Mahidol University
Faculty of Medicine Ramathibodi Hospital
Department of Clinical Epidemiology and Biostatistics

Machine Learning in Health Care and its applications

RADI608: Data Mining and Machine Learning
RADI602: Data Mining and Knowledge Discovery

Lect. Anuchate Pattanateepapon D.Eng.

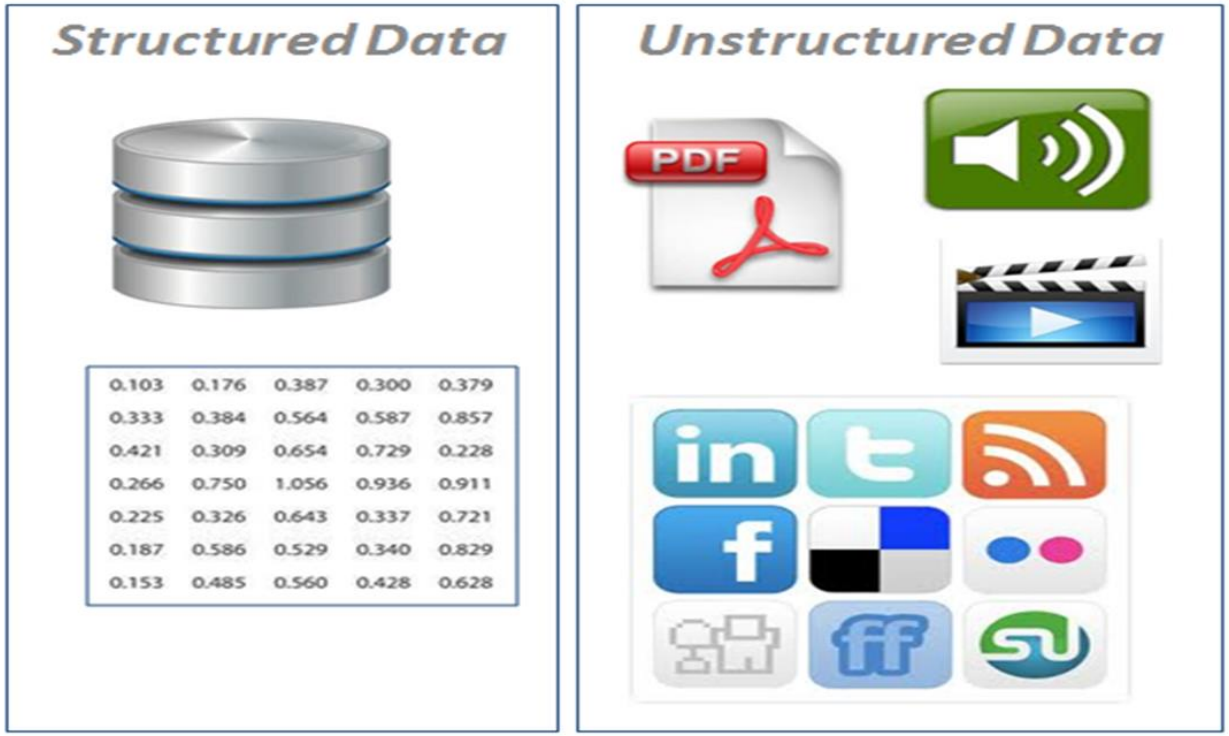
Section of Data Science for Healthcare

Department of Clinical Epidemiology and Biostatistics

Mahidol University Faculty of Medicine Ramathibodi Hospital

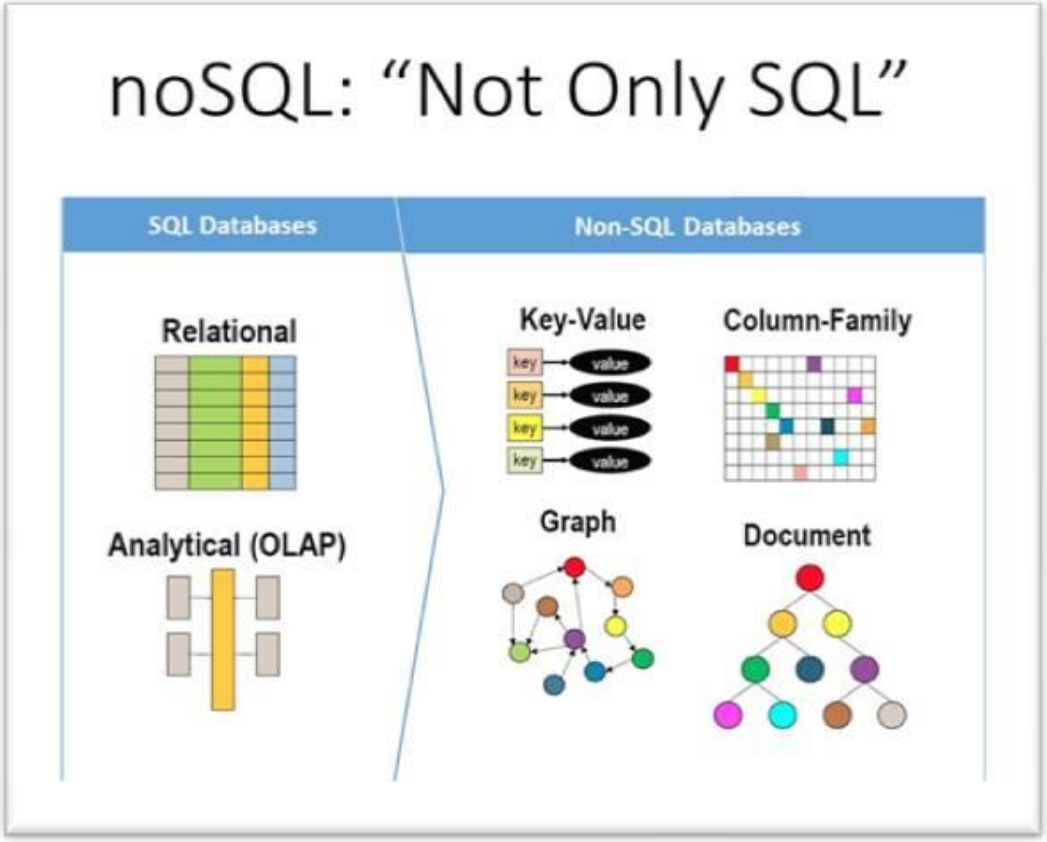
Database designed in health care

Health care data could be a structured-data or an unstructured-data



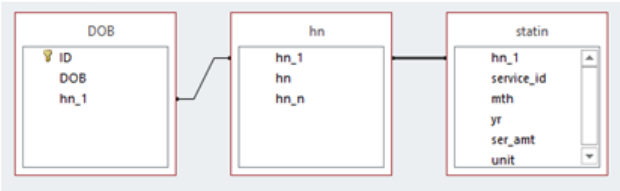
http://www.infosysblogs.com/thought-floor/2012/07/butterfly_effect_analytics_and.html

SQL(structured query language) and noSQL



https://kvaes.wordpress.com/2015/01/21/database-variants-explained-sql-or-nosql-is-that-really-the-question/

Data in health care



Structured Data

Text Data

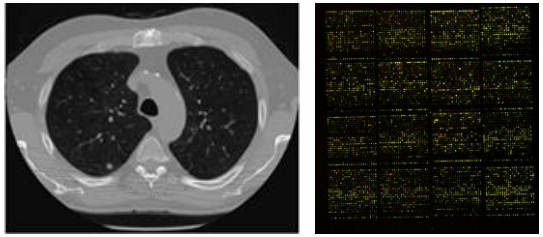
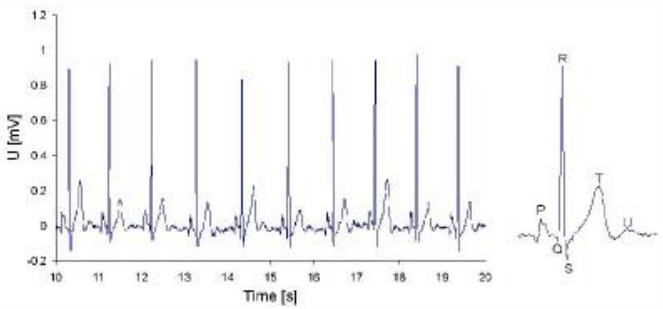


Image Data



Signal Data

Structured Data: Relational Database

***Primary key:* unique**

Person

	PID	SEX	BIRTH	PostalCode
2	559	2	19480307	10100
3	1233	1	19490313	73000
4	2330	2	19580520	73000
5	2355	2	19600506	60000
6	2968	2	19440312	73000
7	3031	1	19740305	80000
8	3075	1	19430313	52000
9	3732	2	19350205	40000
10	4140	2	19740612	80000
11	4675	2	19570311	40000
12	5272	2	19610424	52000
13	5431	2	19440317	80000
14	5754	2	19500318	10100
15	6540	1	19570316	10100

Attribute	Description	Data type	Example
PID	Patient Identification	Variable Character	559
SEX	SEX (1 = male, 2 = female)	Variable Character	1
BIRTH	Date of birth (YYYYMMDD)	Date	19480307
PostalCode	Postal Code or Post Code or Zip Code	Variable Character	10100

Diagnosis

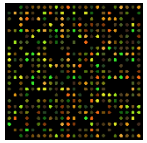
	PID	DATE_SERV	DIAGTYPE	DIAGCODE
2	559	20141211	2	E789
3	559	20141211	1	M170
4	1233	20141224	1	Z094
5	2330	20141208	1	L739
6	2330	20141208	2	Z133
7	2330	20141222	1	J029
8	2330	20141222	2	Z133
9	2968	20141203	1	R300

Attribute	Description	Data type	Example
PID	Patient Identification	Variable Character	559
DATE_SERV	Date of services (YYYYMMDD)	Date	20141211
DIAGTYPE	Diagnosis Type	Variable Character	2
DIACODE	Diagnosis Code	Variable Character	E789

***Foreign key:* referencing key**

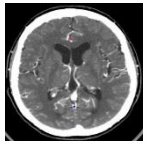
Unstructured data in health care

- gene-expression microarray image



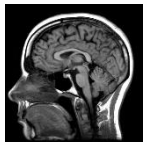
https://upload.wikimedia.org/wikipedia/commons/thumb/2/2a/DNA_microarray.svg/2000px-DNA_microarray.svg.png

- computed tomography (CT)



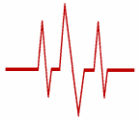
https://www.cedars-sinai.edu/Patients/Programs-and-Services/Imaging-Center/For-Physicians/Neuroradiology/Images/CT-Brain-9541.jpg

- magnetic resonance imaging (MRI)



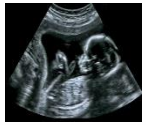
https://writersforensicsblog.files.wordpress.com/2015/07/mri-head1.jpg?w=645

- electrocardiogram (EKG)



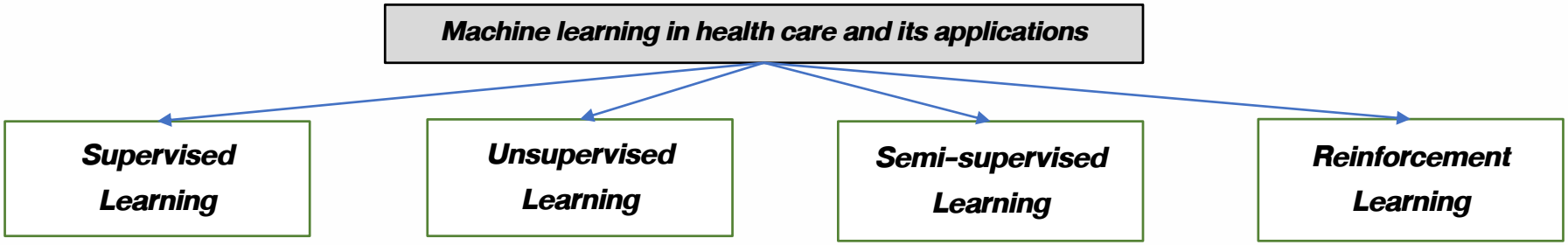
http://i6.glitter-graphics.org/pub/928/928156xf6i3uv9tv.gif

- ultrasound

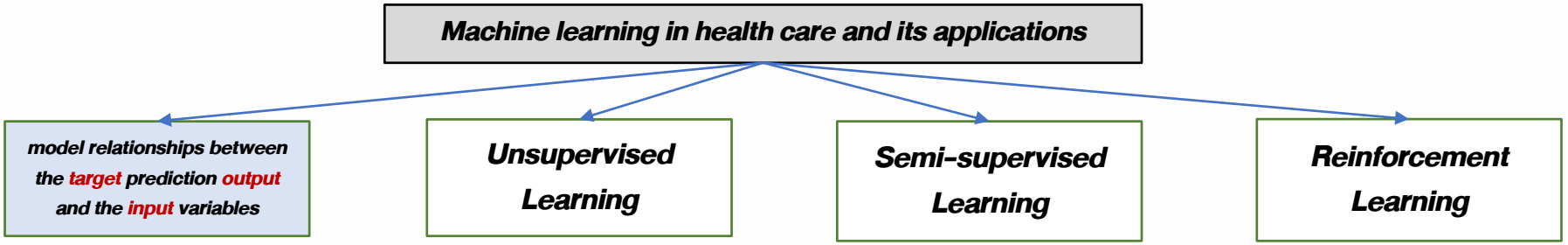


https://media.deseretdigital.com/file/1c416c5954.jpg?crop=top_0~left_0~width_2407~height_1742&resize=width_630~height_456&c=2&a=bb07cb86

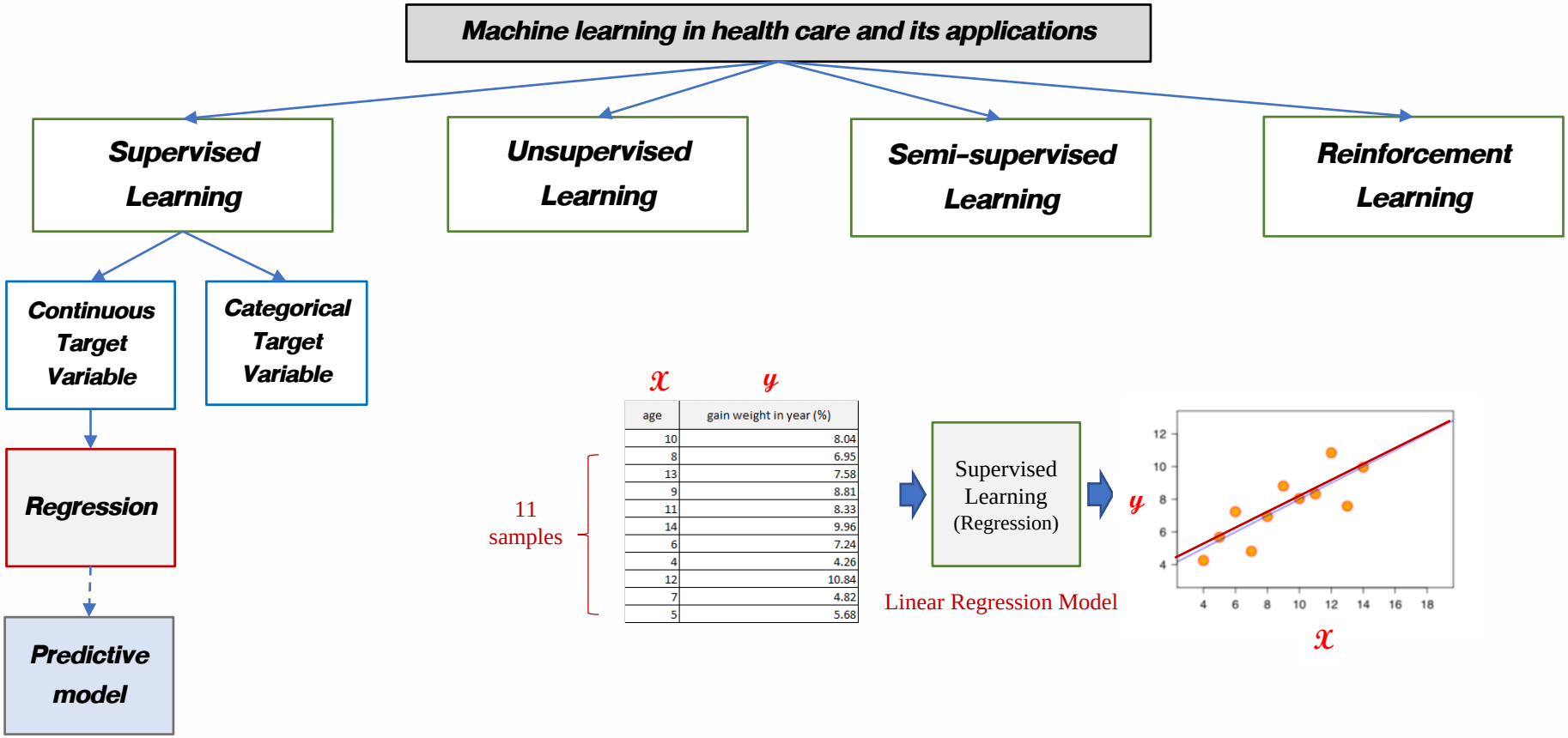
Machine learning in health care and its applications



Machine learning in health care and its applications

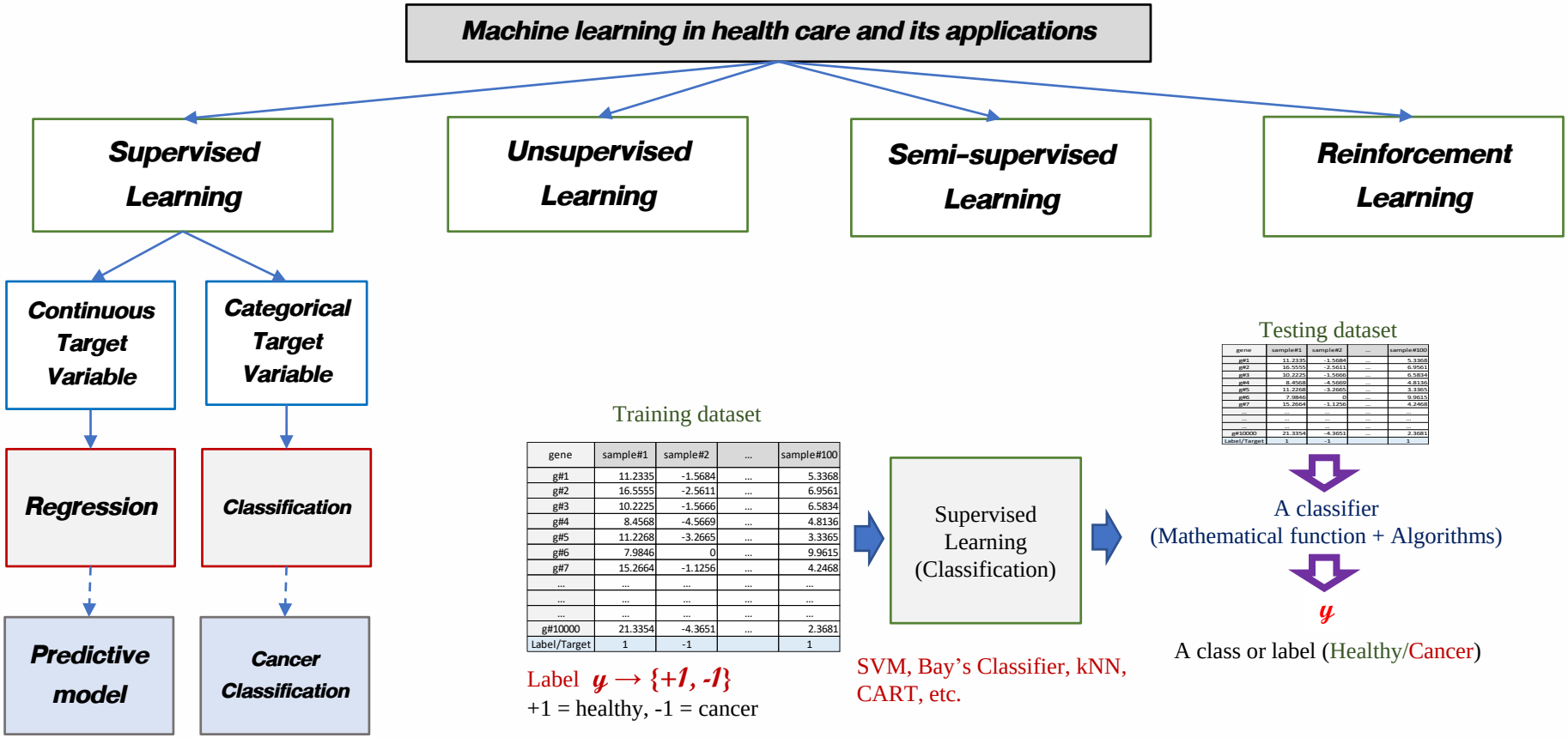


Machine learning in health care and its applications

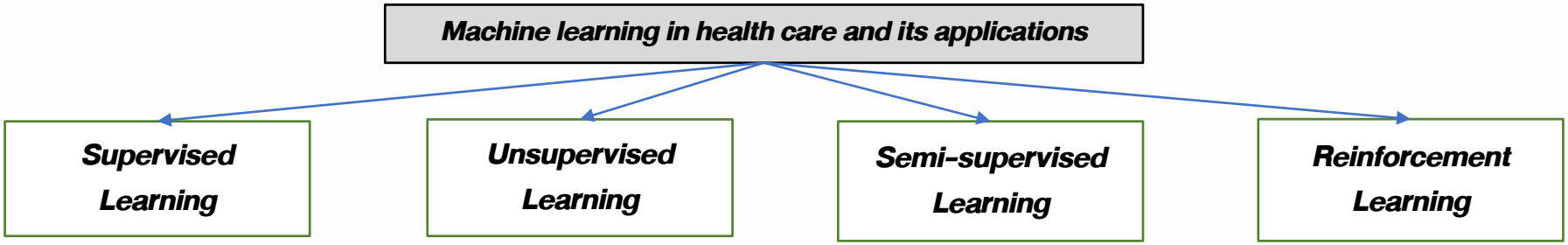


age	gain weight in year (%)
10	8.04
8	6.95
13	7.58
9	8.81
11	8.33
14	9.96
6	7.24
4	4.26
12	10.84
7	4.82
5	5.68

