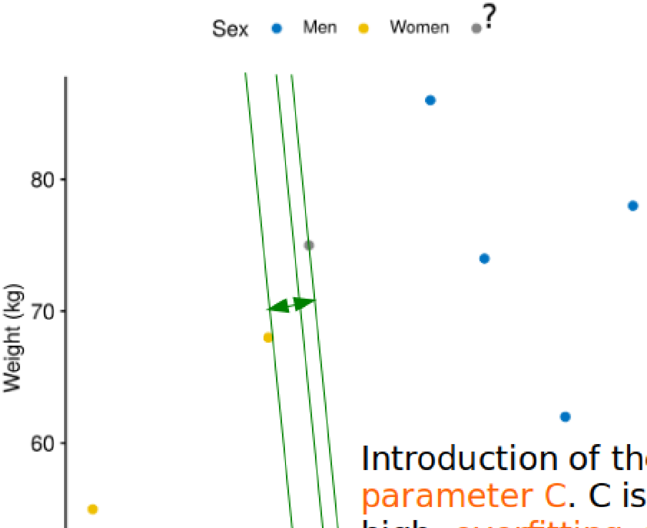
We can also say that the distance between the support and the line should be far as possible. Where support vectors are the extreme points in the datasets and *hyperplane* has the maximum distance to the support vectors of any class. Based on the distance margin we can say the new data point belongs to woman gender.



How does it work ?

Soft Margin

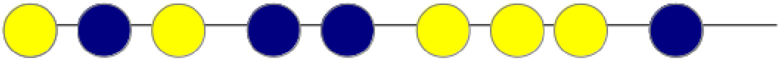
If we select a hyperplane having low margin then there is high chance of misclassification.



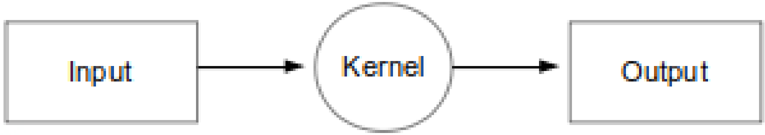
Introduction of the cost parameter C. C is too

How does it work ?

Kernel trick

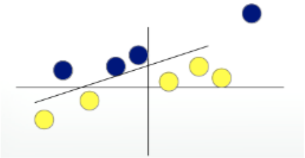


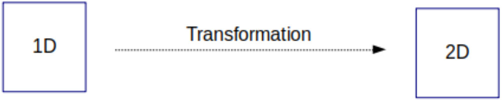
it's necessary to move away from a 1-D view of the data to a 2-D view. For the transformation we use a *kernel* function.



How does it work ?

Kernel trick

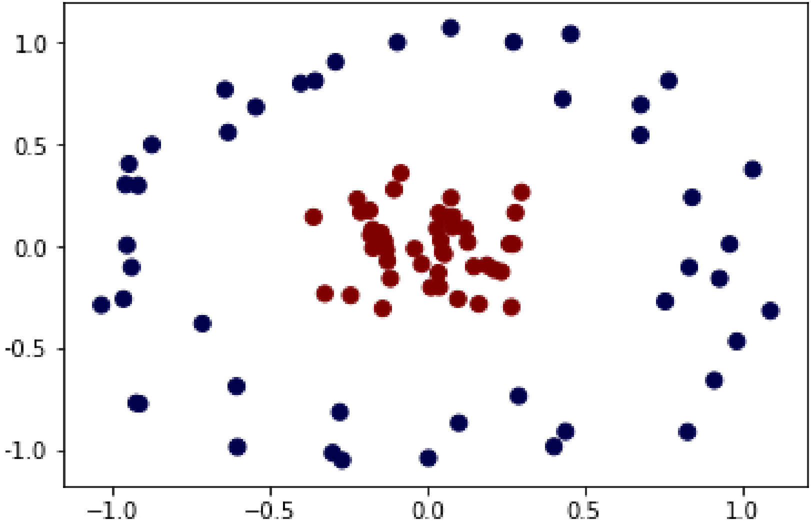




How does it work ?

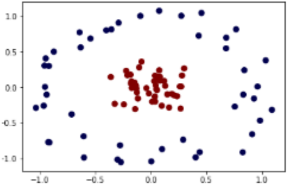
Kernel trick

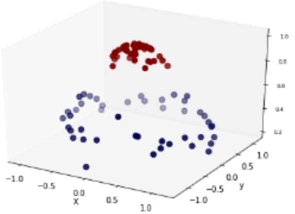
How to perform SVM for this type of dataset ?

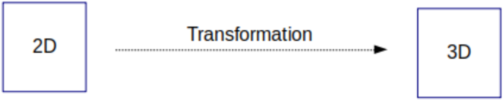


How does it work ?

Kernel trick



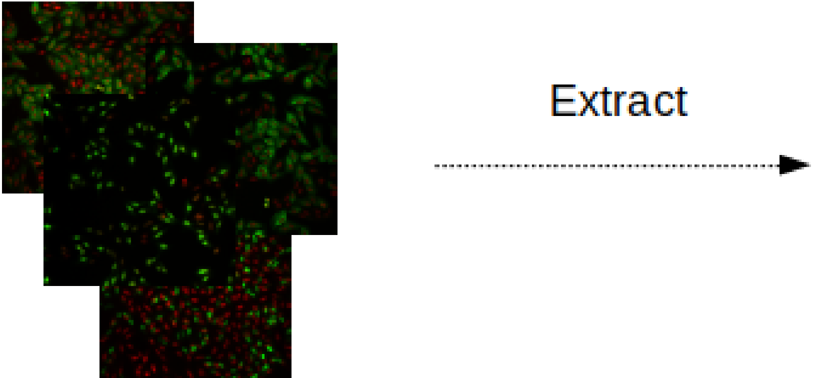




SVM in practice - Implementing biological application with Python

Use Case - Problem Statement

Estimate the lowest dose necessary to induce the cytoplasm to nucleus translocation of the FKHR-EGFP in U2OS (osteosarcoma cell line).



Use Case - Translocation Activity

Importing libraries

Entrée [ ]:

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

Importing the dataset

In [14]:

```
file = "/home/cedric/Documents/ML_FormationBC/my_table.csv"
data = pd.read_csv(file)

data.head(n = 5)
```

	Label	ImageNumber	ObjectNumber	Metadata_Well	Cells_Number_Object_Number	Cells_Children_Cytopla
0	-1	1	1	A01	1	1
1	-1	1	2	A01	2	1
2	1	1	3	A01	3	1
3	1	1	4	A01	4	1
4	1	1	5	A01	5	1

5 rows × 75 columns

Use Case - Translocation Activity

Preprocessing

Entrée [ ]:

```
X = data.drop(columns=['Label', 'Metadata_Well'])
y = data['Label']
```

Split Data

In [22]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)

print(X_train.shape, X_test.shape)
```

```
(32, 73) (8, 73)
```

Use Case - Translocation Activity

Training the Model on the training data

Scikit-Learn contains the *SVC* library, which contains built-in classes for different SVM algorithms. In the case of a simple SVM we simply set this parameter as “linear” since simple SVMs can only classify linearly separable data.
The fit method of SVC class is called to train the algorithm on the training data, which is passed as a parameter to the fit method.

In [23]:

```
from sklearn.svm import SVC

svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, y_train)
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='linear',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

Use Case - Translocation Activity

feature importance

Use Case - Translocation Activity

Prediction on the test data

In [25]:
```
y_pred = svclassifier.predict(X_test)
```

Evaluating the Model

In [41]:

```python
X_newtest = X_test[['ImageNumber', 'ObjectNumber']]
W = data[['ImageNumber', 'ObjectNumber', 'Metadata_Well']]
X_newtest = X_newtest.join(W, rsuffix="_W")
X_newtest = X_newtest[['ImageNumber', 'ObjectNumber', 'Metadata_Well']]
new_testdata = X_newtest.assign(Prediction = y_pred)
print(new_testdata)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

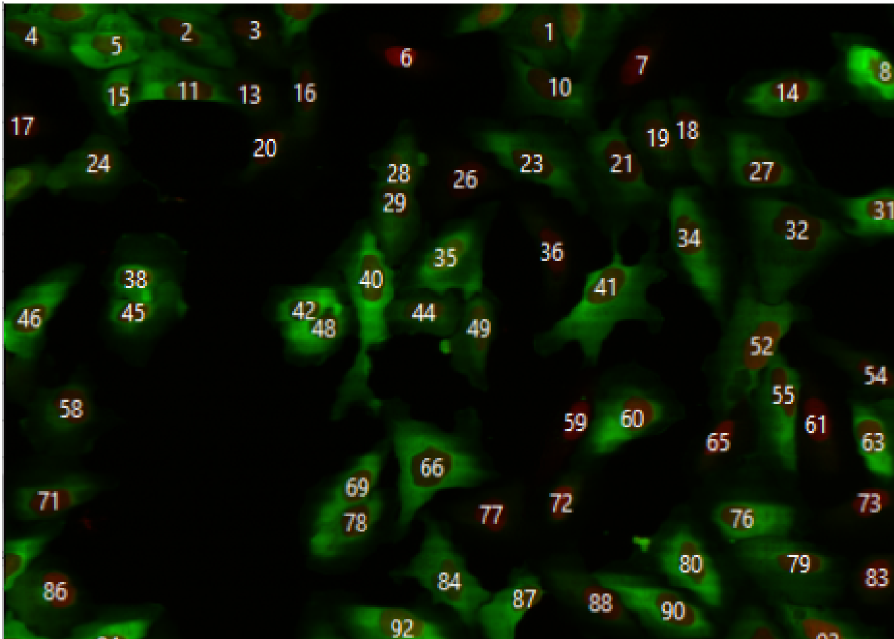
```
    ImageNumber  ObjectNumber Metadata_Well  Prediction
14            2             5          A012           1
25           10             6           E02          -1
12            2             3          A012           1
1             1             2           A01          -1
11            2             2          A012           1
26           10             7           E02          -1
10            2             1          A012           1
5             1             6           A01           1
[[3 0]
 [0 5]]
             precision    recall  f1-score   support

         -1       1.00      1.00      1.00         3
          1       1.00      1.00      1.00         5

avg / total       1.00      1.00      1.00         8
```

Use Case - Translocation Activity

Prediction on the unlabelled dataset



Use Case - Translocation Activity

Prediction on the unlabelled dataset

In [48]:

```python
file_2 = "/home/cedric/Documents/ML_FormationBC/unlabelled_dataset.csv"
unlabelled_data = pd.read_csv(file_2)
unlabelled_data.head(n = 5)
print(*unlabelled_data, sep=', ')
```