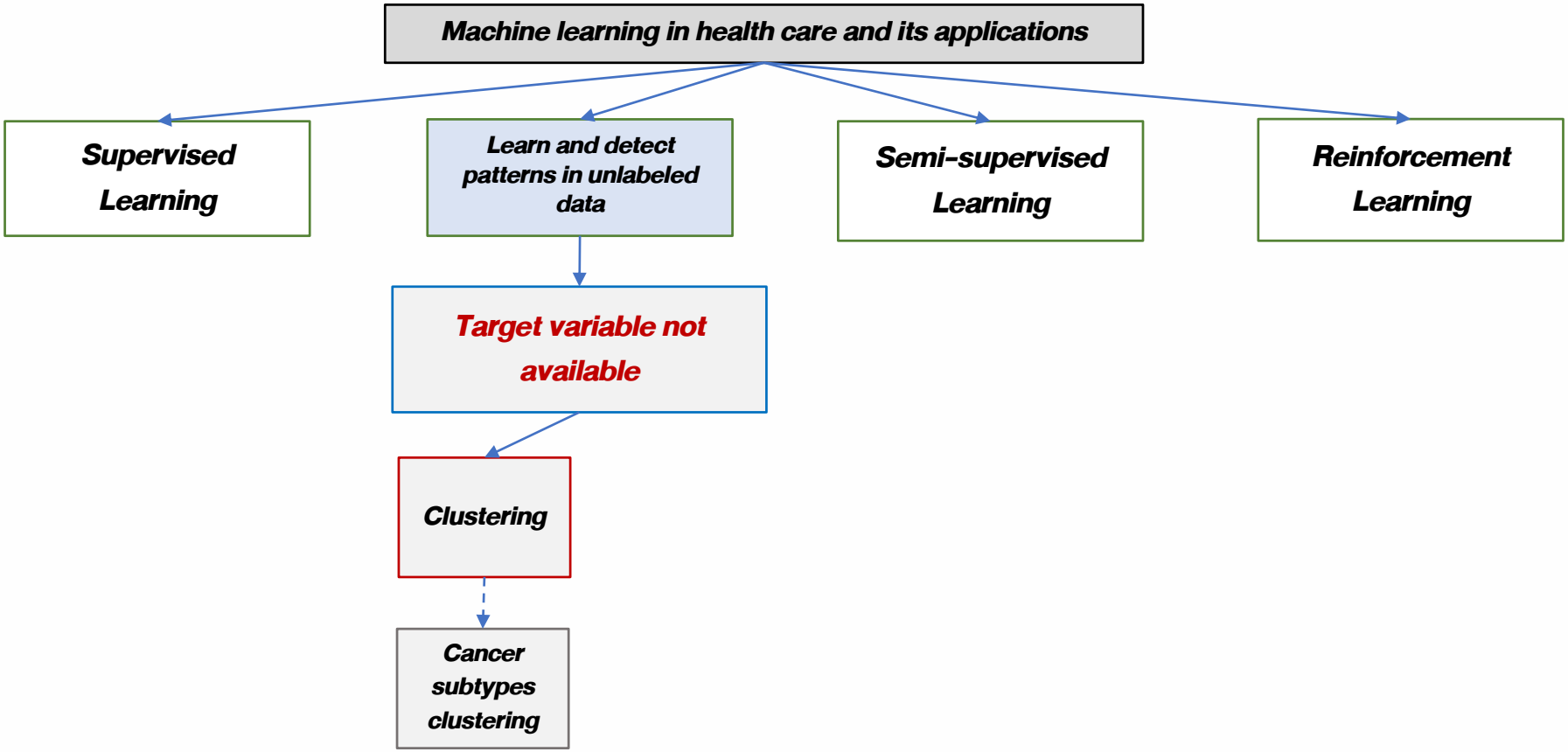
Machine learning in health care and its applications



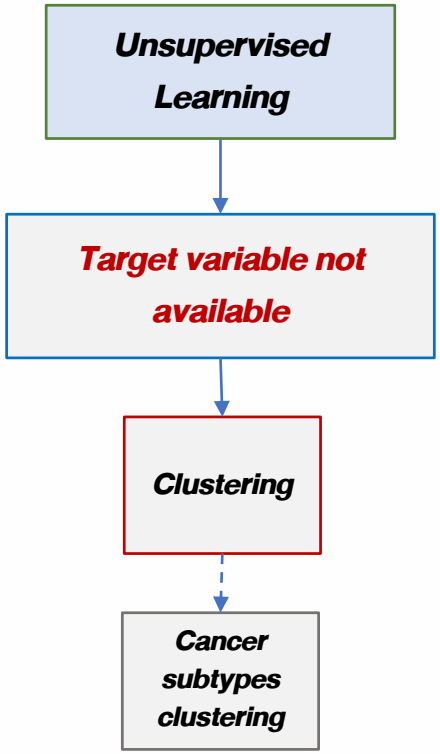
Machine learning in health care and its applications



Machine learning in health care and its applications



Machine learning in health care and its applications



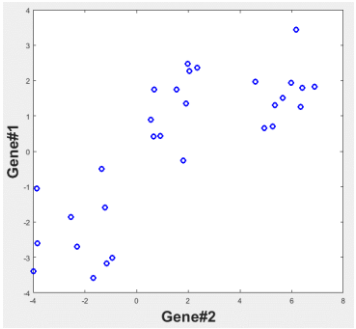
Training dataset

gene	sample#1	sample#2	sample#3	...	sample#30
g#1	11.2335	-1.5684	-10.5692	...	5.3368
g#2	16.5555	-2.5611	-15.3368	...	6.9561
Label/Target	uknown	uknown	uknown	...	uknown

Unsupervised Learning (Clustering)

K means, EM

2-Dimensions plots



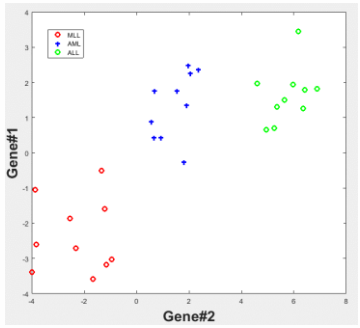
Testing dataset

gene	sample#1	sample#2	sample#3	...	sample#10
g#1	11.2335	-1.5684	-10.5692	...	5.3368
g#2	16.5555	-2.5611	-15.3368	...	6.9561
Label/Target	ALL	AML	MLL	...	ALL

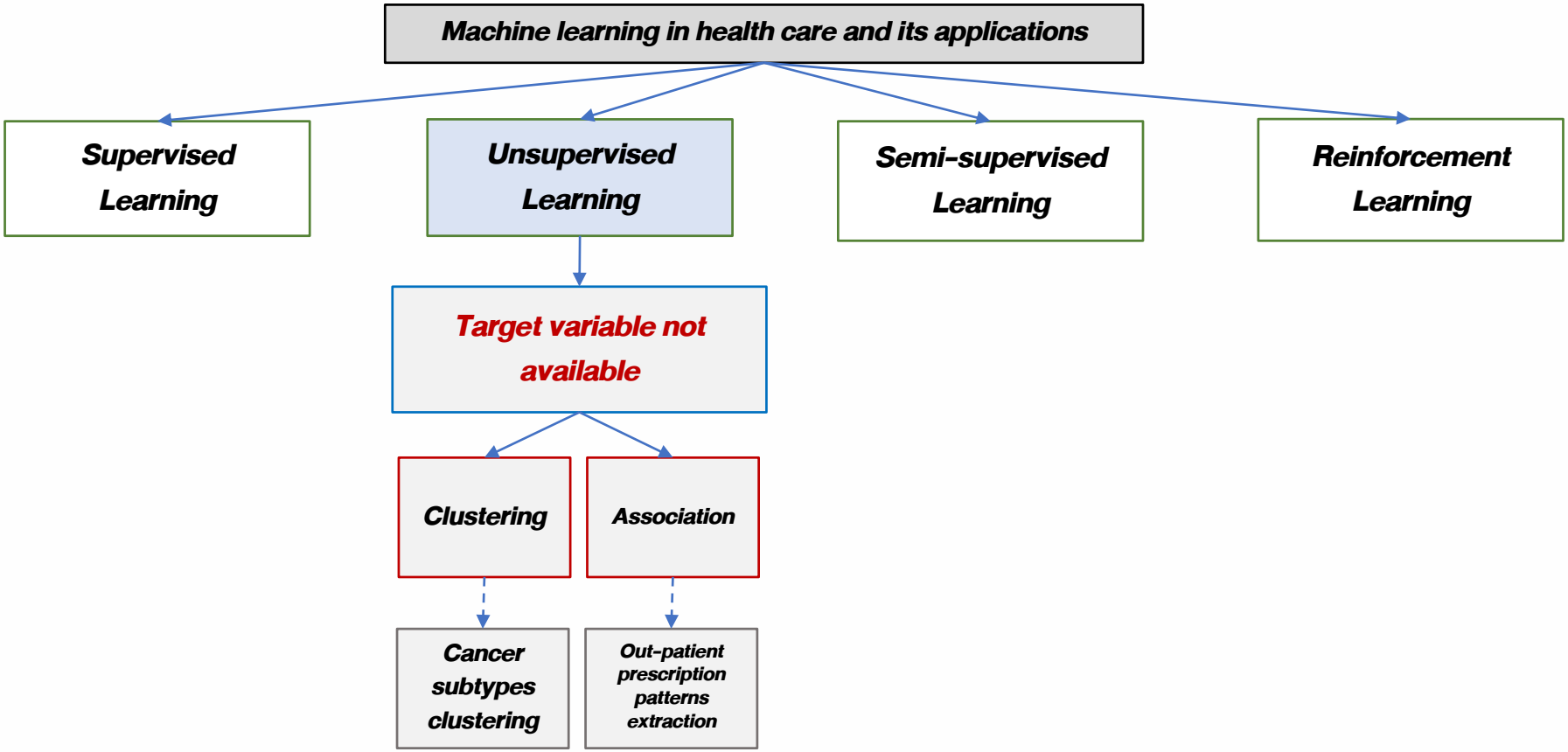
A clustering model
(Mathematical function + Algorithms)

y

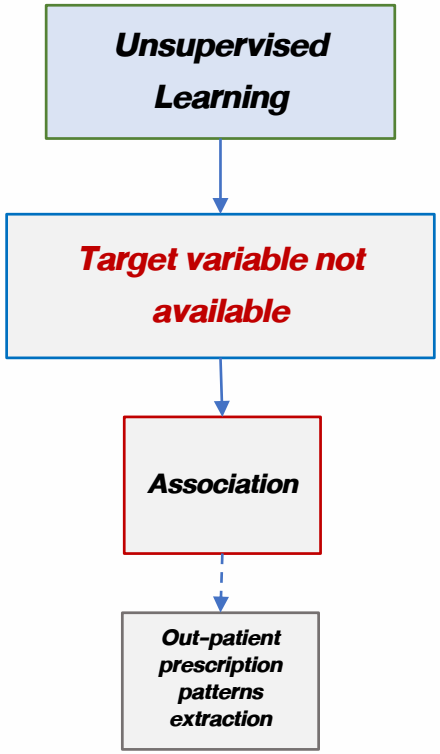
A class or label (ALL/AML/DLL)



Machine learning in health care and its applications



Machine learning in health care and its applications



Training dataset

t_1:

HNDate	DrugCode1	DrugCode2
AA20141001	OMPZ	XAND

t_2:

HNDate	DrugCode1	DrugCode2	DrugCode3	DrugCode4
AA20141002	CELB	OMPZ	XAND	ASPT

• • •

t_n:

HNDate	DrugCode1	DrugCode2	DrugCode3
AB20141001	NAPX	OMPZ	CAPN

T

Unsupervised Learning (Association)

Apriori algorithm

Testing dataset

HNDate	DrugCode1	DrugCode2	DrugCode3
AB20141001	NAPX	OMPZ	CAPN

HNDate	DrugCode1	DrugCode2
AA20141001	OMPZ	XAND

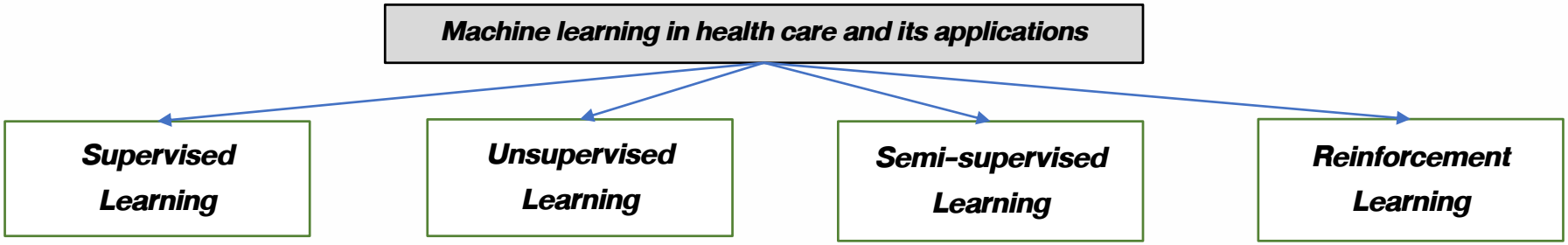
A set of association rules
(Mathematical function + Algorithms)

Rule no.	LHS	RHS	Support	Confidence	Lift
1	OMPZ	XAND	0.0552	0.7944	7.6153
2	XAND	OMPZ	0.0552	0.5288	7.6153
3	NAPX	OMPZ	0.1085	0.9738	1.4363
4	MELO	OMPZ	0.0315	0.9557	1.4096
5	IBUP	OMPZ	0.0362	0.9028	1.3317
6	DICF	OMPZ	0.0109	0.8933	1.3177
7	ASPT	OMPZ	0.0399	0.8723	1.2867
8	ASA	OMPZ	0.2244	0.7840	1.1564
9	CELB	OMPZ	0.0483	0.7582	1.1184
10	MOBC	OMPZ	0.0133	0.7522	1.1096
11	ARCX	OMPZ	0.0860	0.6901	1.0179
12	ANTC	OMPZ	0.0315	0.5543	0.8176

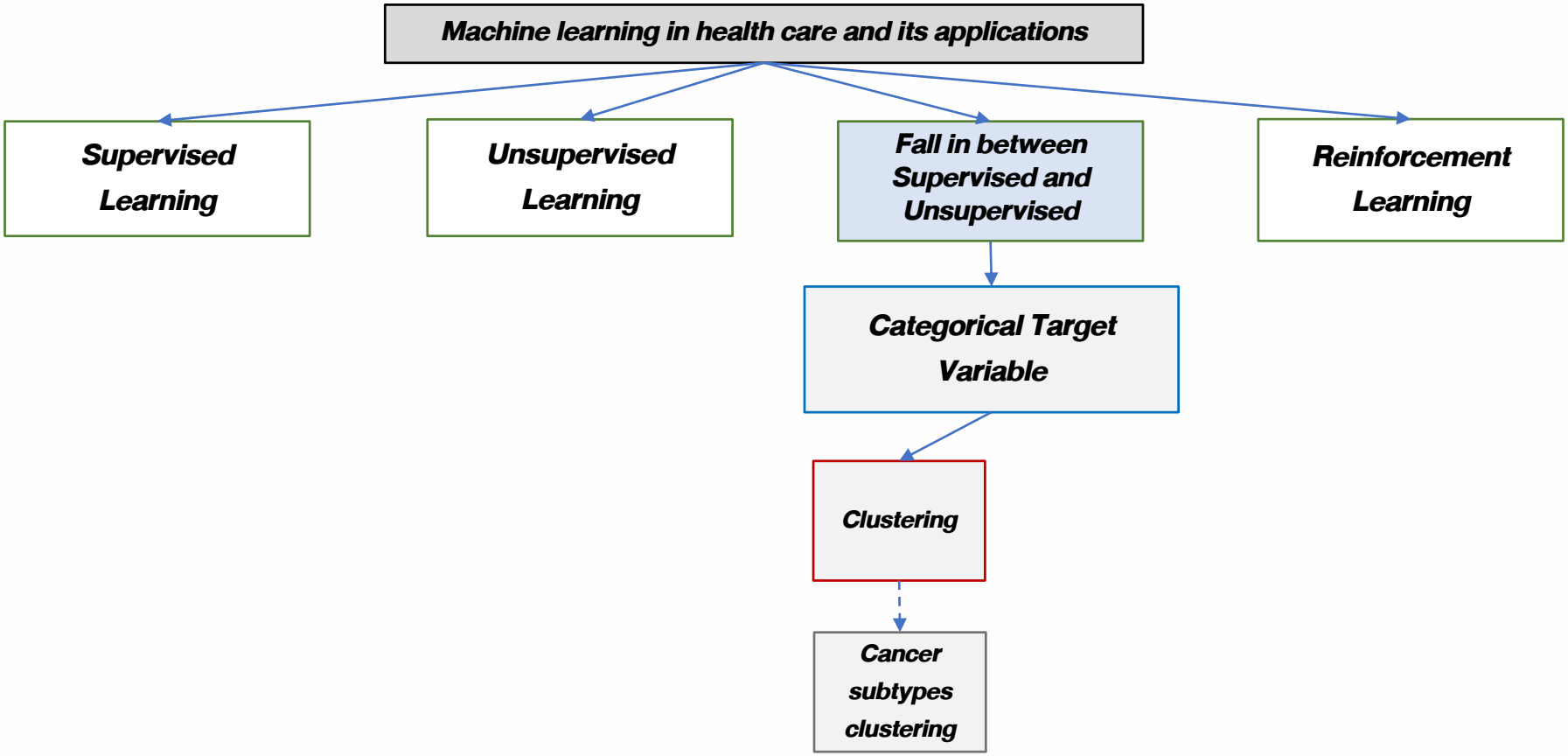
y

A pattern of prescription:
(when X occurs, Y occurs with certain probability)

Machine learning in health care and its applications

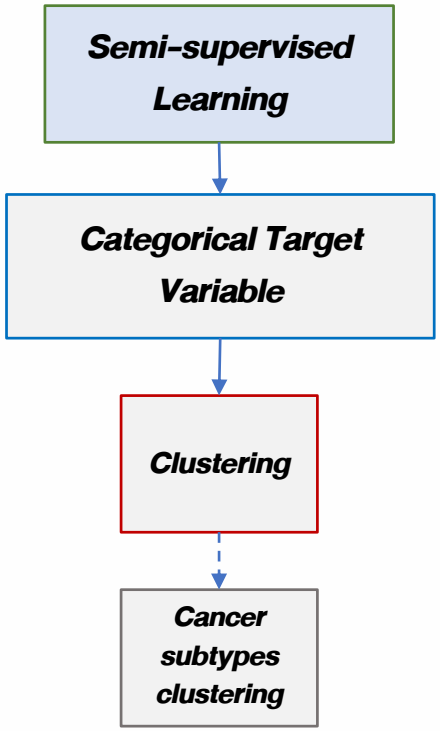


Machine learning in health care and its applications



Mahidol University
Faculty of Medicine Ramathibodi Hospital
Department of Clinical Epidemiology and Biostatistics

Machine learning in health care and its applications



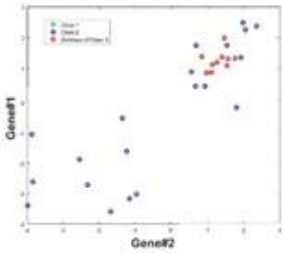
Training dataset

gene	sample#1	sample#2	sample#3	...	sample#30
g#1	200.3365	-50.6635	-152.6875	...	1.0256
g#2	135.3356	-5.6669	-32.7458	...	2.0456
Label/Target	uknown	uknown	uknown	...	uknown

Semi-supervised Learning (Clustering)