```
ImageNumber, ObjectNumber, Cells_Number_Object_Number, Cells_Children_Cytoplasm_Count, Cells_Location_Center_X, Cells_Location_Center_Y, Cells_Location_Center_Z, Cells_Parent_Nuclei, Cytoplasm_Number_Object_Number, Cytoplasm_Intensity_IntegratedIntensityEdge_GFP, Cytoplasm_Intensity_IntegratedIntensity_GFP, Cytoplasm_Intensity_LowerQuartileIntensity_GFP, Cytoplasm_Intensity_MADIntensity_GFP, Cytoplasm_Intensity_MassDisplacement_GFP, Cytoplasm_Intensity_MaxIntensityEdge_GFP, Cytoplasm_Intensity_MaxIntensity_GFP, Cytoplasm_Intensity_MeanIntensityEdge_GFP, Cytoplasm_Intensity_MeanIntensity_GFP, Cytoplasm_Intensity_MedianIntensity_GFP, Cytoplasm_Intensity_MinIntensityEdge_GFP, Cytoplasm_Intensity_MinIntensity_GFP, Cytoplasm_Intensity_StdIntensityEdge_GFP, Cytoplasm_Intensity_StdIntensity_GFP, Cytoplasm_Intensity_UpperQuartileIntensity_GFP, Cytoplasm_Location_CenterMassIntensity_X_GFP, Cytoplasm_Location_CenterMassIntensity_Y_GFP, Cytoplasm_Location_CenterMassIntensity_Z_GFP, Cytoplasm_Location_Center_X, Cytoplasm_Location_Center_Y, Cytoplasm_Location_MaxIntensity_X_GFP, Cytoplasm_Location_MaxIntensity_Y_GFP, Cytoplasm_Location_MaxIntensity_Z_GFP, Cytoplasm_Math_IntensityRatio, Cytoplasm_Parent_Cells, Cytoplasm_Parent_Nuclei, Nuclei_Number_Object_Number, Nuclei_Children_Cells_Count, Nuclei_Children_Cytoplasm_Count, Nuclei_Correlation_Correlation_GFP_DNA, Nuclei_Correlation_Costes_DNA_GFP, Nuclei_Correlation_Costes_GFP_DNA, Nuclei_Correlation_K_DNA_GFP, Nuclei_Correlation_K_GFP_DNA, Nuclei_Correlation_Manders_DNA_GFP, Nuclei_Correlation_Manders_GFP_DNA, Nuclei_Correlation_Overlap_GFP_DNA, Nuclei_Correlation_RWC_DNA_GFP, Nuclei_Correlation_RWC_GFP_DNA, Nuclei_Intensity_IntegratedIntensityEdge_GFP, Nuclei_Intensity_IntegratedIntensity_GFP, Nuclei_Intensity_LowerQuartileIntensity_GFP, Nuclei_Intensity_MADIntensity_GFP, Nuclei_Intensity_MassDisplacement_GFP, Nuclei_Intensity_MaxIntensityEdge_GFP, Nuclei_Intensity_MaxIntensity_GFP, Nuclei_Intensity_MeanIntensityEdge_GFP, Nuclei_Intensity_MeanIntensity_GFP, Nuclei_Intensity_MedianIntensity_GFP, Nuclei_Intensity_MinIntensityEdge_GFP, Nuclei_Intensity_MinIntensity_GFP, Nuclei_Intensity_StdIntensityEdge_GFP, Nuclei_Intensity_StdIntensity_GFP, Nuclei_Intensity_UpperQuartileIntensity_GFP, Nuclei_Location_CenterMassIntensity_X_GFP, Nuclei_Location_CenterMassIntensity_Y_GFP, Nuclei_Location_CenterMassIntensity_Z_GFP, Nuclei_Location_Center_X, Nuclei_Location_Center_Y, Nuclei_Location_Center_Z, Nuclei_Location_MaxIntensity_X_GFP, Nuclei_Location_MaxIntensity_Y_GFP, Nuclei_Location_MaxIntensity_Z_GFP, Nuclei_Math_IntensityRatio
```

Use Case - Translocation Activity

Prediction on the unlabelled dataset

```
In [51]:
pred = svclassifier.pr
labelled_data = unlabe
labelled_data = labell
print(labelled_data)
```

```
   ImageNumber  ObjectNumber  P
0           12             1
1           12             2
2           12             3
3           12             4
4           12             5
```

Use Case - Translocation Activity

Model Accuracy

$$Accuracy = (\frac{CountPositivesCells}{CountTotalCells}) * 100$$

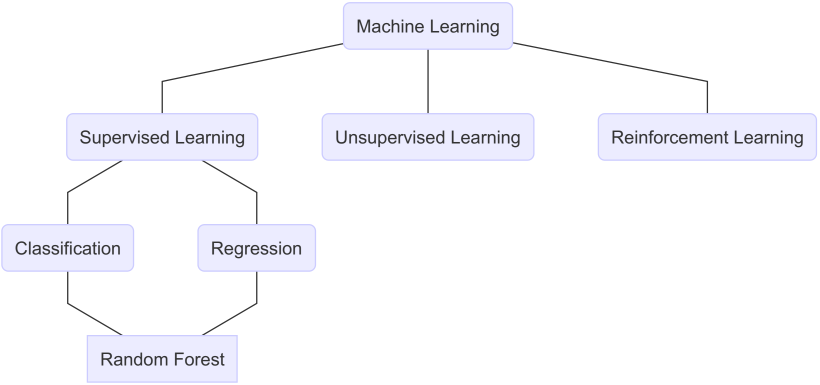
$$Accuracy = (\frac{18}{20} * 100)$$

$$Accuracy = 90\%$$

History of Random Forest

1. 1998: Ho has written a number of papers on "the random subspace" method which does a random selection of a subset of features to use to grow each tree.
2. 1997: In an important paper on written character recognition, Amit and Geman define a large number of geometric features and search over a random selection of these for the best split at each node.
3. 2001: The introduction of random forests proper was first made in a paper by Leo Breiman. This paper describes a method of building a forest of uncorrelated trees using a CART like procedure, combined with randomized node optimization and bagging.

Types of Machine Learning



Why Random Forest ?

Overfitting:
* Number of trees increase * Training time is less

Accuracy: * Run efficiently on large database

Missing data: * Accuracy when large proportion of data is missing

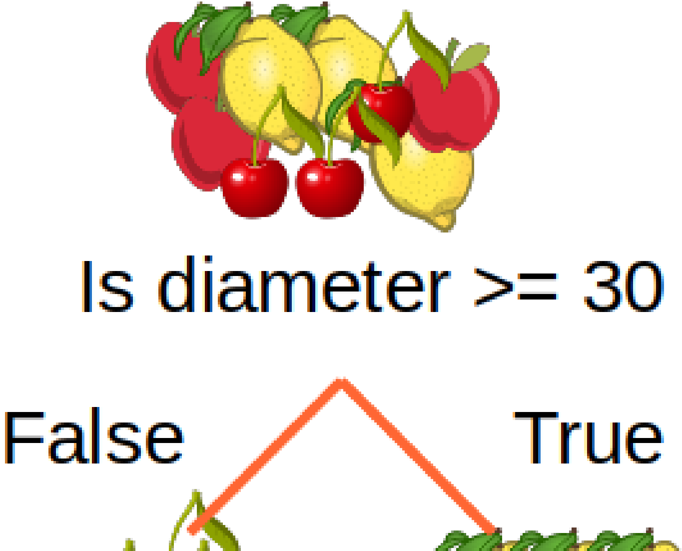
What is Random Forest ?

Random Forest creates multiple Decision Trees during training phase.

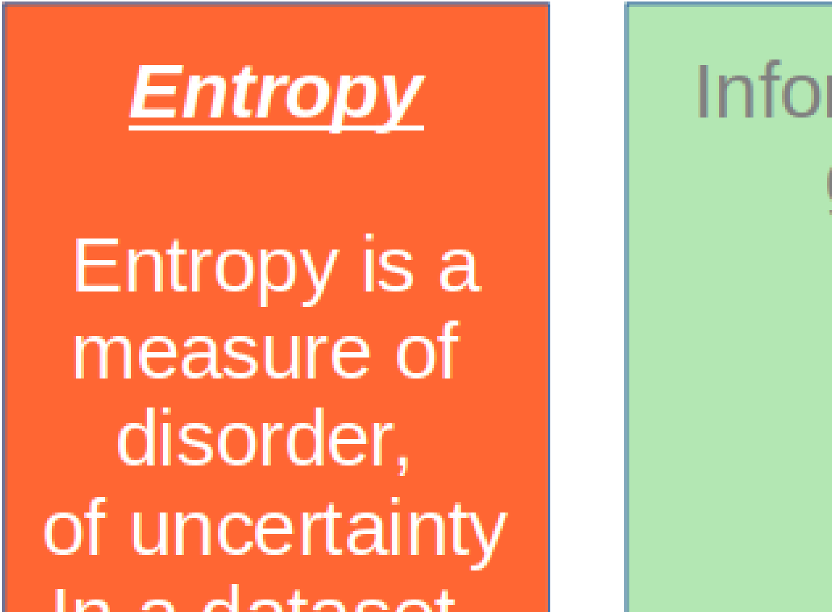
The Decision of the majority of the trees is chosen by the random forest as the final decision.

Decision Tree

Decision Tree is a tree shaped diagram. Each branch of the tree is an action and each node as a result of the decision taken.



Decision Tree - Entropy



Decision Tree - Entropy

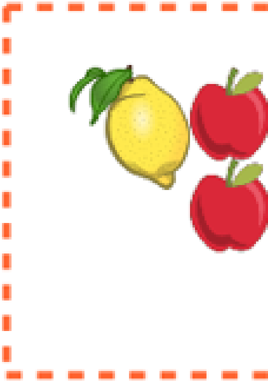
Entropy

Entropy is a measure of disorder, of uncertainty in a dataset.

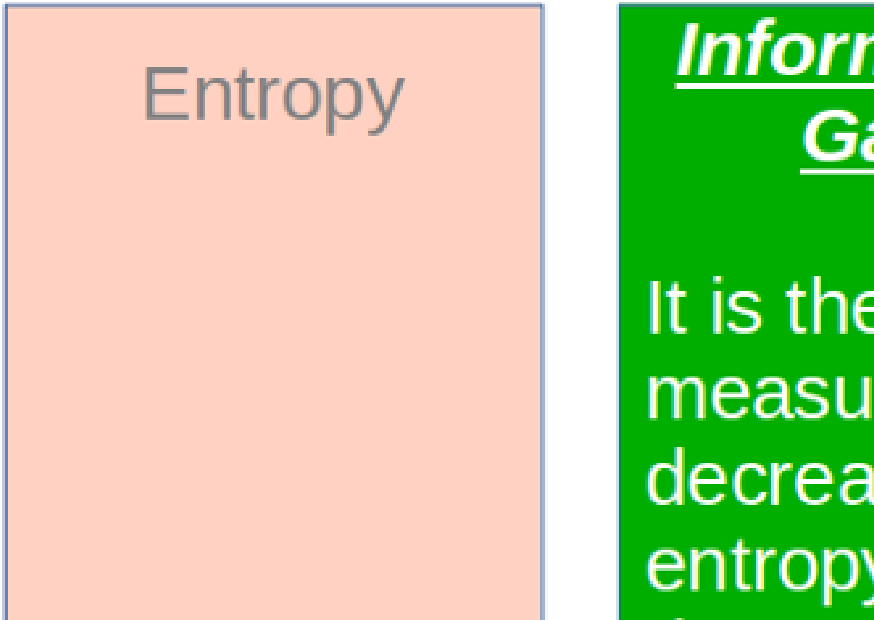