K means recursive clustering

2-Dimensions plots



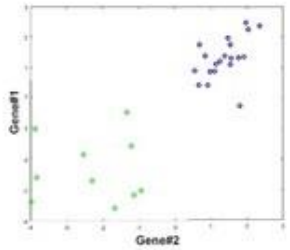
Testing dataset

gene	sample#1	sample#2	sample#3	...	sample#10
g#1	11.2335	-1.5684	-10.5692	...	5.3368
g#2	16.5555	-2.5611	-15.3368	...	6.9561
Label/Target	Class 1	Unknown	Unknown	...	Class 2

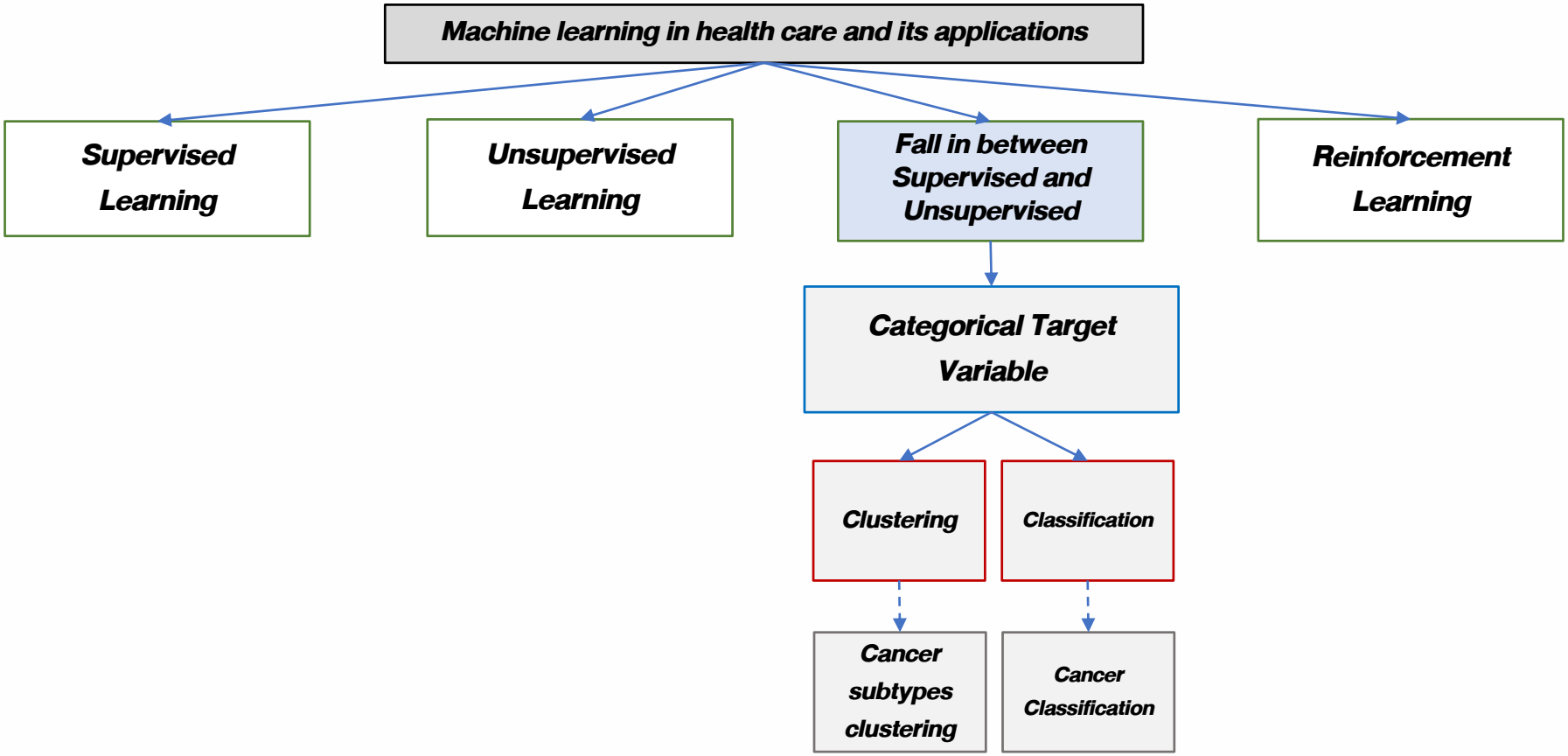
A clustering model
(Mathematical function + Algorithms)

y

A class or label
(Sub Class 1/ Sub Class 2)

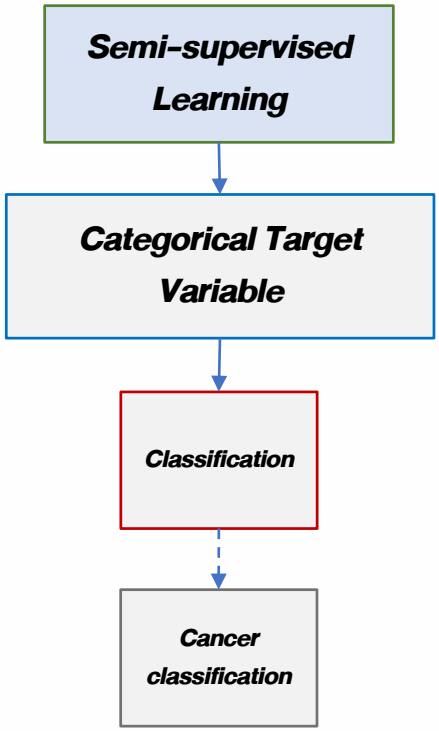


Machine learning in health care and its applications



Mahidol University
Faculty of Medicine Ramathibodi Hospital
Department of Clinical Epidemiology and Biostatistics

Machine learning in health care and its applications



Training dataset

gene	sample#1	sample#2	...	sample#100
g#1	11.2335	-1.5684	...	5.3368
g#2	16.5555	-2.5611	...	6.9561
g#3	10.2225	-1.5666	...	6.5834
g#4	8.4568	-4.5669	...	4.8136
g#5	11.2268	-3.2665	...	3.3365
g#6	7.9846	0	...	9.9615
g#7	15.2664	-1.1256	...	4.2468
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
g#10000	21.3354	-4.3651	...	2.3681
Label/Target	1	-1		Unknown

Label $y \rightarrow \{+1, -1, \text{unknown}\}$
+1 = healthy, -1 = cancer

Semi-supervised Learning (Classification)

Transductive SVM

Testing dataset

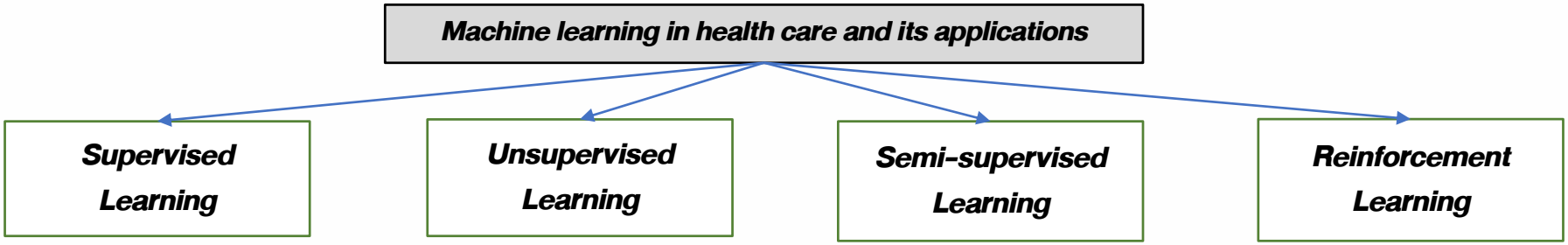
gene	sample#1	sample#2	...	sample#100
g#1	11.2335	-1.5684	...	5.3368
g#2	16.5555	-2.5611	...	6.9561
g#3	10.2225	-1.5666	...	6.5834
g#4	8.4568	-4.5669	...	4.8136
g#5	11.2268	-3.2665	...	3.3365
g#6	7.9846	0	...	9.9615
g#7	15.2664	-1.1256	...	4.2468
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
g#10000	21.3354	-4.3651	...	2.3681
Label/Target	1	-1		1

A classifier
(Mathematical function + Algorithms)

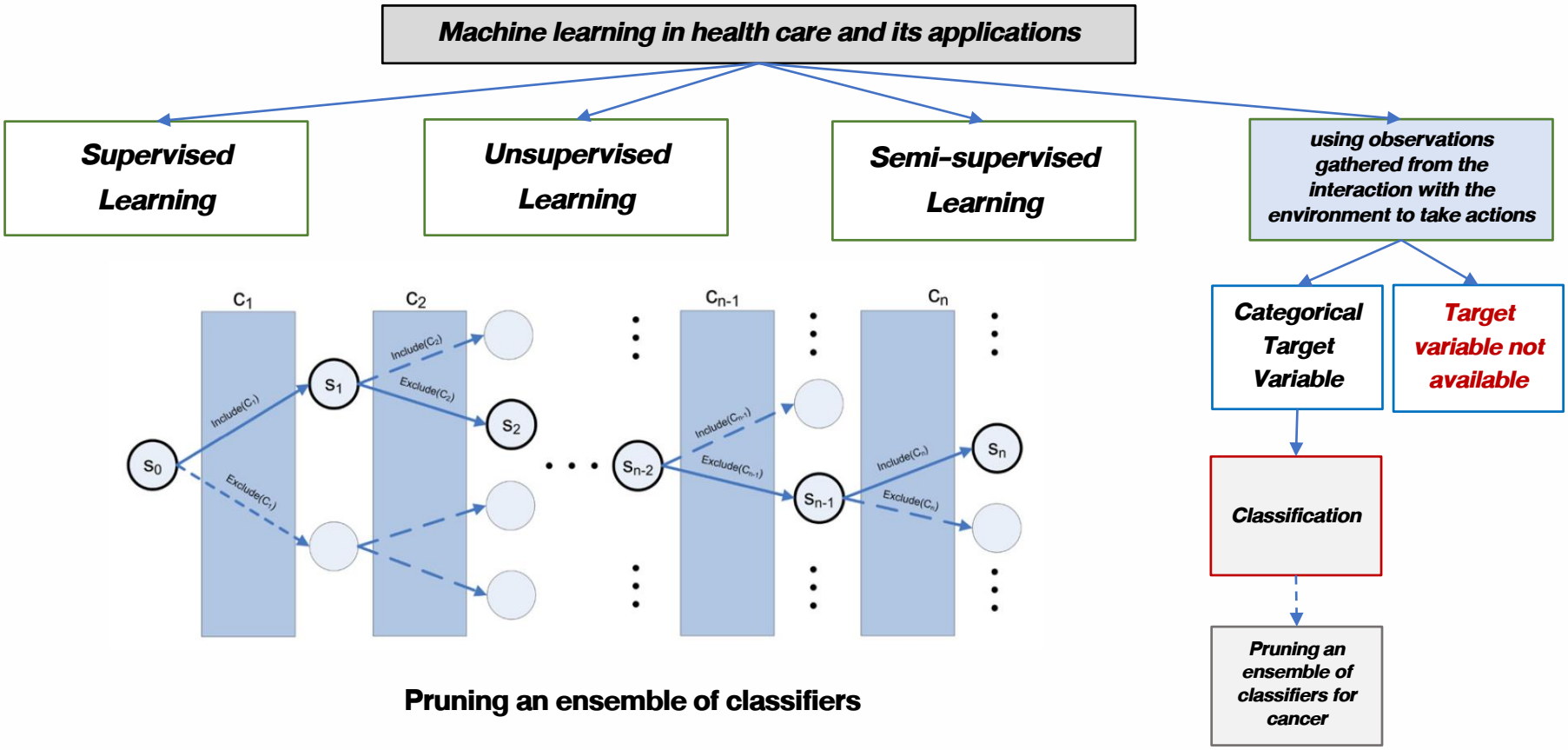
y

A class or label (Healthy/Cancer)

Machine learning in health care and its applications



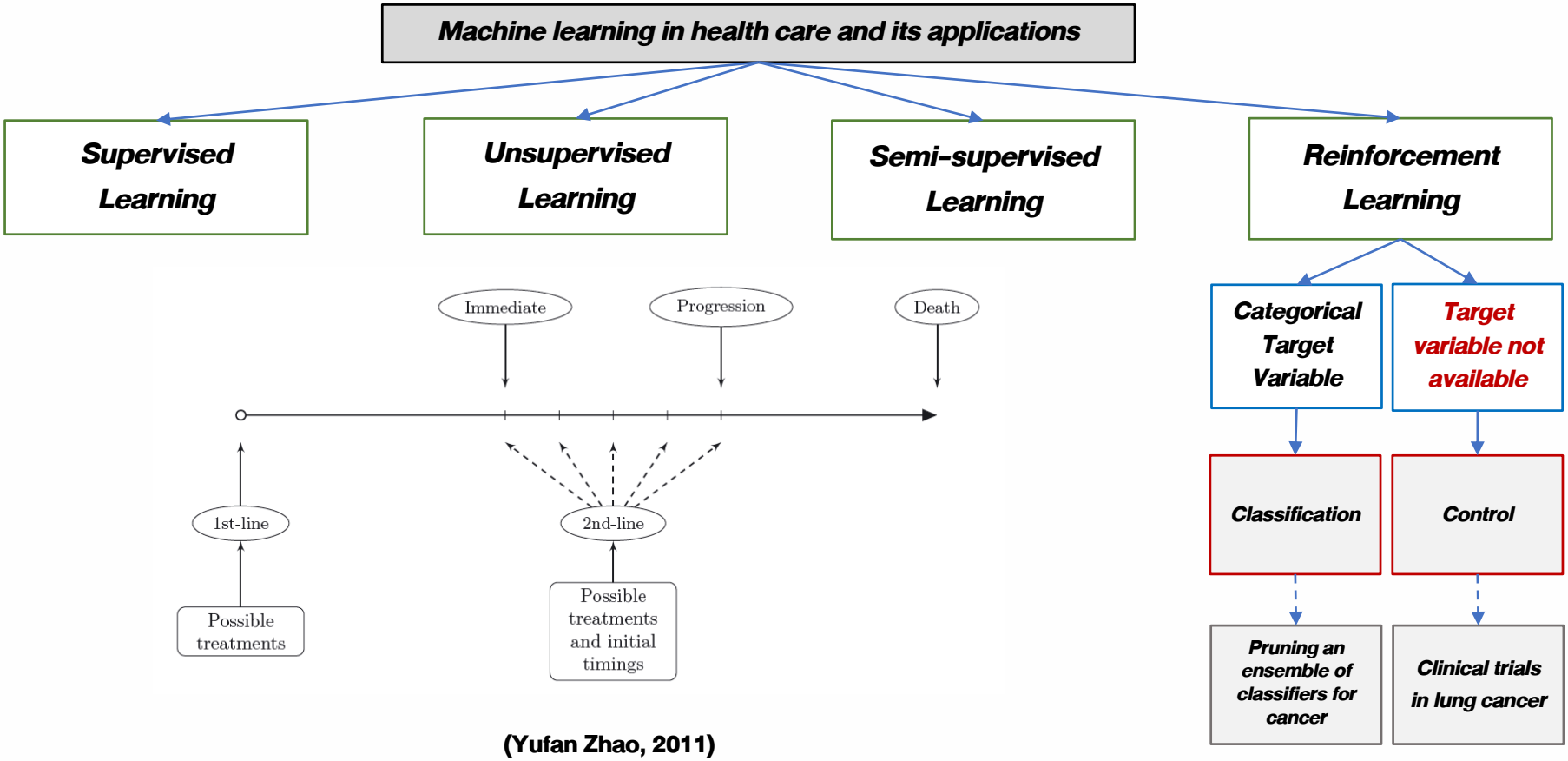
Machine learning in health care and its applications



Pruning an ensemble of classifiers

(Ioannis Partalas, 2009)

Machine learning in health care and its applications



(Yufan Zhao, 2011)

Machine learning in health care and its applications

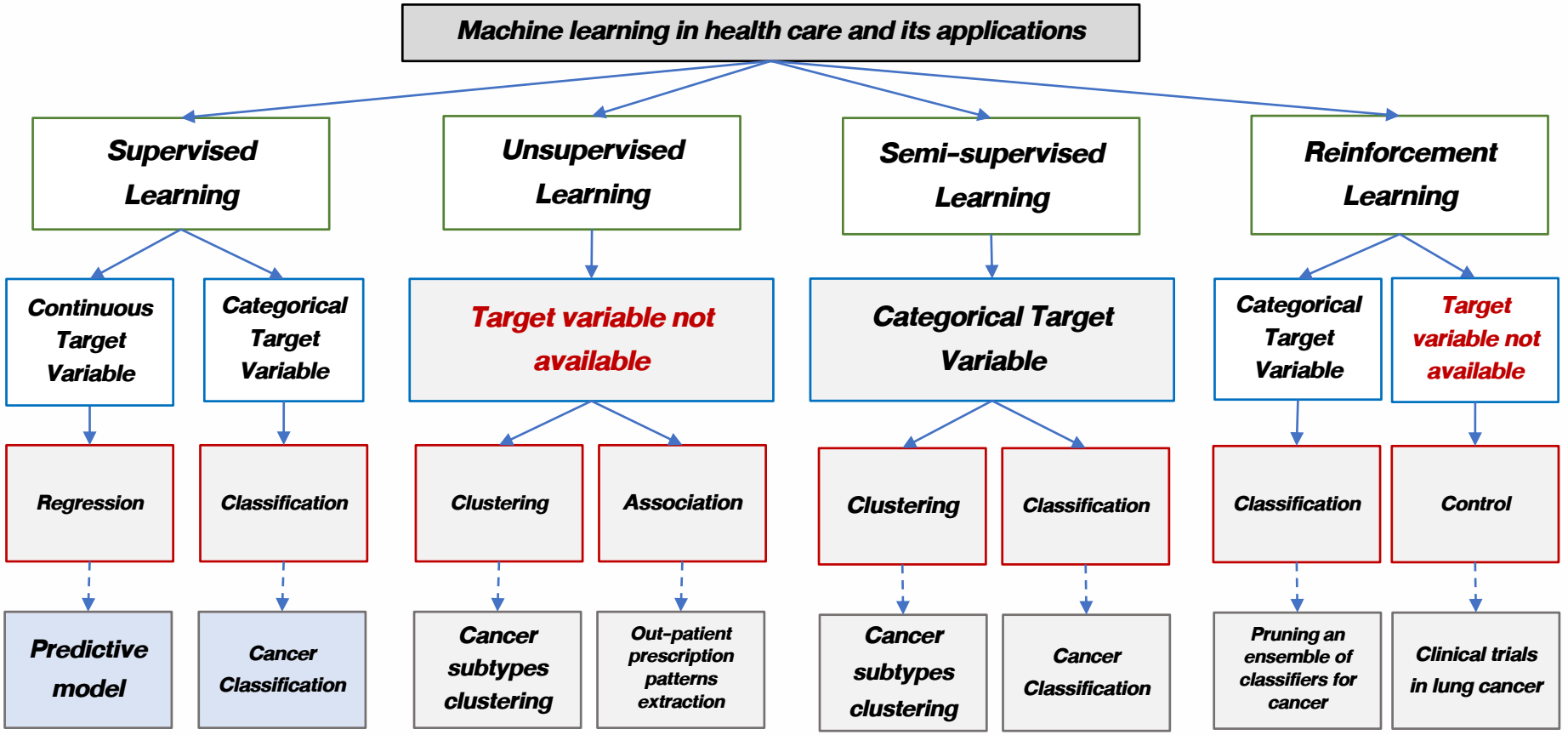


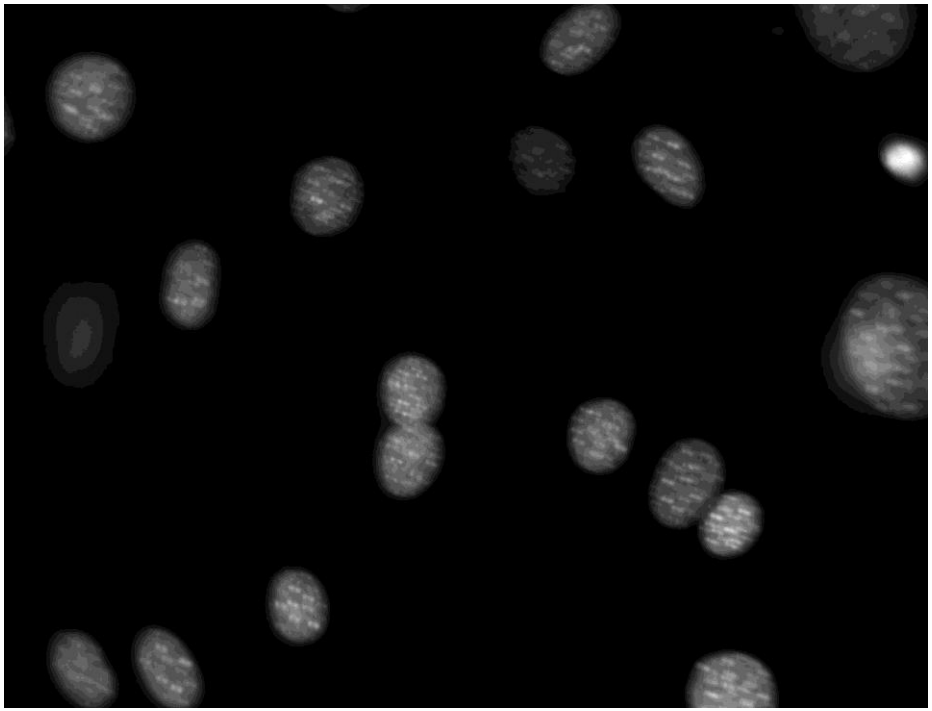
Image Analysis Applications

Image analytics in health care

- Image classification
  - Prediction: predict whether a particular pathology exits or not
  - Diagnosis: skin disease diagnosis
- Image segmentation
  - Study anatomical structures
  - Locate tumour, lesions, cancer or other abnormaliaties
- Image description
  - Generate image captions
  - Generate automated report from image analytics