Decision Tree

Entropy

Entropy is a measure of disorder, of uncertainty



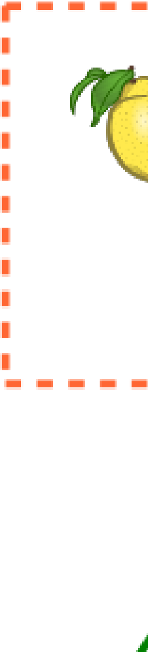
Decision Tree - Information Gain



Decision Tree - Information Gain

Information Gain

It is the measure of decrease in entropy after dataset is split



Decision Tree - Leaf node / Decision node

Entropy

Information Gain

Leaf node

caries the classification of the decision

Decision node

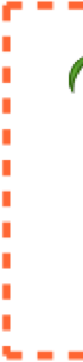
Decision node has two branch

Root node

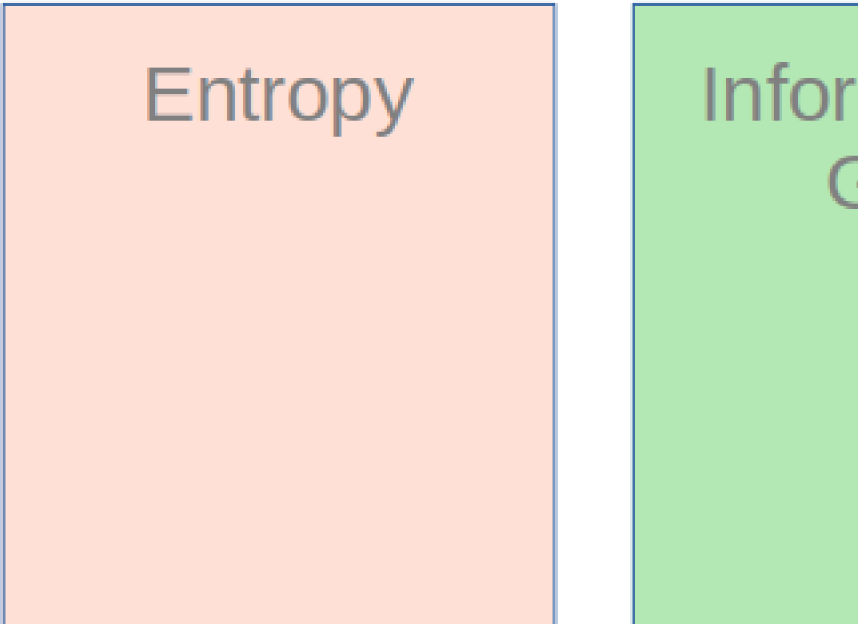
Decision Tree - Leaf node / Decision node

Leaf node

caries the classification of the decision



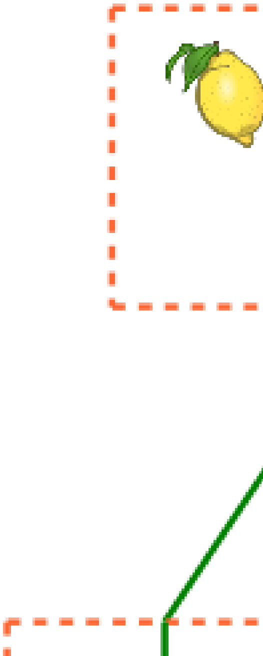
Decision Tree - Root node



Decision Tree - Root node

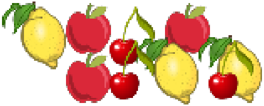
Root node

The top of the decision tree is known as the root node



How Does a decision tree work ?

Use case :
To classify the
Different types of
Fruits based on features



How Does a decision tree work ?

Use case :
To classify the
Different types of
Fruits based on features

The dataset is looking
disorder and the entropy
is high in this case

How Does a decision tree work ?

Use case :
To classify the
Different types of
Fruits based on features

The dataset is looking
disorder and the entropy
is high in this case

Training dataset

Color	*Diameter*	*Label*
Red	30	Cherry
Yellow	80	Lemon
Red	90	Apple
Red	30	Cherry
Yellow	80	Lemon
Red	90	Apple

How Does a decision tree work ?

Use case :
To classify the
Different types of
Fruits based on features

The dataset is looking
disorder and the entropy
is high in this case

How to split the data

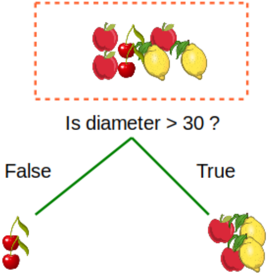
We looking for a high information gain to split the dataset

Training dataset

Color	***Diameter***	***Label***
Red	30	Cherry
Yellow	80	Lemon
Red	90	Apple
Red	30	Cherry
Yellow	80	Lemon