Counting Objects

We are going to count the number of nuclei from the image (dna.jpeg)

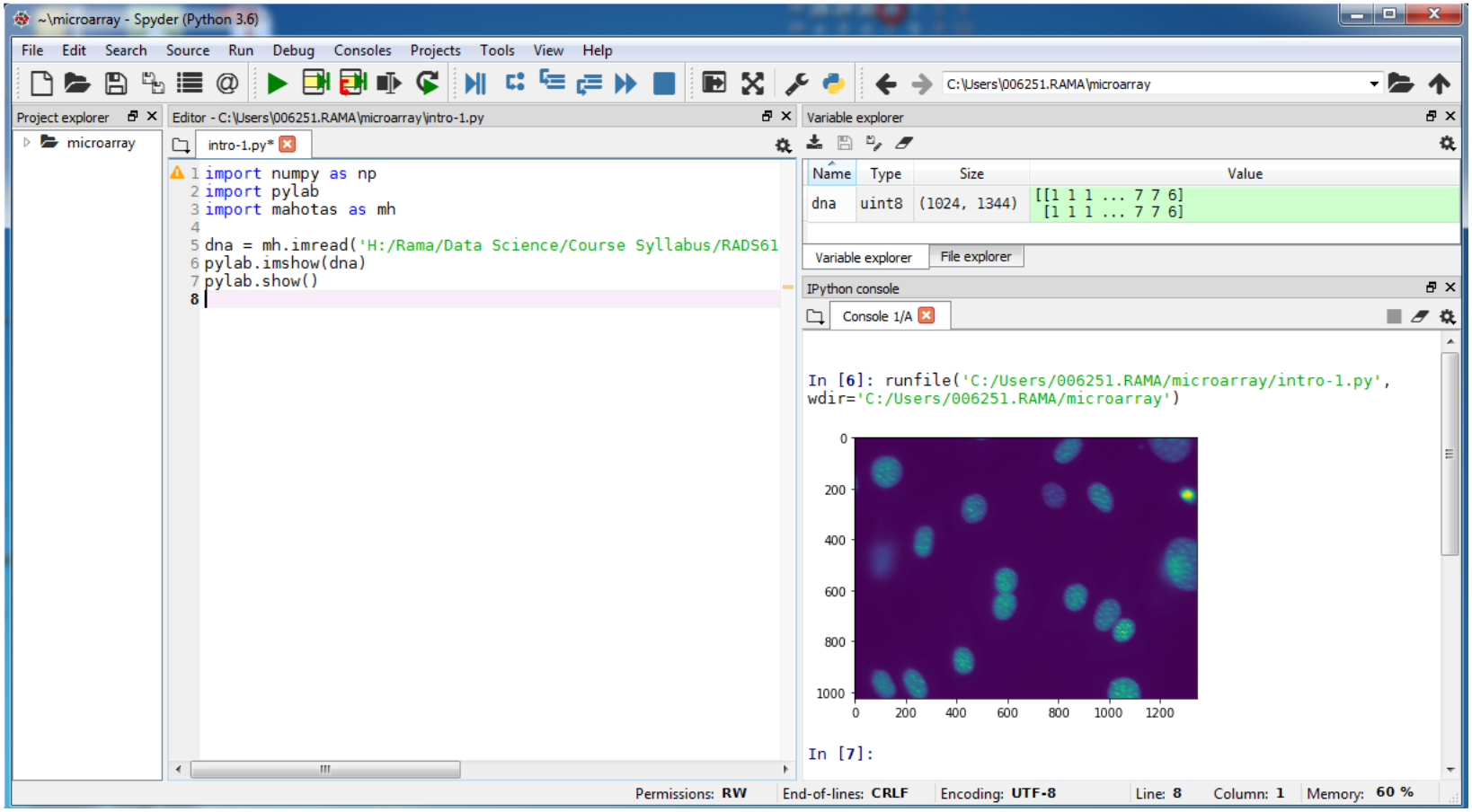


http://pythonvision.org/basic-tutorial/

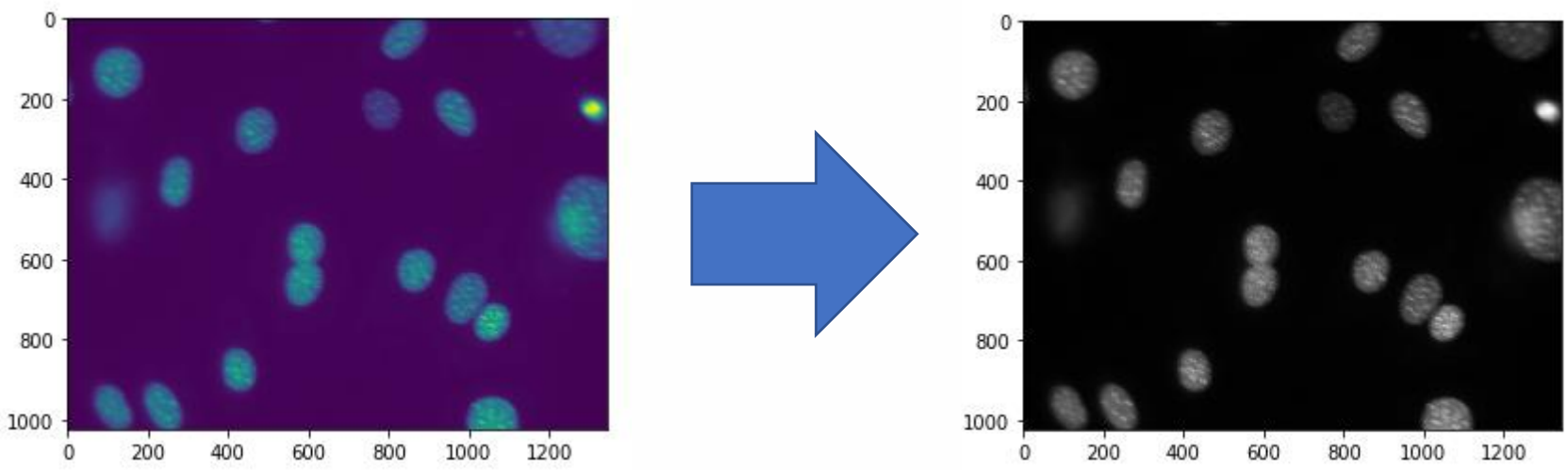
Get image from disk into a memory array

```
import numpy as np
import pylab
import mahotas as mh

dna = mh.imread('H:/image-processing/dna.jpg')
pylab.imshow(dna)
pylab.show()
```

Get image from disk into a memory array



Change the heatmap to the grey-scale



pylab shows images as a heatmap. You can set it to the traditional grey-scale image by:

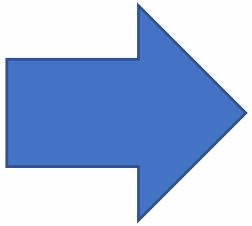
pylab.imshow(dna)

pylab.gray()

pylab.show()

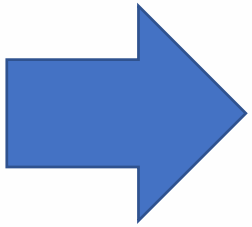
Explore a memory array

```
print(dna.shape)
print(dna.dtype)
print(dna.max())
print(dna.min())
```



(1024, 1344)	height x width
Uint8	unsigned 8-bit integer
252	Maximum value is 252
0	Minimum value is 0

```
pylab.imshow(dna // 2)
pylab.show()
```



displaying an image where all the values have been divided by 2.

pylab contrast-stretches a image before display

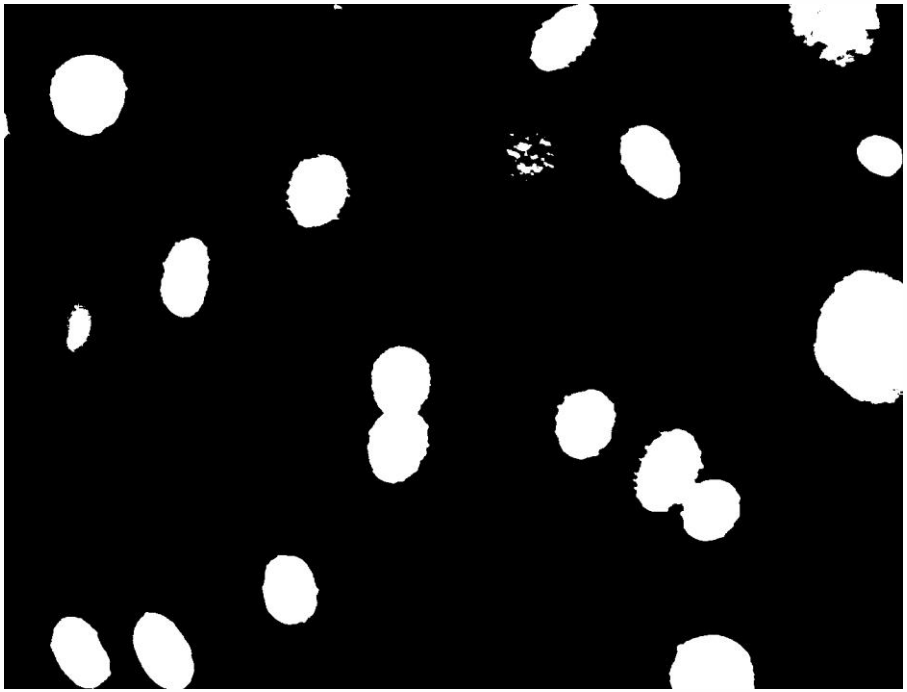
Threshold the image and count the number of objects

the fact that dna is a numpy array and using it in logical operations (dna > T).

The result is a numpy array of booleans, which pylab shows as a black and white image (or red and blue if you have not previously called pylab.gray())

```
T = mh.thresholding.otsu(dna)          #T=45
pylab.imshow(dna > T)
pylab.show()
```

Threshold the image and count the number of objects



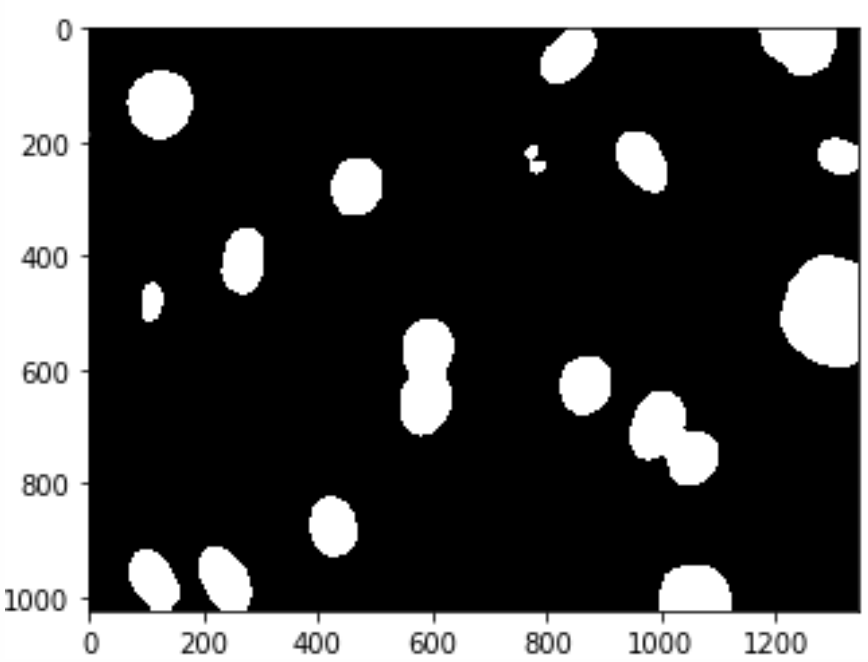
The image contains many small objects.

We can count the number of objects by using a Gaussian filter.

Objects counting with a Gaussian filter

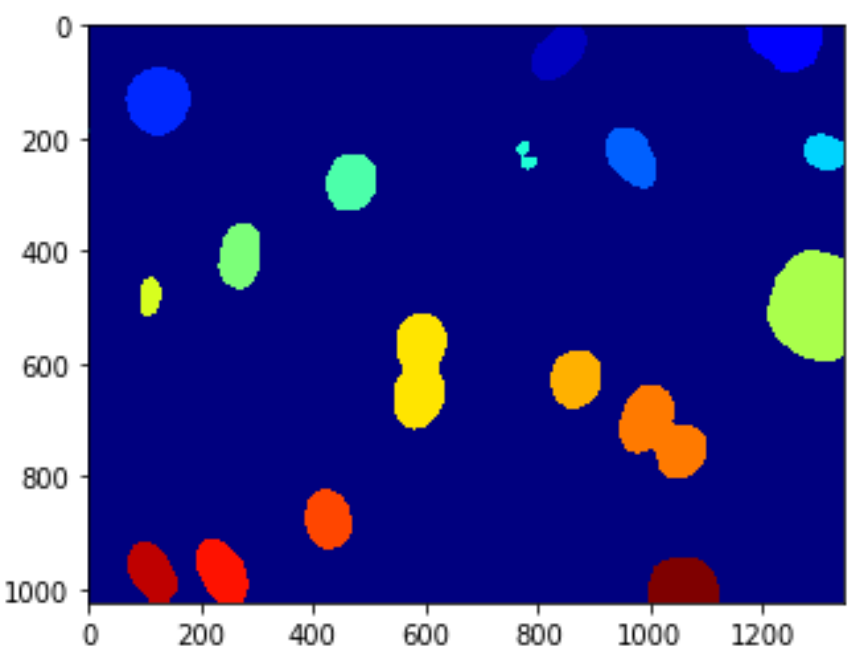
```
dnaf = mh.gaussian_filter(dna,8)          #sigma = 8
dnaf = dnaf.astype('uint8')
T = mh.thresholding.otsu(dnaf)
pylab.imshow(dnaf > T)
pylab.show()
```

The function mh.gaussian_filter takes an image and the standard deviation of the filter (in pixel units) and returns the filtered image.



Objects counting with a Gaussian filter

```
labeled,nr_objects = mh.label(dnaf > T)
print(nr_objects)
pylab.imshow(labeled)
pylab.jet()
pylab.show()
```



The number of objects in the image =18.

pylab.jet() is for the resetting of colourmap to jet (if we have the greyscale map active)

the distance transform of the thresholded image

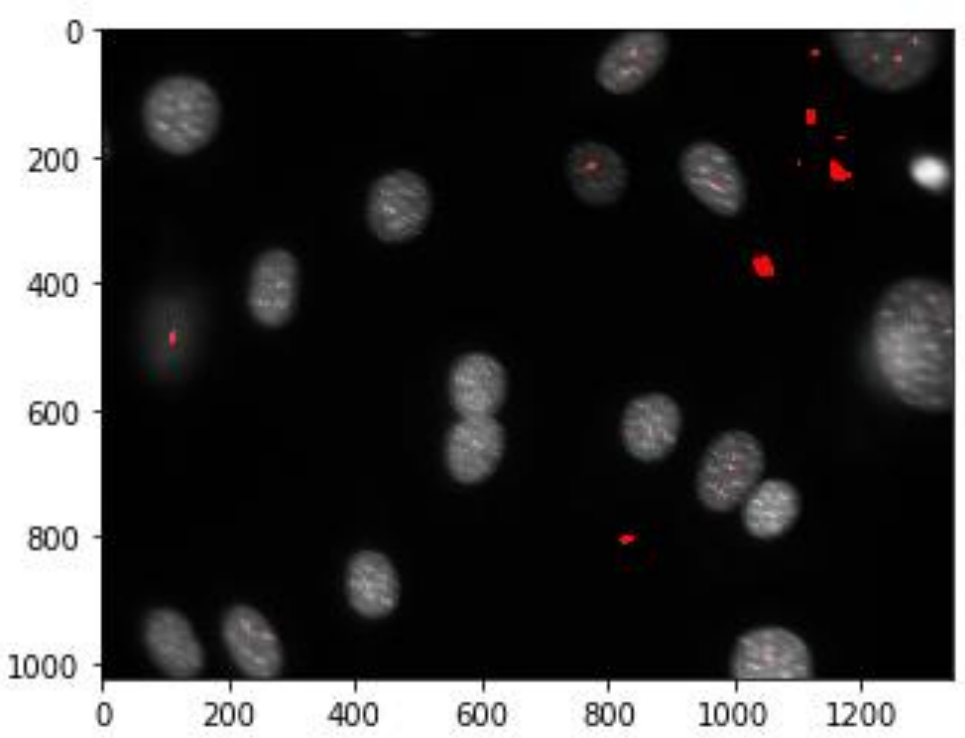
Traditional idea:

- Smooth the image
- Find regional maxima
- Use the regional maxima as seeds for watershed

Finding the seeds, sigma value = 8

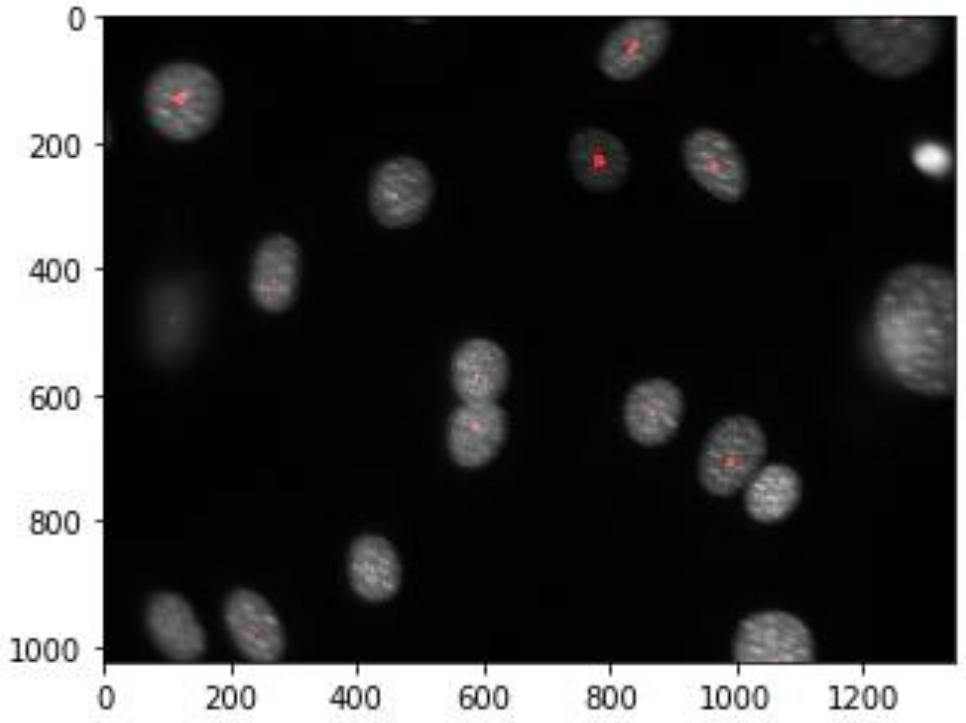
```
dnaf = mh.gaussian_filter(dna, 8)
dnaf = dnaf.astype('uint8')
rmax = mh.regmax(dnaf)
pylab.imshow(mh.overlay(dna, rmax))
pylab.show()
```

mh.overlay() returns a colour image with the grey level component being given by its first argument while overlaying its second argument as a red channel.