Red	90	Apple

How Does a decision tree work ?

We split the data



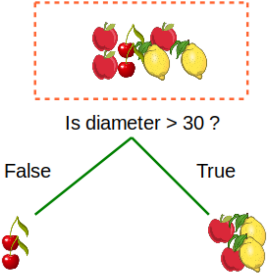
After the split, entropy has decreases considerably.

This node has an entropy equals to zero. No split is required.

How Does a decision tree work ?

We split the data

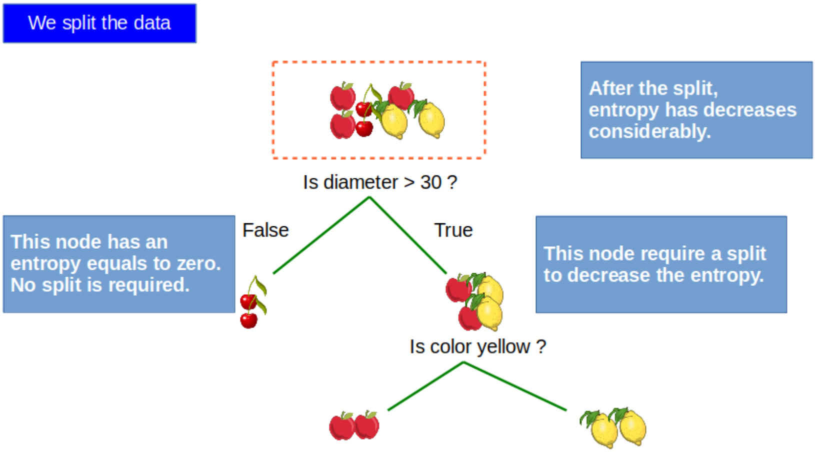
After the split, entropy has decreases considerably.



This node has an entropy equals to zero. No split is required.

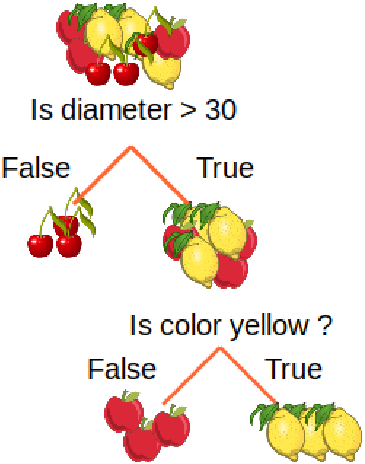
This node require a split to decrease the entropy.

How Does a decision tree work ?



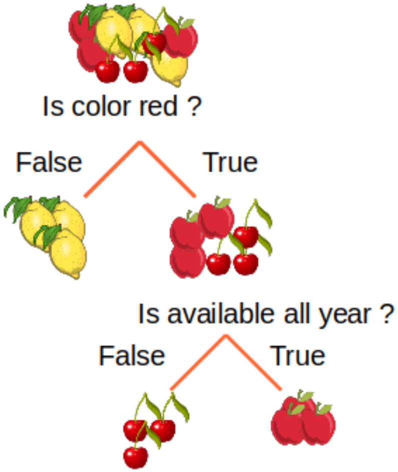
How Does a random forest work ?

Let this be tree 1



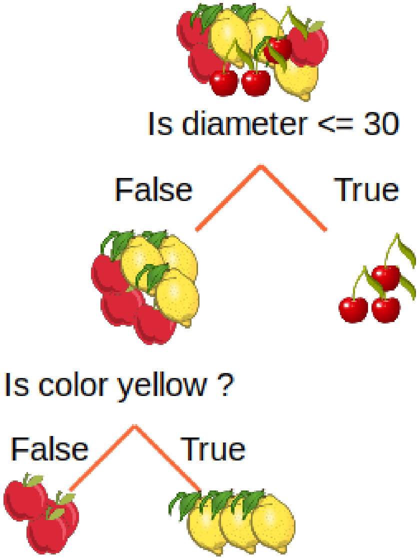
How Does a random forest work ?

Let this be tree 2

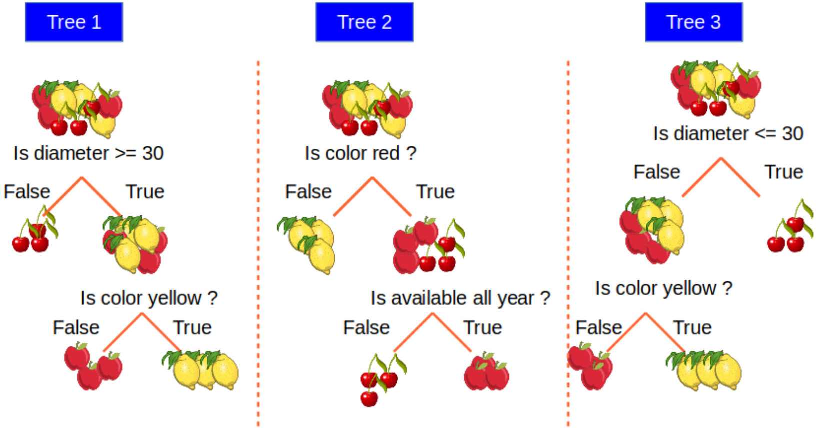


How Does a random forest work ?

Let this be tree 3



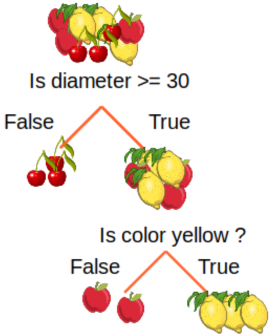
How Does a random forest work ?



How Does a random forest work ?

Now lets try to classify this fruit

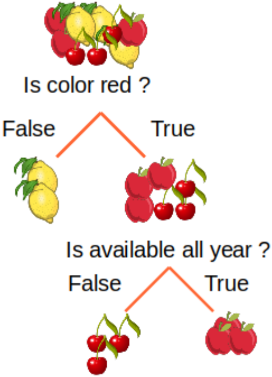
Tree 1 classify this fruit as a lemon



How Does a random forest work ?

Now lets try to classify this fruit

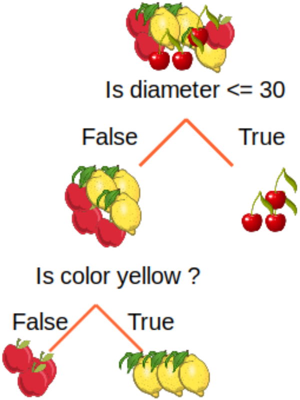
Tree 2 classify this fruit as an apple



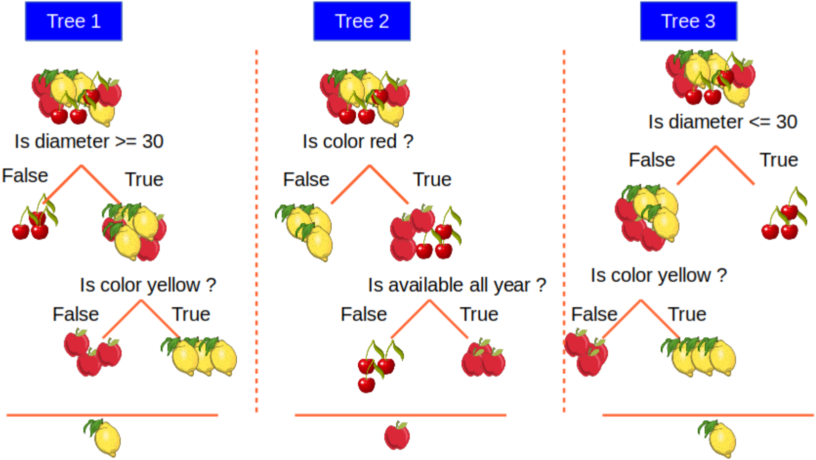
How Does a random forest work ?

Now lets try to classify this fruit

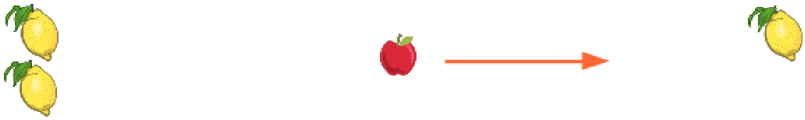
Tree 3 classify this fruit as a lemon



How Does a random forest work ?



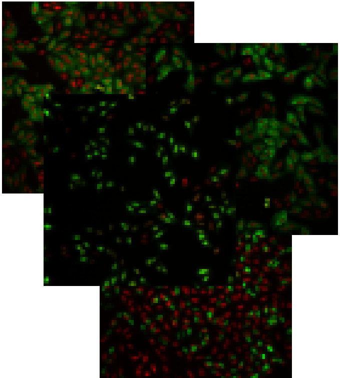
How Does a random forest work ?

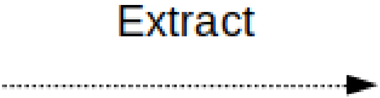


RF in practice - Implementing biological application with Python

Use Case - Problem Statement

Estimate the lowest dose necessary to induce the cytoplasm to nucleus translocation of the FKHR-EGFP in U2OS (osteosarcoma cell line).





Use Case - Translocation Activity

Preprocessing

Entrée [ ]:

```
X = data.drop(columns=['Label', 'Metadata_Well'])
y = data['Label']
```

Split Data

In [22]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)

print(X_train.shape, X_test.shape)
```

```
(32, 73) (8, 73)
```

Use Case - Translocation Activity

Training the Model on the training data

Scikit-Learn contains the *RandomForestClassifier* library. In this case we set two parameters n_jobs for parallelization task and random_state for get reproducible results. The fit method of RandomForestClassifier class is called to train the algorithm on the training data, which is passed as a parameter to the fit method.

```
Entrée [8]:
from sklearn.ensemble import RandomForestClassifier

rfclassifier = RandomForestClassifier(n_jobs = 2, random_state = 0)
rfclassifier.fit(X_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=2,
            oob_score=False, random_state=0, verbose=0, warm_start=False)
```

Use Case - Translocation Activity

feature importance

Use Case - Translocation Activity

Prediction on the test data

Entrée [9]:

```
y_pred = rfclassifier.predict(X_test)
```

Evaluating the Model

Entrée [10]:

```python
X_newtest = X_test[['ImageNumber', 'ObjectNumber']]
W = data[['ImageNumber', 'ObjectNumber', 'Metadata_Well']]
X_newtest = X_newtest.join(W, rsuffix="_W")
X_newtest = X_newtest[['ImageNumber', 'ObjectNumber', 'Metadata_Well']]
new_testdata = X_newtest.assign(Prediction = y_pred)
print(new_testdata)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

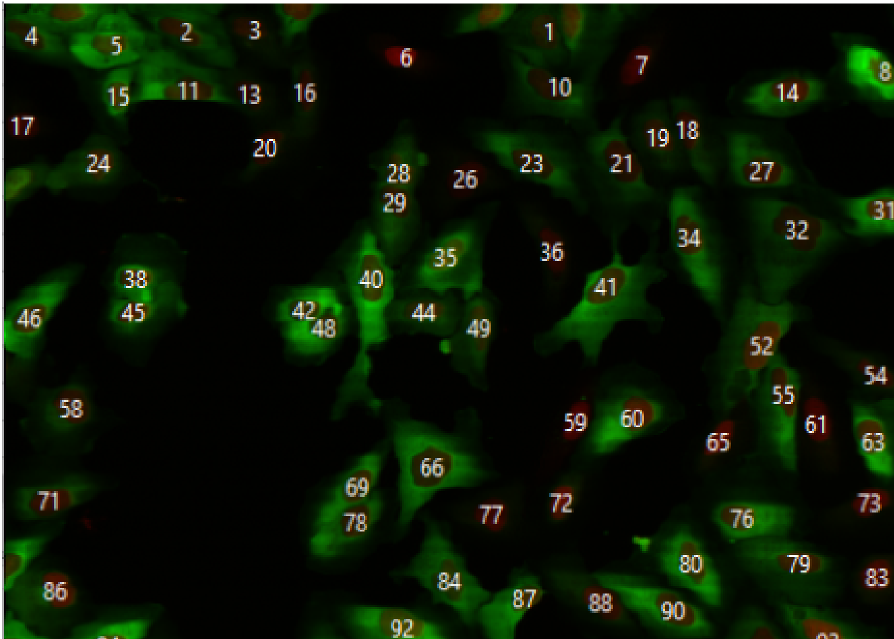
```
    ImageNumber  ObjectNumber Metadata_Well  Prediction
23           10             4           E02          -1
28           10             9           E02          -1
11            2             2          A012           1
20           10             1           E02          -1
39           18            10           E10           1
9             1            10           A01          -1
0             1             1           A01          -1
32           18             3           E10           1
[[5 0]
 [0 3]]
             precision    recall  f1-score   support

         -1       1.00      1.00      1.00         5
          1       1.00      1.00      1.00         3

avg / total       1.00      1.00      1.00         8
```

Use Case - Translocation Activity

Prediction on the unlabelled dataset



Use Case - Translocation Activity

Prediction on the unlabelled dataset

In [48]:

```python
file_2 = "/home/cedric/Documents/ML_FormationBC/unlabelled_dataset.csv"
unlabelled_data = pd.read_csv(file_2)
unlabelled_data.head(n = 5)
print(*unlabelled_data, sep=', ')
```

```
ImageNumber, ObjectNumber, Cells_Number_Object_Number, Cells_Children_Cytoplasm_Count, Cells_Location_Center_X, Cells_Loc
ation_Center_Y, Cells_Location_Center_Z, Cells_Parent_Nuclei, Cytoplasm_Number_Object_Number, Cytoplasm_Intensity_Integra
tedIntensityEdge_GFP, Cytoplasm_Intensity_IntegratedIntensity_GFP, Cytoplasm_Intensity_LowerQuartileIntensity_GFP, Cytopl
asm_Intensity_MADIntensity_GFP, Cytoplasm_Intensity_MassDisplacement_GFP, Cytoplasm_Intensity_MaxIntensityEdge_GFP, Cytop
lasm_Intensity_MaxIntensity_GFP, Cytoplasm_Intensity_MeanIntensityEdge_GFP, Cytoplasm_Intensity_MeanIntensity_GFP, Cytopl
asm_Intensity_MedianIntensity_GFP, Cytoplasm_Intensity_MinIntensityEdge_GFP, Cytoplasm_Intensity_MinIntensity_GFP, Cytopl
asm_Intensity_StdIntensityEdge_GFP, Cytoplasm_Intensity_StdIntensity_GFP, Cytoplasm_Intensity_UpperQuartileIntensity_GFP,
Cytoplasm_Location_CenterMassIntensity_X_GFP, Cytoplasm_Location_CenterMassIntensity_Y_GFP, Cytoplasm_Location_CenterMass
Intensity_Z_GFP, Cytoplasm_Location_Center_X, Cytoplasm_Location_Center_Y, Cytoplasm_Location_MaxIntensity_X_GFP, Cytopla
sm_Location_MaxIntensity_Y_GFP, Cytoplasm_Location_MaxIntensity_Z_GFP, Cytoplasm_Math_IntensityRatio, Cytoplasm_Parent_Ce
lls, Cytoplasm_Parent_Nuclei, Nuclei_Number_Object_Number, Nuclei_Children_Cells_Count, Nuclei_Children_Cytoplasm_Count,
Nuclei_Correlation_Correlation_GFP_DNA, Nuclei_Correlation_Costes_DNA_GFP, Nuclei_Correlation_Costes_GFP_DNA, Nuclei_Corr
elation_K_DNA_GFP, Nuclei_Correlation_K_GFP_DNA, Nuclei_Correlation_Manders_DNA_GFP, Nuclei_Correlation_Manders_GFP_DNA,
Nuclei_Correlation_Overlap_GFP_DNA, Nuclei_Correlation_RWC_DNA_GFP, Nuclei_Correlation_RWC_GFP_DNA, Nuclei_Intensity_Inte
gratedIntensityEdge_GFP, Nuclei_Intensity_IntegratedIntensity_GFP, Nuclei_Intensity_LowerQuartileIntensity_GFP, Nuclei_In
tensity_MADIntensity_GFP, Nuclei_Intensity_MassDisplacement_GFP, Nuclei_Intensity_MaxIntensityEdge_GFP, Nuclei_Intensity_
MaxIntensity_GFP, Nuclei_Intensity_MeanIntensityEdge_GFP, Nuclei_Intensity_MeanIntensity_GFP, Nuclei_Intensity_MedianInte
nsity_GFP, Nuclei_Intensity_MinIntensityEdge_GFP, Nuclei_Intensity_MinIntensity_GFP, Nuclei_Intensity_StdIntensityEdge_GF
P, Nuclei_Intensity_StdIntensity_GFP, Nuclei_Intensity_UpperQuartileIntensity_GFP, Nuclei_Location_CenterMassIntensity_X_
GFP, Nuclei_Location_CenterMassIntensity_Y_GFP, Nuclei_Location_CenterMassIntensity_Z_GFP, Nuclei_Location_Center_X, Nucl
ei_Location_Center_Y, Nuclei_Location_Center_Z, Nuclei_Location_MaxIntensity_X_GFP, Nuclei_Location_MaxIntensity_Y_GFP, N
uclei_Location_MaxIntensity_Z_GFP, Nuclei_Math_IntensityRatio
```

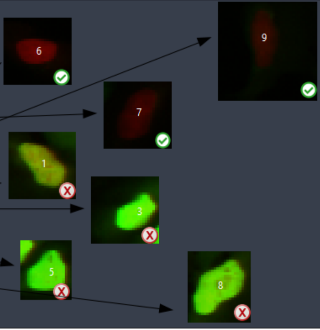
Use Case - Translocation Activity

Prediction on the unlabelled dataset

Entrée [12]:

```
pred = rfclassifier.predict(unlabelled_data)
labelled_data = unlabelled_data.assign(Prediction = pred)
labelled_data = labelled_data[['ImageNumber', 'ObjectNumber', 'Prediction']]
print(labelled_data)
```

	ImageNumber	ObjectNumber	Prediction
0	12	1	-1
1	12	2	-1
2	12	3	-1
3	12	4	-1
4	12	5	-1
5	12	6	1
6	12	7	1
7	12	8	-1
8	12	9	1
9	12	10	-1
10	16	1	-1
11	16	2	1
12	16	3	1
13	16	4	1
14	16	5	-1
15	16	6	1
16	16	7	1
17	16	8	-1
18	16	9	1
19	16	10	1



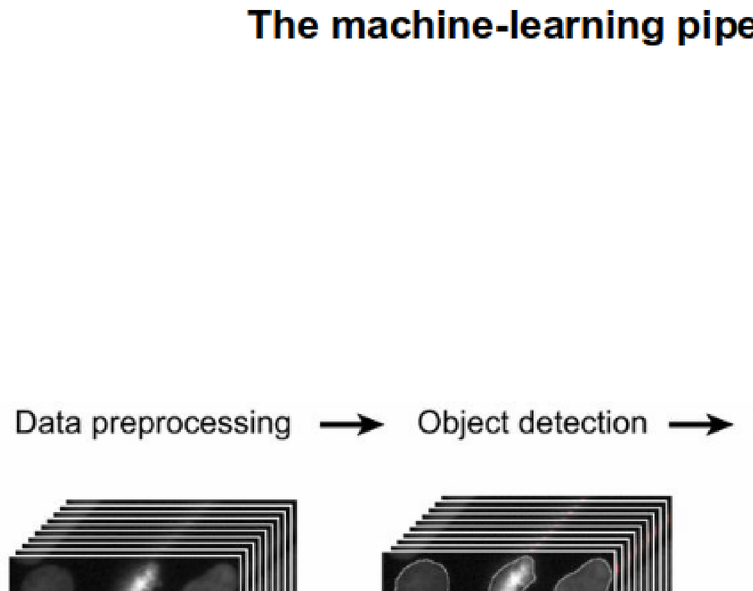
Use Case - Translocation Activity

Model Accuracy

$$Accuracy = (\frac{CountPositivesCells}{CountTotalCells}) * 100$$

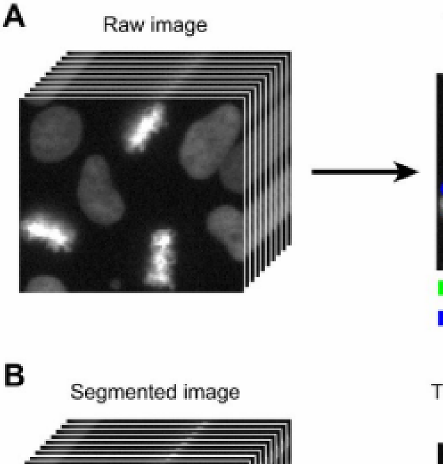
$$Accuracy = (\frac{16}{20} * 100)$$

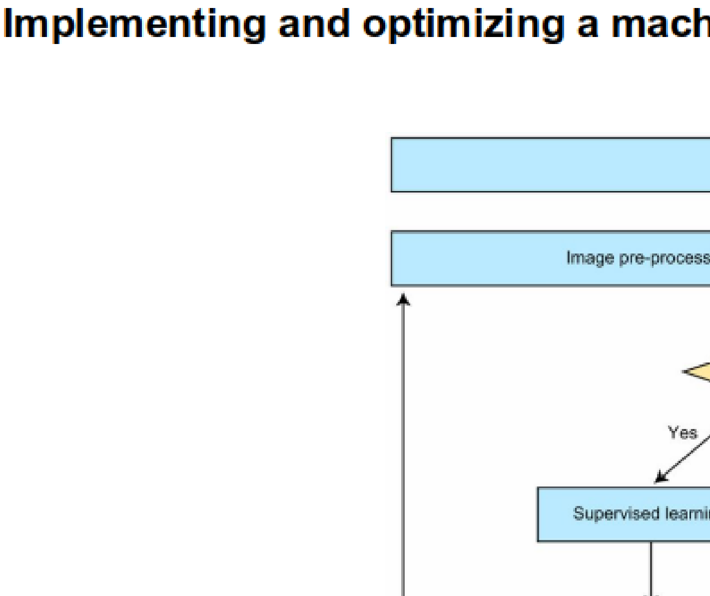
$$Accuracy = 80\%$$


The machine-learning pipe
Data preprocessing
Object detection

Machine Learning in Bioimage

Image classification by supervised machi




Implementing and optimizing a mach
Image pre-process
Yes
Supervised learni