Finding the seeds, sigma value = 16

```
dnaf = mh.gaussian_filter(dna, 16)
dnaf = dnaf.astype('uint8')
rmax = mh.regmax(dnaf)
pylab.imshow(mh.overlay(dna, rmax))
pylab.show()
```



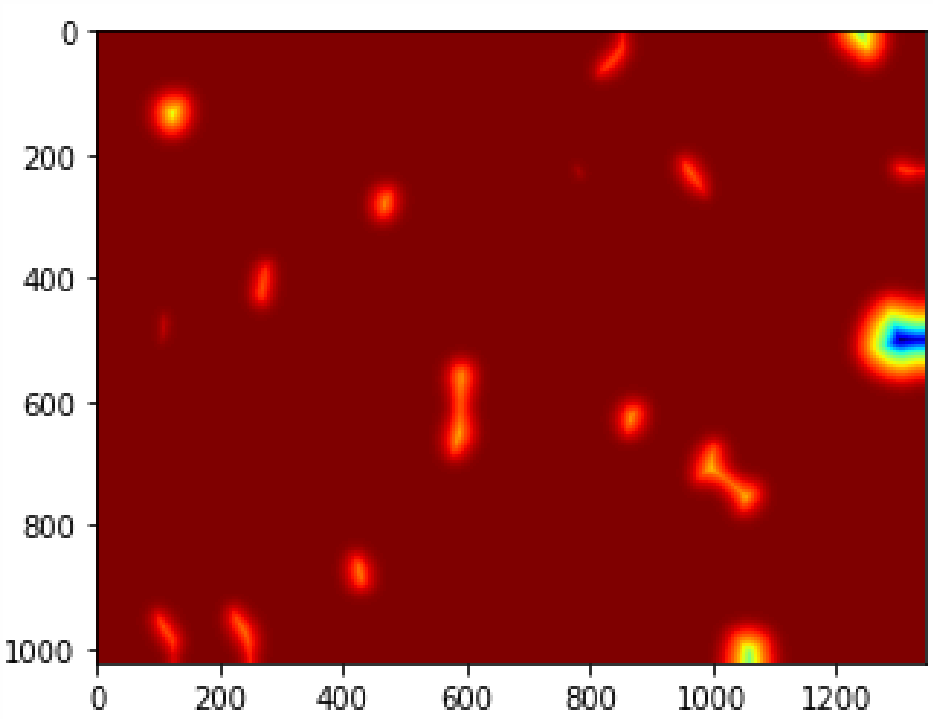
Count the number of nuclei

```
seeds,nr_objects = mh.label(rmax)
print(nr_objects)
```

Apply watershed to the distance transform of the thresholded image

```
T = mh.thresholding.otsu(dnaf)
dist = mh.distance(dnaf > T)
dist = dist.max() - dist
dist -= dist.min()
dist = dist/float(dist.ptp()) * 255
dist = dist.astype(np.uint8)
pylab.imshow(dist)
pylab.jet()
pylab.show()
```

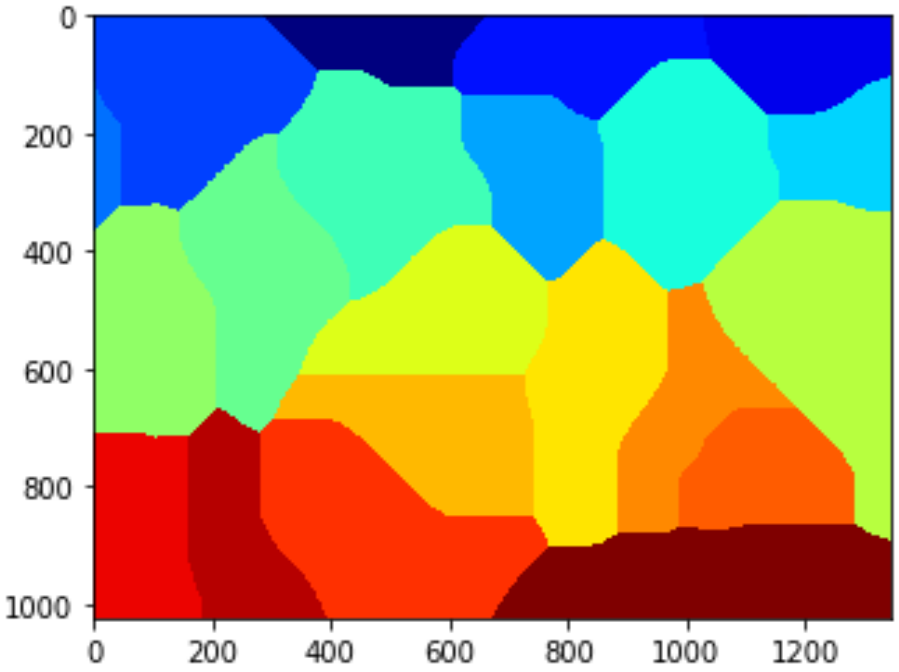
we contrast stretched the dist image



Apply watershed to the distance transform of the thresholded image

```
nuclei = mh.cwatershed(dist, seeds)
whole = mh.segmentation.gvoronoi(nuclei)
pylab.imshow(whole)
pylab.show()
```

each pixel gets assigned to its nearest




```
print("nuclei in the image : ", len(np.unique(whole)))
```

Mahidol University
Faculty of Medicine Ramathibodi Hospital
Department of Clinical Epidemiology and Biostatistics

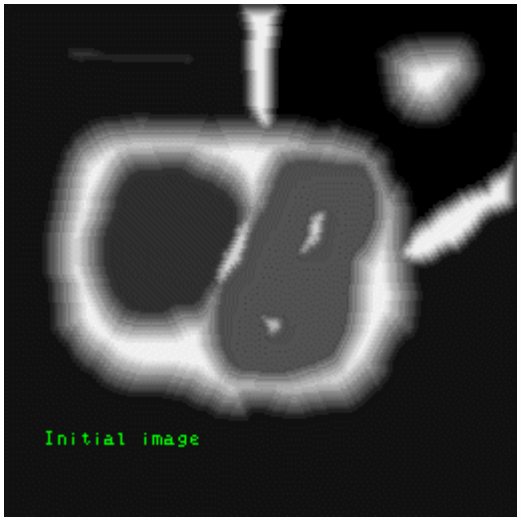
Image Segmentation with Watershed by using a cv2 library

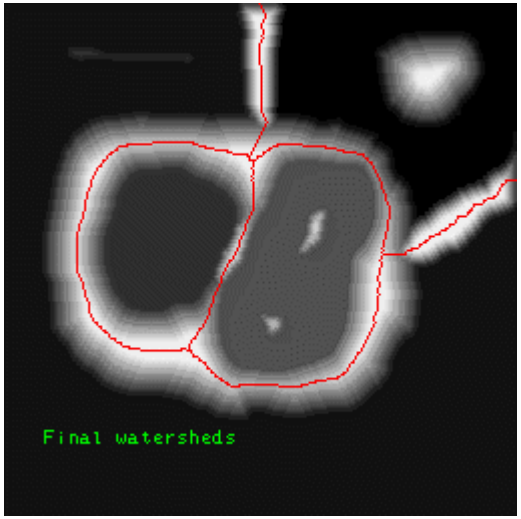
What's a Watershed algorithm

- Use in image processing primarily for segmentation purposes
- Is a transformation defined on a grayscale image

The term watershed refers to a ridge that divides areas drained by different river systems.

A catchment basin is the geographical area draining into a river or reservoir.





What's a Watershed algorithm

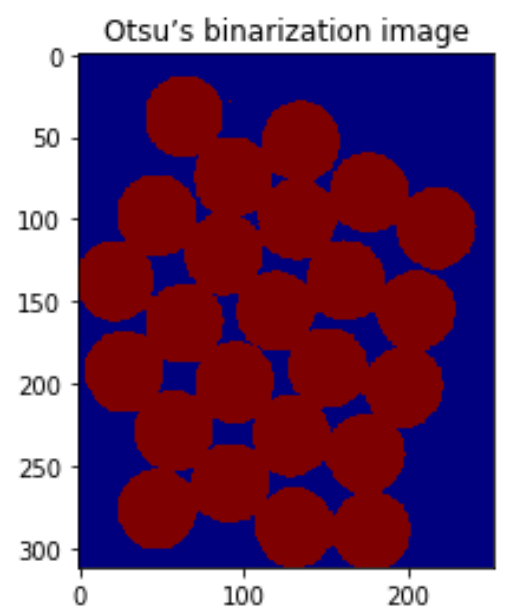
If we flood this surface from its minima and, if we prevent the merging of the waters coming from different sources, we partition the image into two different sets: the catchment basins and the watershed lines.

Image Segmentation with Watershed by using a cv2 library

```
import numpy as np
import cv2
from matplotlib import pyplot as plt


img = cv2.imread('H:/image-processing/water_coins.jpg')
plt.imshow(img)
plt.title('Original image')
plt.show()
```

Image Segmentation with Watershed by using a cv2 library

```
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
ret, thresh = cv2.threshold(gray,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
plt.imshow(thresh)
plt.title('Otsu's binarization image')
plt.show()
```



We start with finding an approximate estimate of the coins. For that, we can use the Otsu's binarization.

Image Segmentation with Watershed by using a cv2 library

```
# noise removal
kernel = np.ones((3,3),np.uint8)
opening = cv2.morphologyEx(thresh,cv2.MORPH_OPEN,kernel, iterations = 2)
plt.imshow(opening)
plt.title('Noise removal image')
plt.show()
```

We remove small white noises in the image by using a morphological opening.

To remove any small holes in the object, we can use morphological closing.

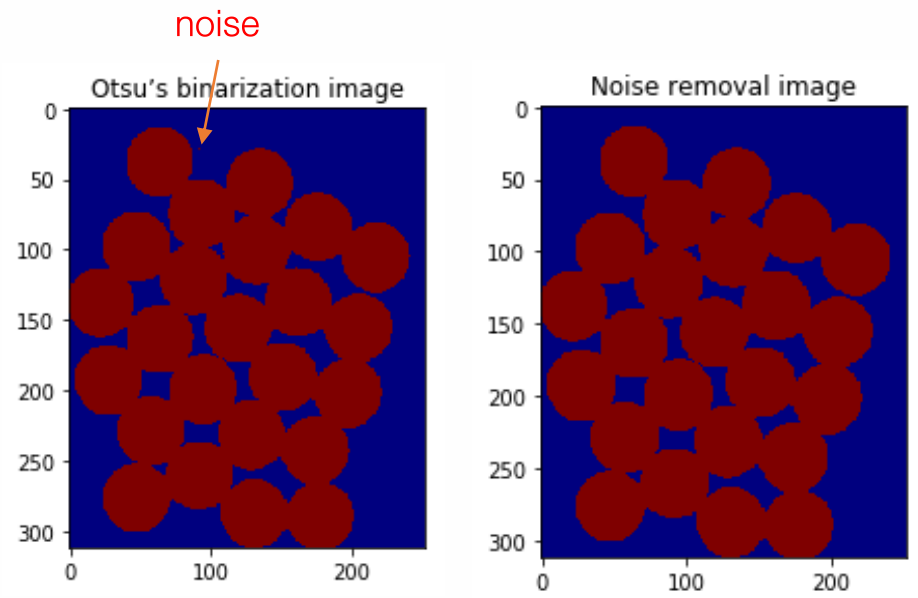


Image Segmentation with Watershed by using a cv2 library

```
# sure background area
sure_bg = cv2.dilate(opening,kernel,iterations=3)
plt.imshow(sure_bg)
plt.title('Sure background image')
plt.show()
```

The region away from the object are background

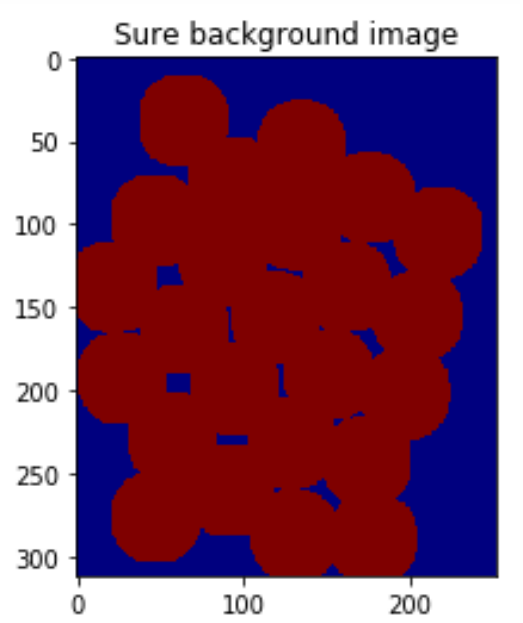
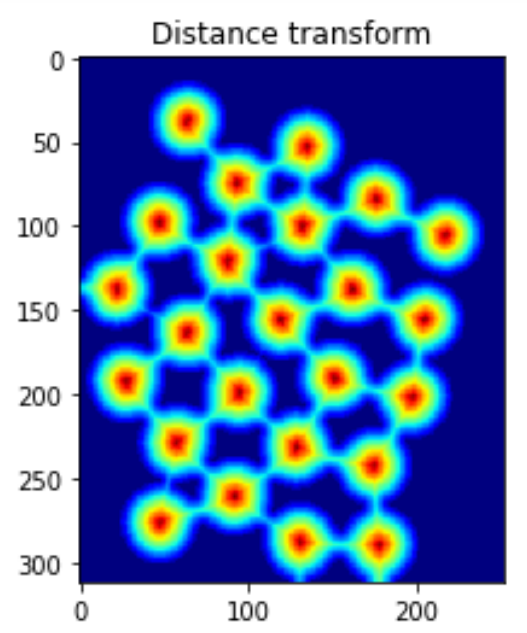


Image Segmentation with Watershed by using a cv2 library

```
# Finding sure foreground area
dist_transform = cv2.distanceTransform(opening,cv2.DIST_L2,5)
plt.imshow(dist_transform)
plt.title('Distance transform')
plt.show()
```



Use the distance transform to find sure foreground area

Image Segmentation with Watershed by using a cv2 library

```
# Finding sure foreground area
ret, sure_fg = cv2.threshold(dist_transform,0.7*dist_transform.max(),255,0)
plt.imshow(sure_fg)
plt.title('Red region shows sure foreground')
plt.show()
```

The region near to center of objects are foreground

Only region we are not sure is the boundary region of coins

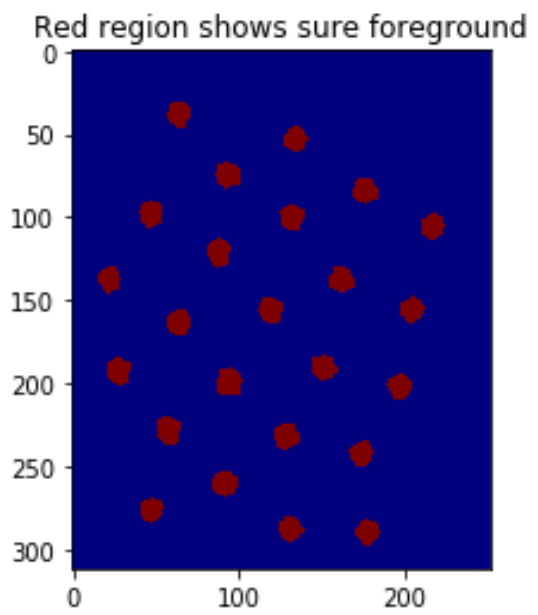
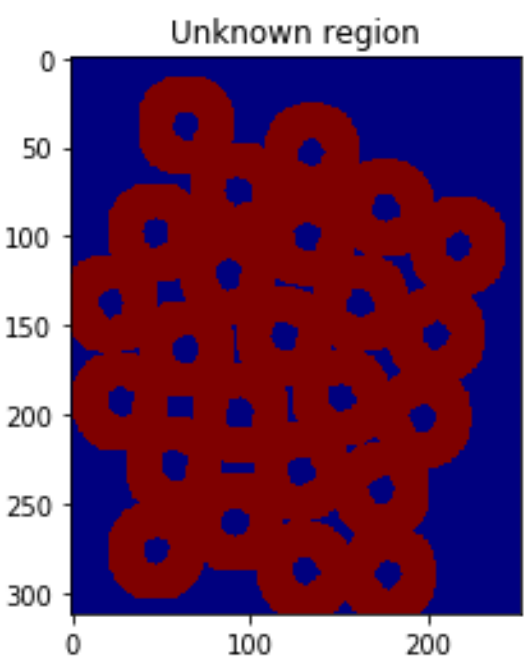


Image Segmentation with Watershed by using a cv2 library

```
# Finding unknown region
sure_fg = np.uint8(sure_fg)
unknown = cv2.subtract(sure_bg, sure_fg)
plt.imshow(unknown)
plt.title('Unknown region')
plt.show()
```

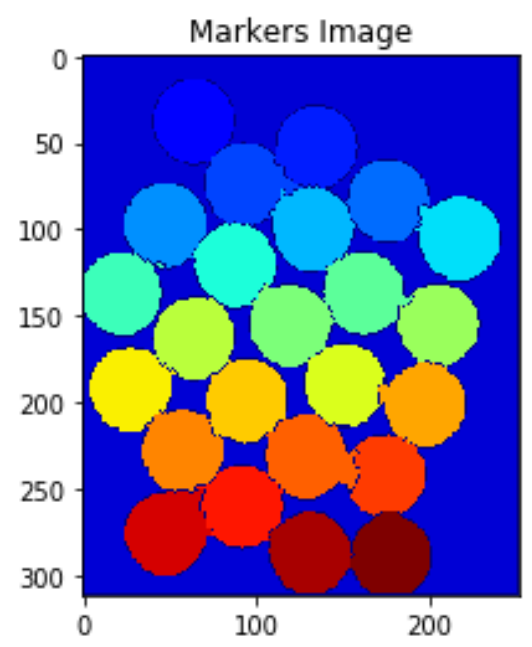


The red region shows unknown region

Next, the marker image will be modified.

Image Segmentation with Watershed by using a cv2 library

```
# Marker labelling
ret, markers = cv2.connectedComponents(sure_fg)

# Add one to all labels so that sure background is not 0, but 1
markers = markers+1

# Now, mark the region of unknown with zero
markers[unknown==255] = 0
markers = cv2.watershed(img,markers)
plt.imshow(markers)
plt.title('Markers Image')
plt.show()
```

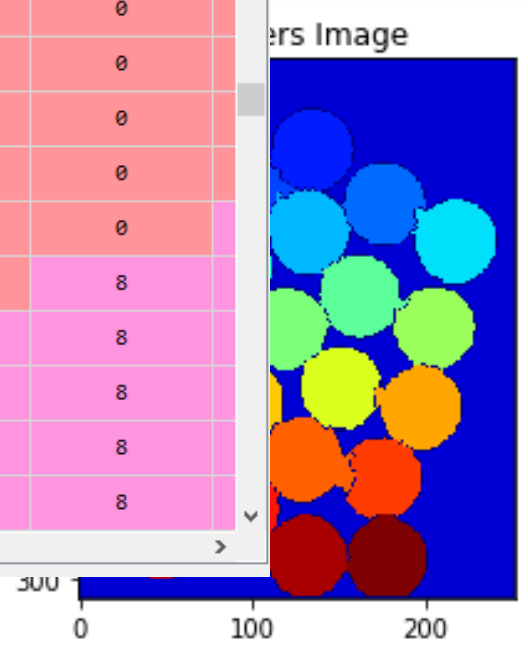


Mahidol University
Faculty of Medicine Ramathibodi Hospital
Department of Clinical Epidemiology and Biostatistics

Image Segmentation with Watershed by using a cv2 library

```
# Marker labelling
ret, markers = cv2.conn

# Add one to all labels s
markers = markers+1

# Now, mark the region
markers[unknown==25
markers = cv2.watershe
plt.imshow(markers)
plt.title('Markers Image'
plt.show()
```

	78	79	80	81	82
110	0	0	0	0	0
111	0	0	0	0	0
112	0	0	0	0	0
113	0	0	0	0	0
114	0	0	0	0	0
115	0	0	0	0	0
116	0	0	0	0	0
117	0	0	0	0	8
118	0	0	0	8	8
119	0	0	0	8	8
120	0	0	0	8	8
121	0	0	0	8	8

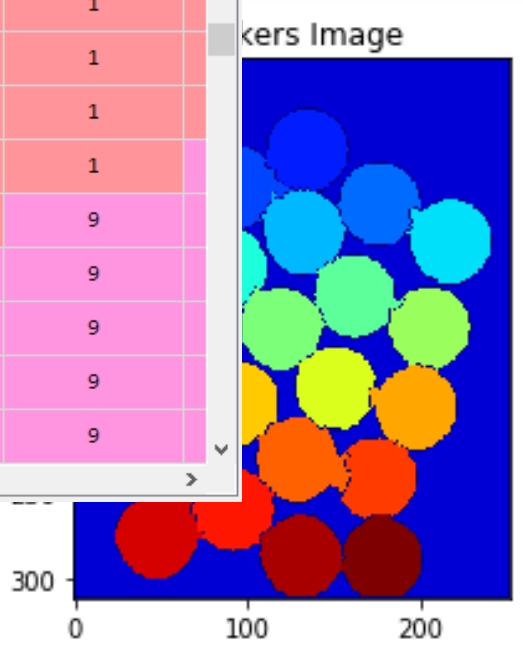


Mahidol University
Faculty of Medicine Ramathibodi Hospital
Department of Clinical Epidemiology and Biostatistics

Image Segmentation with Watershed by using a cv2 library

```
# Marker labelling
ret, markers = cv2.con

# Add one to all labels
markers = markers+1

# Now, mark the regior
markers[unknown==25
markers = cv2.watersh
plt.imshow(markers)
plt.title('Markers Image')
plt.show()
```

	78	79	80	81	82
110	1	1	1	1	1
111	1	1	1	1	1
112	1	1	1	1	1
113	1	1	1	1	1
114	1	1	1	1	1
115	1	1	1	1	1
116	1	1	1	1	1
117	1	1	1	1	9
118	1	1	1	9	9
119	1	1	1	9	9
120	1	1	1	9	9
121	1	1	1	9	9



Mahidol University
Faculty of Medicine Ramathibodi Hospital
Department of Clinical Epidemiology and Biostatistics

Image Segmentation with Watershed by using a cv2 library

```
# Marker labelling
ret, markers = cv2.con

# Add one to all labels
markers = markers+1

# Now, mark the regior
markers[unknown==25
markers = cv2.watersh
plt.imshow(markers)
plt.title('Markers Image')
plt.show()
```

	78	79	80	81	82
110	0	0	0	0	0
111	0	0	0	0	0
112	0	0	0	0	0
113	0	0	0	0	0
114	0	0	0	0	0
115	0	0	0	0	0
116	0	0	0	0	0
117	0	0	0	0	9
118	0	0	0	9	9
119	0	0	0	9	9
120	0	0	0	9	9
121	0	0	0	9	9

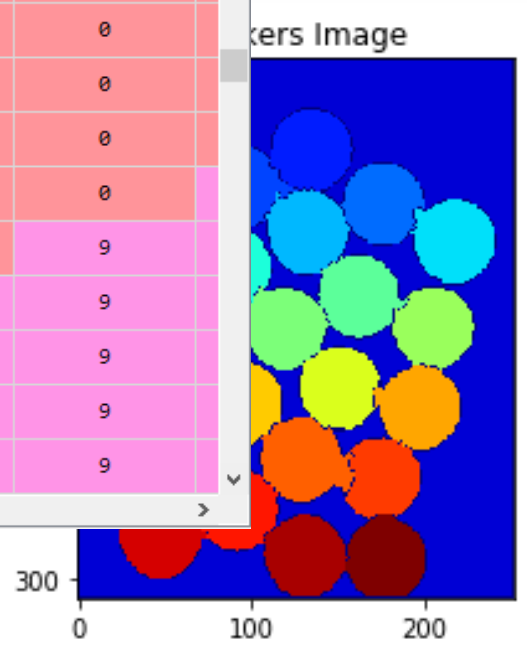


Image Segmentation with Watershed by using a cv2 library

```
# Marker labelling
ret, markers = cv2.cc

# Add one to all label
markers = markers+1

# Now, mark the regi
markers[unknown==2
markers = cv2.waters
plt.imshow(markers)
plt.title('Markers Imag
plt.show()
```

	61	62	63	64	65
110	6	6	6	6	6
111	6	6	6	6	6
112	6	6	6	6	-1
113	6	6	6	-1	9
114	6	6	-1	9	9
115	6	6	6	-1	9
116	6	6	-1	9	9
117	6	-1	1	-1	9
118	-1	1	1	-1	9
119	1	1	1	1	-1
120	1	1	1	-1	9
121	1	1	1	-1	9

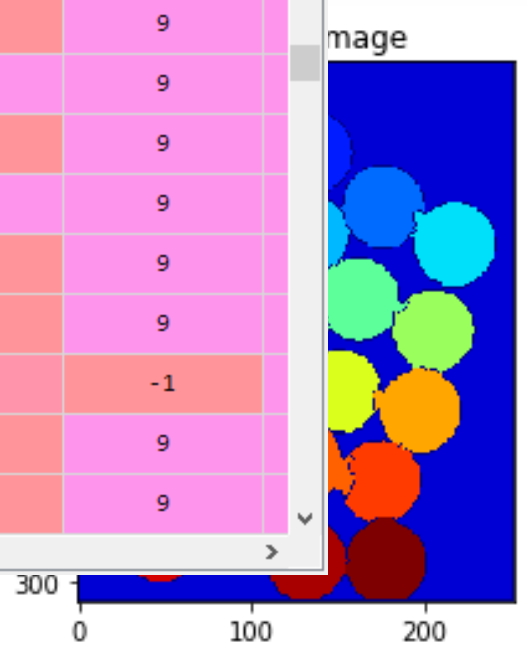
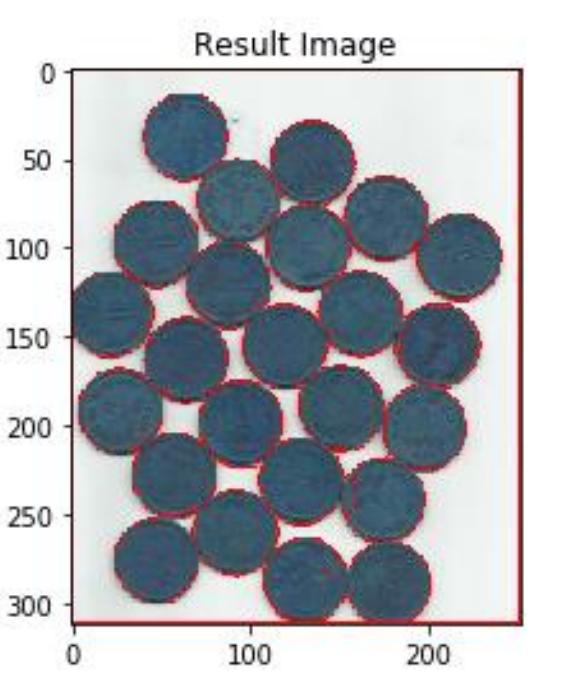


Image Segmentation with Watershed by using a cv2 library

red

```
img[markers == -1] = [255,0,0]
plt.imshow(img)
plt.title('Result Image')
plt.show()
```



Mahidol University
Faculty of Medicine Ramathibodi Hospital
Department of Clinical Epidemiology and Biostatistics

How to import and resize images using Python library

X-Ray Image dataset

https://towardsdatascience.com/deep-learning-in-healthcare-x-ray-imaging-part-3-analyzing-images-using-python-915a98fbf14c

What's a image processing?

Image processing is a method to perform some operations on an image

- get an enhanced image
- extract some useful information from image

It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image.

Analyzing images using Python

Image Dataset:

The image dataset (Chest X-Rays) was obtained from Kaggle. Dataset is available on the following link —

https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia/data

The dataset is organized into 3 folders (train, test, val) and contains subfolders for each image category (Pneumonia/Normal). There are 5,863 X-Ray images (JPEG) and 2 categories (Pneumonia/Normal). Chest X-ray images (anterior-posterior) were selected from retrospective cohorts of pediatric patients of one to five years old from Guangzhou Women and Children's Medical Center, Guangzhou.

Analyzing images using Python

```
#importing all the necessary libraries

import numpy as np
import matplotlib.pyplot as plt
import os
import cv2 as cv
import random
```

A module that comes built-in with python. It provides functions for interacting with the operating system.

OpenCV (Open Source Computer Vision Library)
pip install opencv-python

Analyzing images using Python

```
#load a single image from the bacteria folder
def load_image(path):
    for img in os.listdir(path):
        print('Image name =',img)
        image = cv.imread(os.path.join(path, img))
        break
    return image
```

The above code snippet is creating a function load_image, which will be used to load a single image from the training sets, Bacteria folder. os.listdir is used to list all the files present inside that directory. In this case, it can be used to access all the images present inside the folder Bacteria. Next, it will print the name of the image. Finally, the OpenCV library is used to read the image.

Investigating a single image from the Dataset

```
# Investigate a single image
bacteria_path = 'H://chest_xray/train/PNEUMONIA/'
image = load_image(bacteria_path)
plt.imshow(image, cmap='gray')
plt.colorbar()
plt.title('Raw Chest X Ray Image')
print(f"The dimensions are {image.shape[0]} pixels height and
{image.shape[1]} pixels width")
print(f"The maximum pixel value is {image.max():.4f}")
print(f"The minimum pixel value is {image.min():.4f}")
print(f"The mean value of the pixels is {image.mean():.4f}")
print(f"The standard deviation is {image.std():.4f}")
```

Analyzing images using Python

In this code snippet, first, the path of the images is defined. Then the first image from the folder is loaded into variable ‘image’ by calling the function load_image. The image is then viewed by using matplotlib.imshow. After this, the dimensions of the image, the maximum pixel value, and the minimum pixel value in the grayscale bar is printed. Also the mean and standard deviation of the image pixels are calculated.

Analyzing images using Python

Image name = person1000_bacteria_2931.jpeg

person1000_bacteria_2931.jpeg

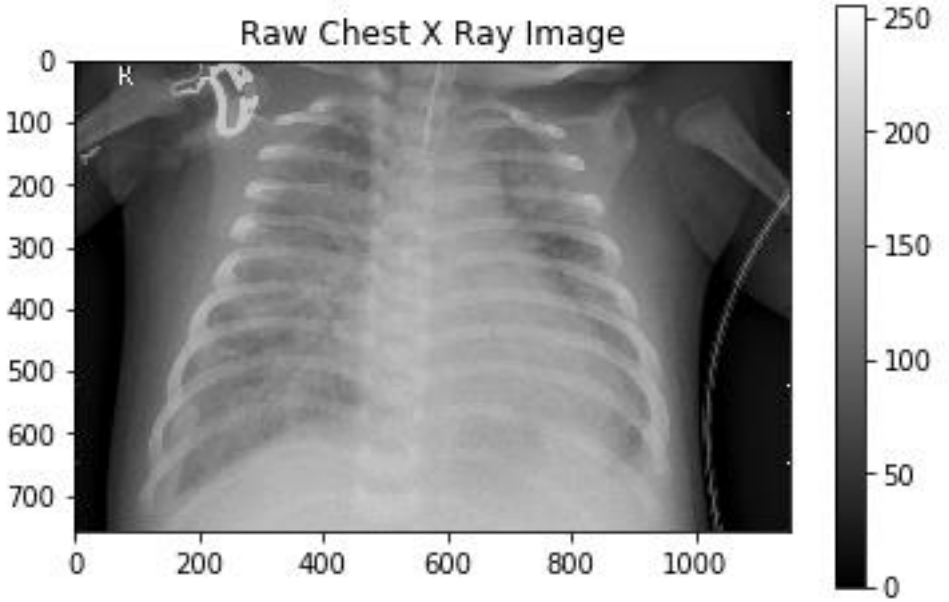
The dimensions are 760 pixels height and 1152 pixels width

The maximum pixel value is 255.0000

The minimum pixel value is 0.0000

The mean value of the pixels is 114.5373

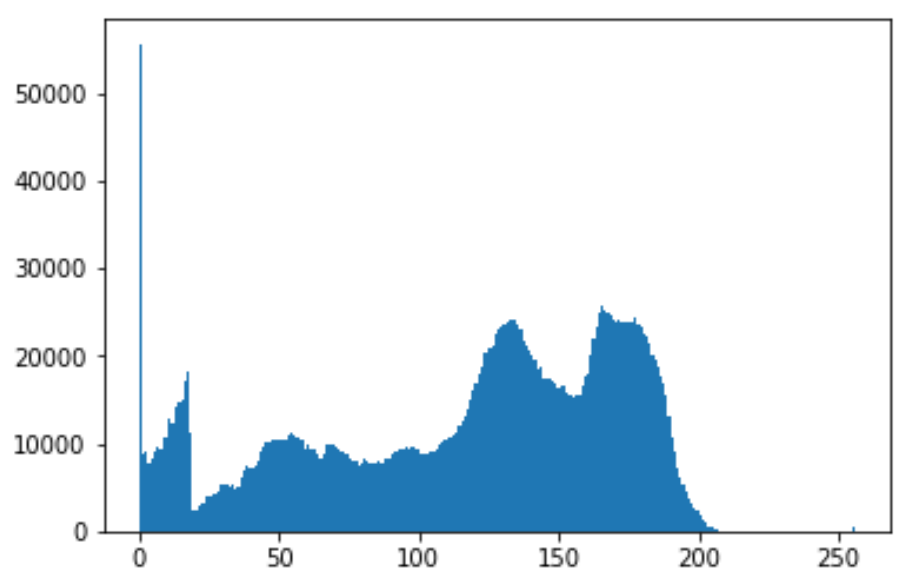
The standard deviation is 56.2341



Analyzing images using Python

```
# plot a histogram
plt.hist(image.ravel(),256,[0,256])
plt.show()
```

Next, we plot the histogram of all the pixels of the image. A histogram is a graphical display of data using bars of different heights.



Matplotlib.hist is used to plot the histogram. As the image is mostly dark, we see a huge cluster of pixels on position zero of the grayscale bar.

Loading images and labels together and resizing images

```
# loading the path of the train images


path = 'H://chest_xray/train/'
train = os.listdir(path)


folders=[]
folders = [f for f in sorted(os.listdir(path))]
print(folders)
```

The path of the training set is defined, and the directories under the path are saved in 'train'. In this case, there are two folders

['NORMAL', 'PNEUMONIA']

Loading images and labels together and resizing images

```
folders=[]
folders = [f for f in sorted(os.listdir(path))]
print(folders)


labels = folders
print (f'The labels are {labels}')


# setting the size of images that we want


image_size = 256
print(f'All images to be resized into {image_size}*{image_size} pixels')
```

We create an empty list — folders. Then, iterate over the path, using os.listdir, and sort and store the folder names in the list — 'folders'.
['NORMAL', 'PNEUMONIA']
The labels are ['NORMAL', 'PNEUMONIA']
All images to be resized into 256*256 pixels

Loading images and labels together and resizing images

If we go through the dataset, we see all the images are of varying dimensions, and to feed images into a Convolutional Neural Network (CNN) it is necessary to resize the images into the same dimensions.

Analyzing images using Python

```
# defining a function to load images and labels together
# this function will also resize the images

def load_train(path):
    images = []
    for label in labels:
        direc = os.path.join(path, label)
        class_num = labels.index(label)

        for image in os.listdir(direc):
            image_read = cv.imread(os.path.join(direc,image),cv.IMREAD_GRAYSCALE)
            image_resized = cv.resize(image_read,(image_size,image_size))
            images.append([image_resized,class_num])

    return np.array(images)
```

Here we define a function to load in all the images according to the label names, resize them into 256*256 pixels, and return the image arrays.

Loading images and labels together and resizing images

```
#load all the training images to train_images

train_images = load_train(path)

print(f'Shape of the training images = {train_images.shape}')
```

Shape of the training images = (5216, 2)

5,216 Images and 2 Labels: ['NORMAL', 'PNEUMONIA']

Loading images and labels together and resizing images

```
#loading the images and labels seperately in X and y, to be used later for training
X = []
y = []

for feature, label in train_images:
    X.append(feature)
    y.append(label)

print (f'Length of X = {len(X)}')
print (f'Length of y = {len(y)}')
```

```
Length of X = 5216
Length of y = 5216
```

check the number of images in each class

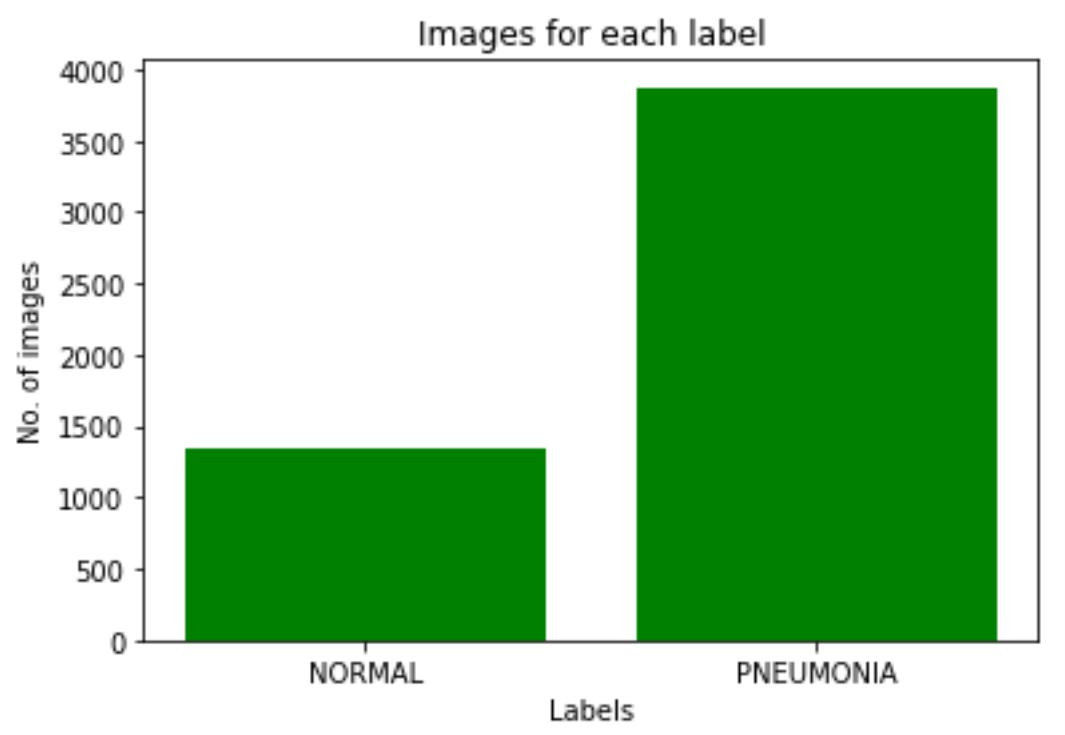
```
# checking the number of images of each class
a = 0
b = 0

for label in y:
    if label == 0:
        a += 1
    if label == 1:
        b += 1
```

check the number of images in each class

```
print (f'Number of Normal images = {a}')
print (f'Number of Bacteria images = {b}')
# plotting the data
x_pos = [i for i, _ in enumerate(labels)]
numbers = [a,b]
plt.bar(x_pos,numbers,color = 'green')
plt.xlabel("Labels")
plt.ylabel("No. of images")
plt.title("Images for each label")
plt.xticks(x_pos, labels)
plt.show()
```

```
Number of Normal images = 1341
Number of Bacteria images = 3875
```

check the number of images in each class

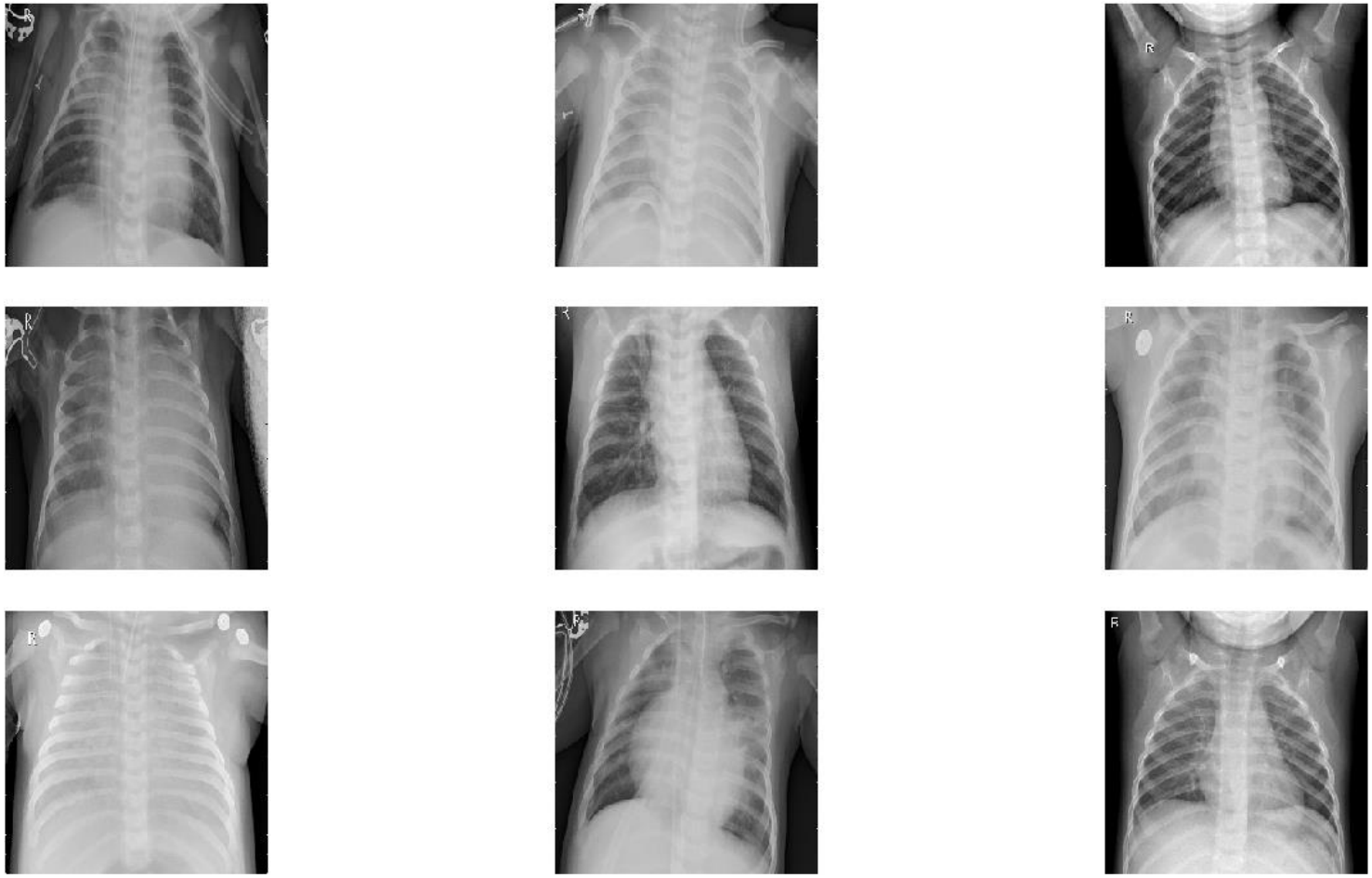


Extract 9 random images

```
# Displays images
# Extract 9 random images
print('Display Random Images')
# Adjust the size of your images
plt.figure(figsize=(20,10))
for i in range(9):
    num = random.randint(0,len(X)-1)
    plt.subplot(3, 3, i + 1)
    plt.imshow(X[num],cmap='gray')
    plt.axis('off')
# Adjust subplot parameters to give specified padding
plt.tight_layout()
```

Finally, we use the random module to generate nine random images from the training set and then used matplotlib to plot these images.

Extract 9 random images



Electrical Signal (ECG or EKG)

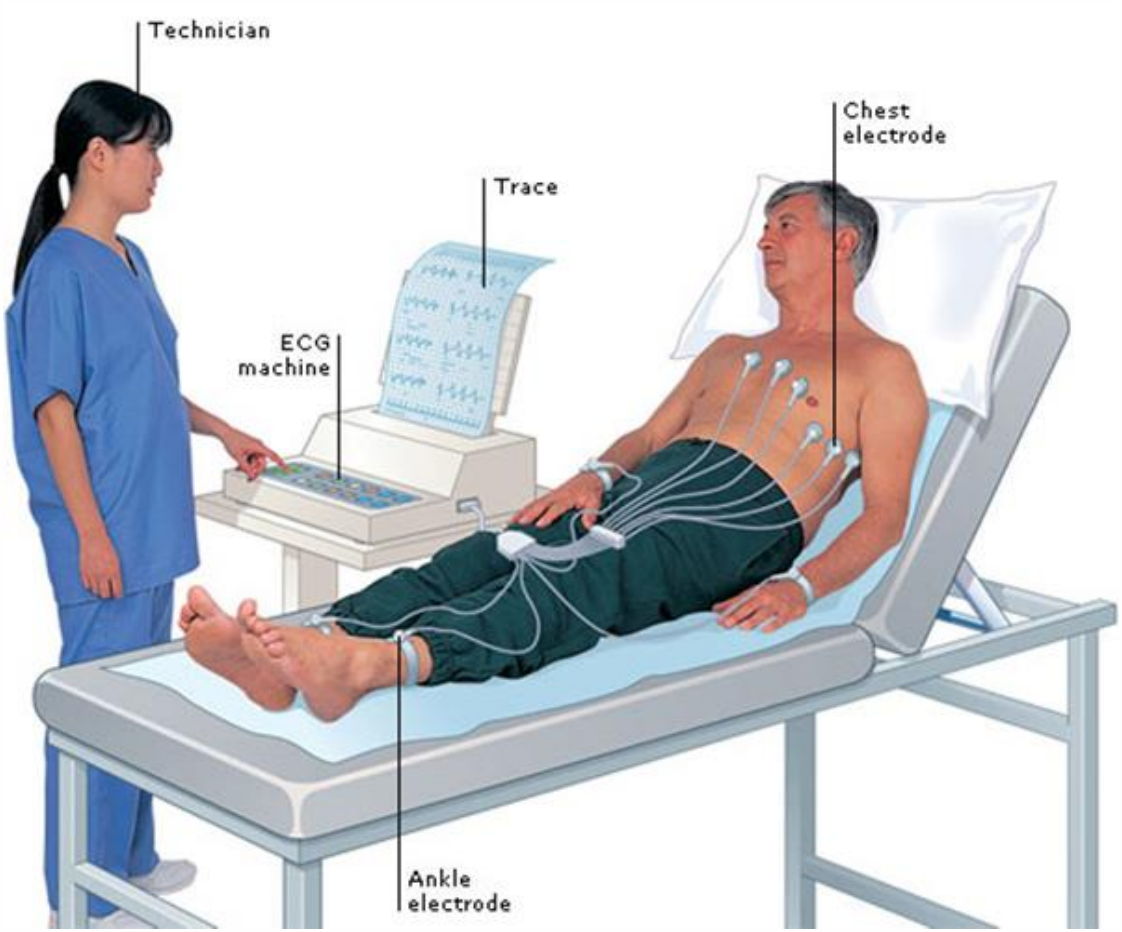
Analytic applications

What is ECG signal processing

Electrocardiogram (ECG or EKG) is a non-invasive test that records and displays the electrical activities produced by heart muscle during a cardiac cycle.

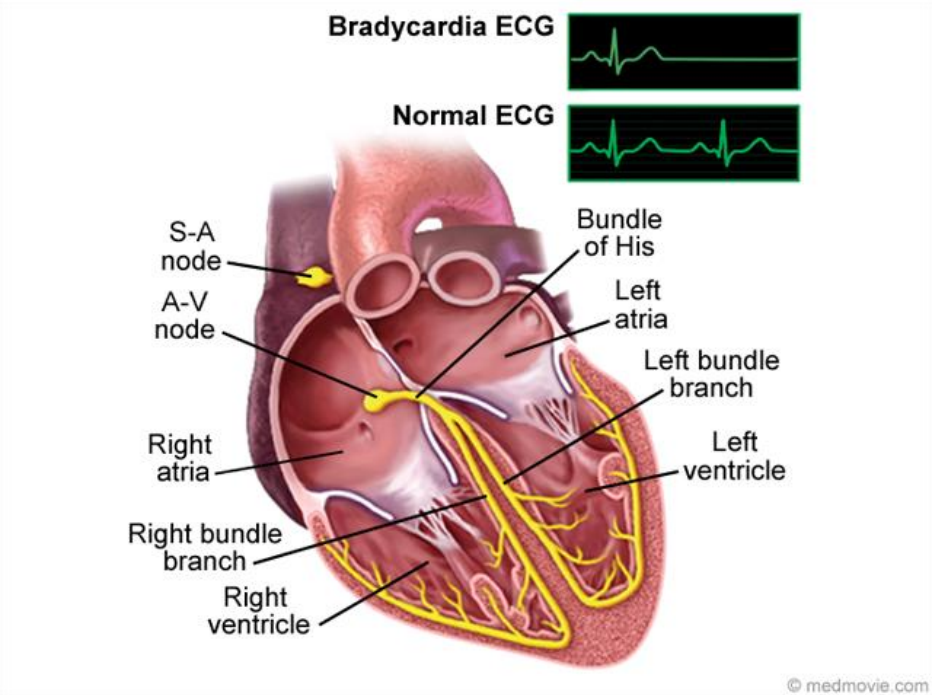
The ECG test is a standard clinical tool for diagnosing abnormal heart rhythms and to assess the general condition of a heart, such as myocardial infarctions, atrial enlargements, ventricular hypertrophies, and bundle branch blocks.

ECG signal processing

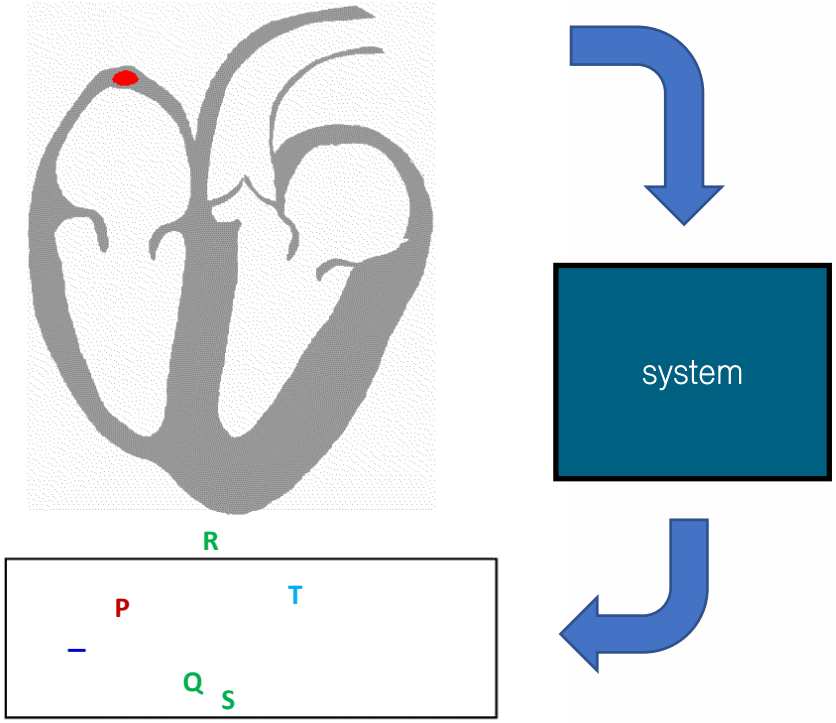


Electrical Signal

The electrocardiogram (ECG or EKG) is a diagnostic tool that is routinely used to assess the electrical and muscular functions of the heart



ECG signal processing



https://www.amperordirect.com/pc/help-ecg-monitor/z-interpreting-ecg.html

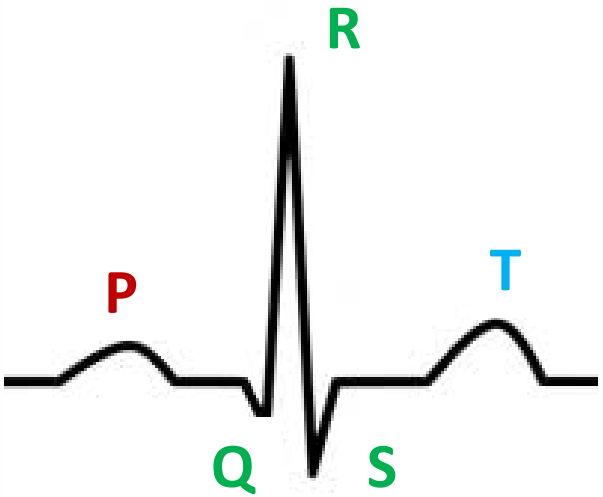
ECG cycle

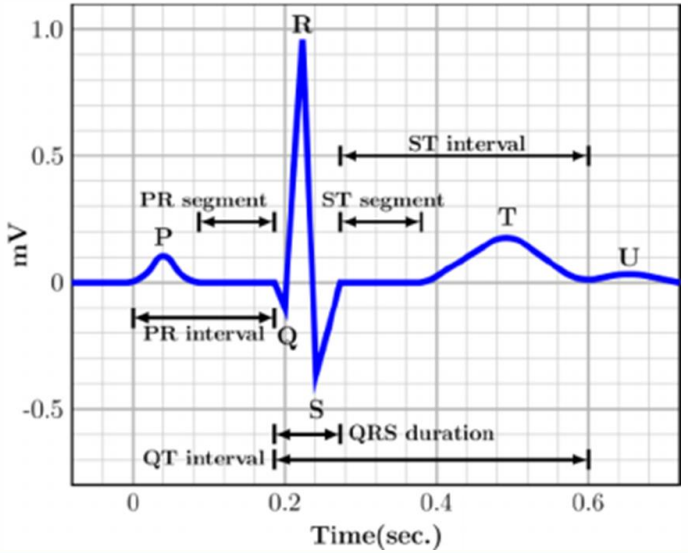
ECG cycle consists of 5 waves called PQRST

The P wave represents the normal atrial depolarization;

The QRS complex (one single heart beat) corresponds to the depolarization of the right and left ventricles;

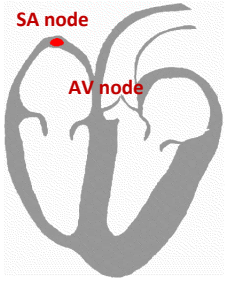
The T wave represents the re-polarization (or recovery) of the ventricles.

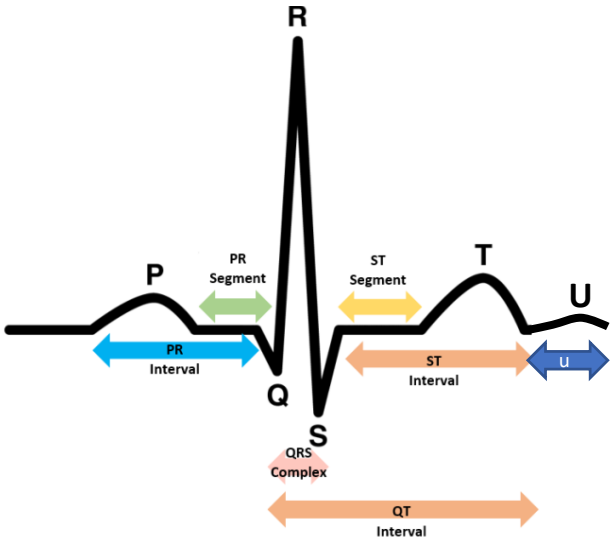




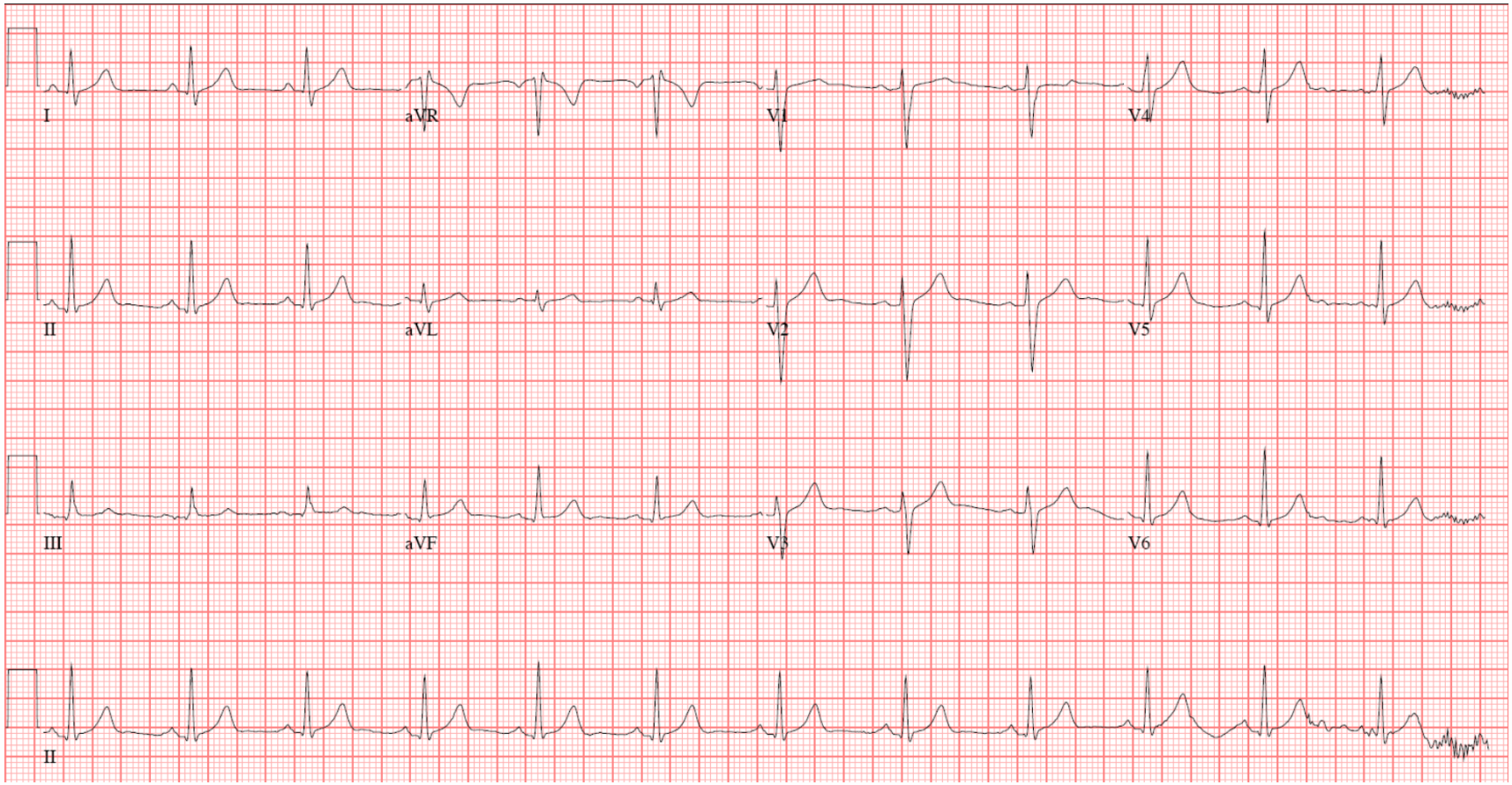
a depolarisation–repolarisation cycle

P Wave	Spread of electrical impulse across atria from SA node Atrial depolarisation
QRS Complex	Spread of electrical impulse through the ventricles Ventricular depolarisation
T Wave	Ventricular repolarisation
PR Interval	Electrical impulse spreads over atrium, through AV node and Bundle of His
ST Segment	Isoelectric point. No change in electrical activity
QT Interval	Depolarisation and repolarisation of the ventricles
U Wave	Hypothesised to be Purkinje repolarisation





Normal ECG



Normal ECG characteristics

Rhythm	<10% variation in RR intervals)
Rate	60- 99bpm
Cardiac Axis	-30° – 90°
P Waves	0.2-0.3mV
	0.06 – 0.12s
	Upright in I, II, aVF, V2- V6
	Inverted in aVR
	Varies in III, aVLSinus origin
PR Interval	0.12 – 0.2s
Q Waves	Small in I, II, aVL, V5, V6
QRS Complex	<0.12s

Normal ECG characteristics

ST Segment	Isoelectric
T Waves	<2/3 height of preceding R wave
	0.5mm in I, II, III
	<10mm in V1 – V6
	Same direction as preceding R wave
U Waves	<25% of T wave
	Same direction as T wave
QT_c	<440ms in males
	<460ms in females

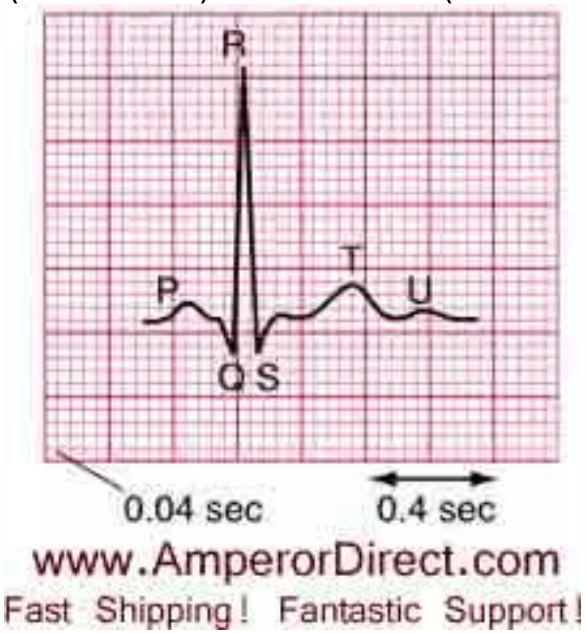
How to sensed health care signals

- The signals can be sensed by qualitative or quantitative manner
- Measurement
  - Scalar
  - Function of time
  - discrete x[n]
  - continuous x(f)
  - digital
  - Multivariant vector

How to sensed health care signals

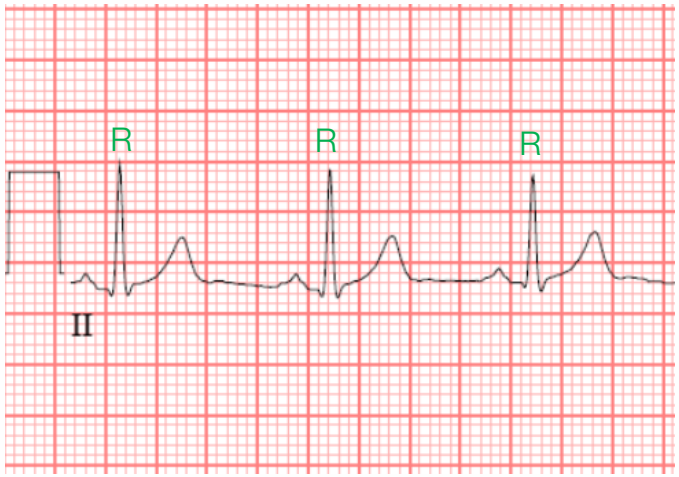
- ECG

ECG is recorded on standard paper typically running at 25 mm/sec and is measuring millivolts versus time. Each big box is 5 mm square and corresponds to 200ms (horizontal) and 0.5mV (vertical).



How to calculate heart rate

- 1500/Count the number of small squares between two R waves



Example

- 20 small squares between first and second R waves
- 1500/20 = 75
- **Heart rate** = 75 beats per minute (bpm)

Mahidol University
Faculty of Medicine Ramathibodi Hospital
Department of Clinical Epidemiology and Biostatistics

Introduction to ECG signal processing with Python

ECG signal processing with python

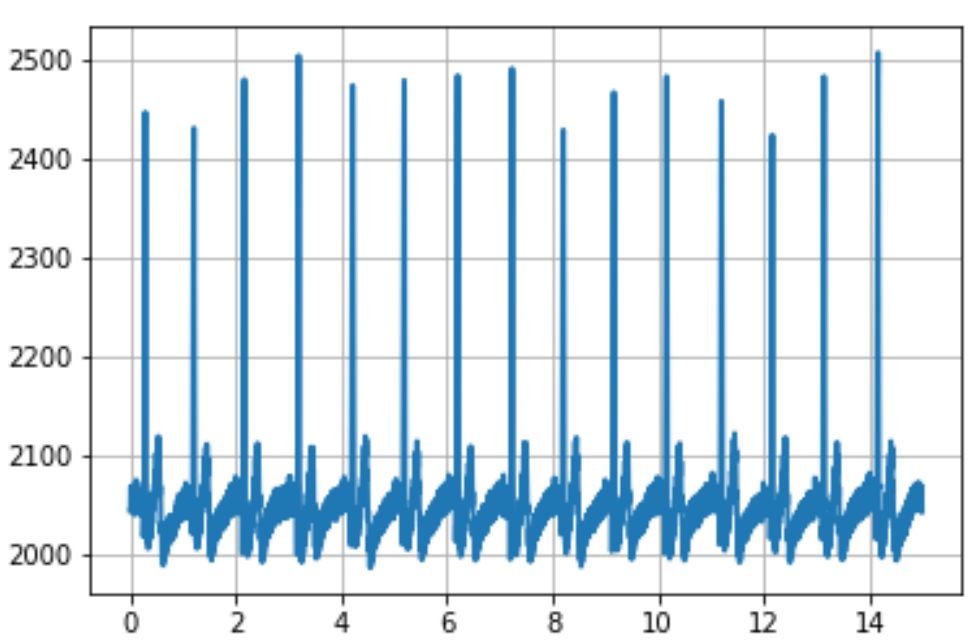
Install package

- pip install biosppy
- conda install h5py
- pip install shortuuid
- pip install -U scikit-learn
- conda install –c conda-forge imageio

Plot raw signal ECG

```
import numpy as np
import pylab as pl
from biosppy import storage
signal, mdata = storage.load_txt('H:/signal-processing/ecg.txt')
Fs = mdata['sampling_rate']
N = len(signal)  # number of samples
T = (N - 1) / Fs  # duration
ts = np.linspace(0, T, N, endpoint=False)  # relative timestamps
pl.plot(ts, signal, lw=2)
pl.grid()
pl.show()
```

Plot raw signal ECG

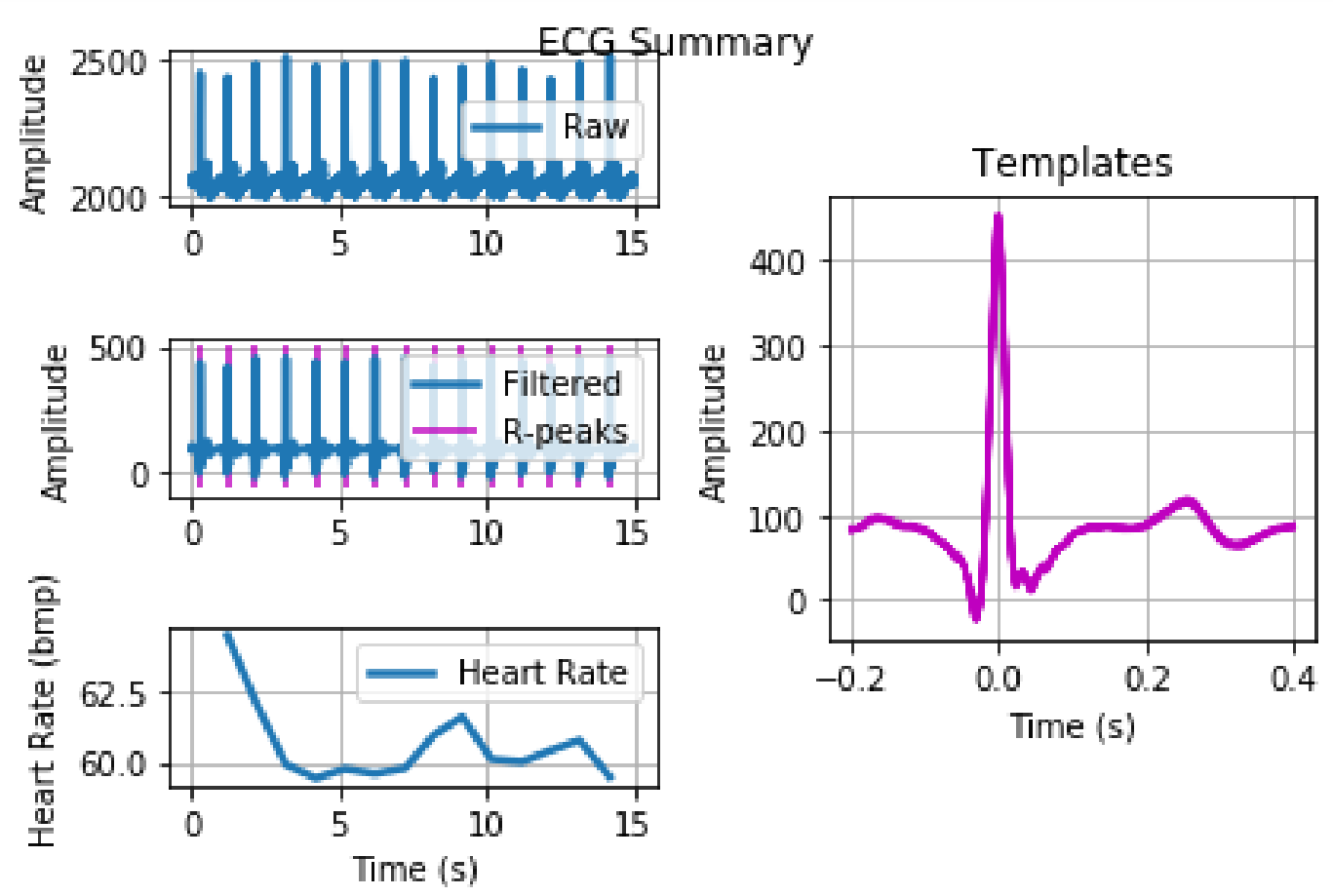


This signal is a Lead I ECG signal acquired at 1000 Hz, with a resolution of 12 bit. Although of good quality, it exhibits powerline noise interference, has a DC offset resulting from the acquisition device, and we can also observe the influence of breathing in the variability of R-peak amplitudes.

Plot summary ECG with python

```
# process it and plot
from biosppy.signals import ecg
out = ecg.ecg(signal=signal, sampling_rate=1000., show=True)
```

Plot summary ECG with python



Example of Biosignals Data

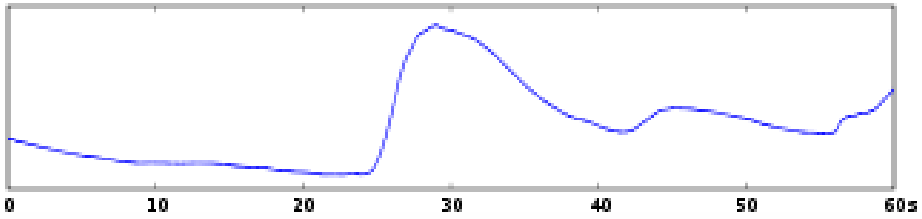
Example of Biosignals Data

Include:

- ECG signals
- Electrodermal Activity (EDA) signals
- Photosensor
- Respiration (RSP) equipment
- Etc.

Example of Biosignals Data

Electrodermal Activity (EDA) signals is the property of the human body that causes continuous variation in the electrical characteristics of the skin. Historically, EDA has also been known as skin conductance, galvanic skin response (GSR), electrodermal response (EDR), psychogalvanic reflex (PGR)

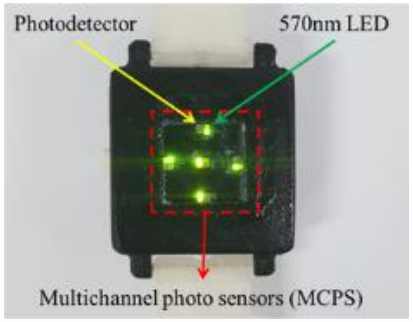


Example of Biosignals Data

Photosensor is a type of electronic component that enables the detection of light, infrared and other forms of electromagnetic energy.

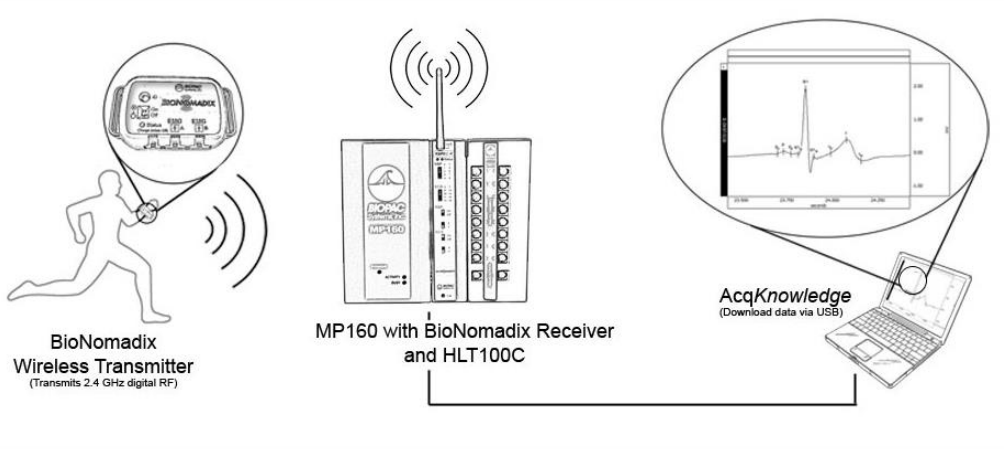
It is used in electronic and computing devices to receive input and/or transmit data in the form of light or electromagnetic signals.

Photosensors are also known as photodetctors.



Example of Biosignals Data

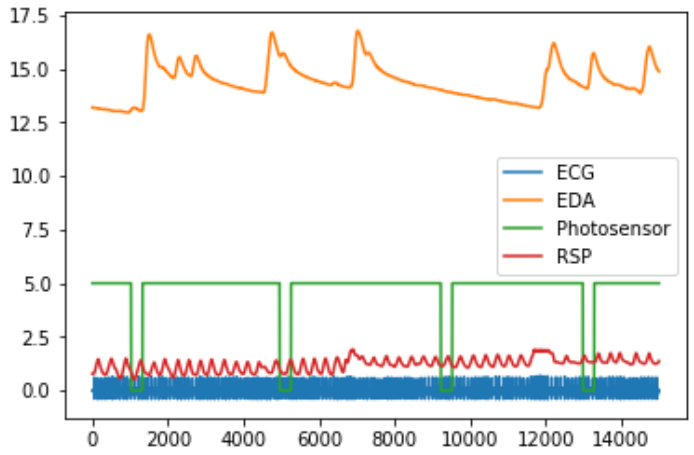
Respiration (RSP) equipment is designed for measure abdominal or thoracic expansion and contraction while breathing



Example of Biosignals Data

```
import numpy as np
import pandas as pd
df = pd.read_csv('H:/signal-processing//bio_100Hz.csv')
# Plot it
df.plot()
```

df contains 2.5 minutes of data recorded at 100Hz (2.5 x 60 x 100 = 15000 data points). There are 4 channels, EDA, ECG, RSP and the Photosensor used to localize events

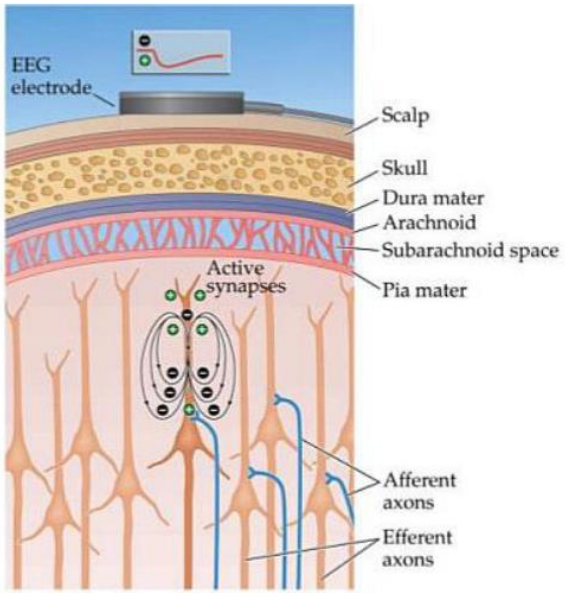


Signal processing : EEG

https://mne.tools/dev/auto_tutorials/intro/plot_10_overview.html

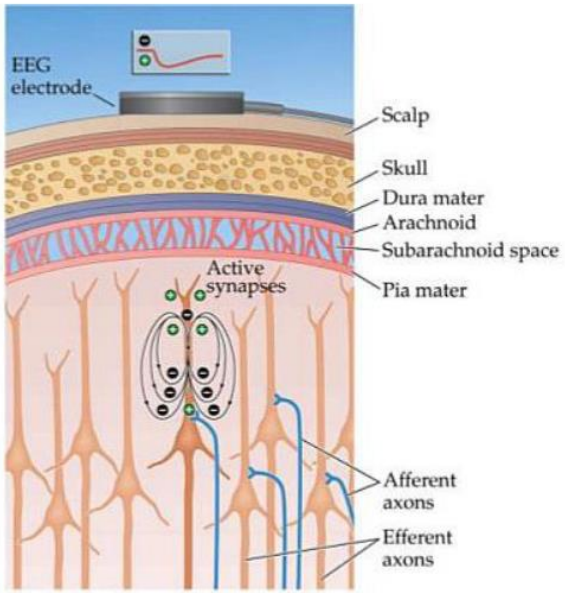
Electroencephalography (EEG)

EEG is first used with a human in 1929 by Hans Berge. It is a noninvasive method to record brain electricity at the cerebral cortex, and the result can help predict many neurologic conditions.



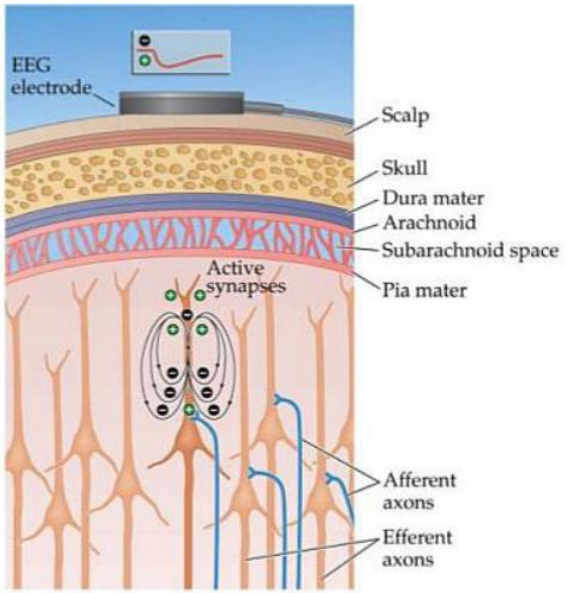
Electroencephalography (EEG)

The brain voltage is recorded by metal electrodes and presents as the summation of postsynaptic potentials (i.e., excitatory and inhibitory) from relatively large groups of cortical pyramidal neurons.



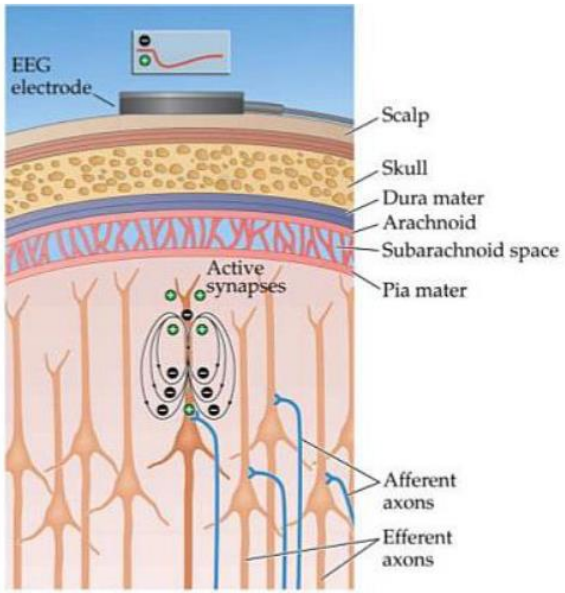
Electroencephalography (EEG)

The signals are mostly measured in a millisecond (msec.) as it is equal to action potential scale (i.e., a process to generate brain ion waves).



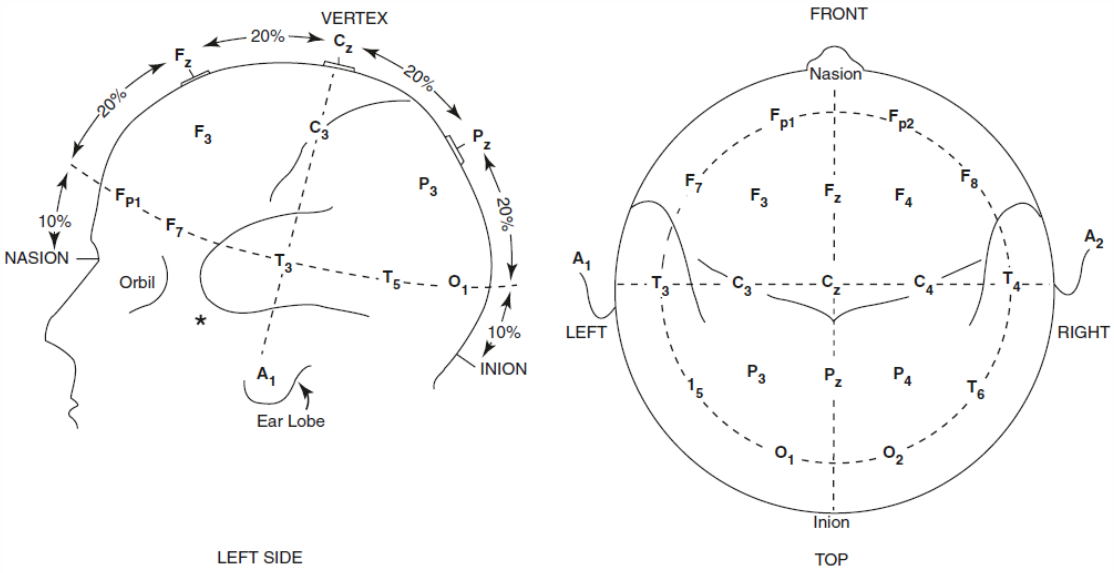
Electroencephalography (EEG)

The signals are mostly measured in a millisecond (msec.) as it is equal to action potential scale (i.e., a process to generate brain ion waves).



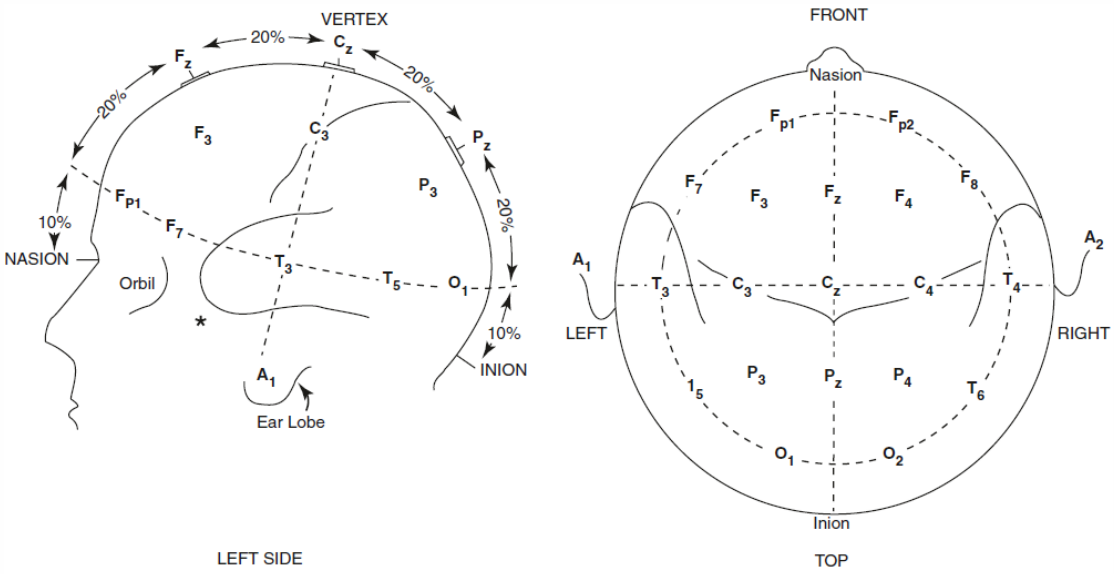
Electroencephalography (EEG)

The international standard of EEG electrode localization is the 10-20 system. The standard guideline in adults includes 21 scalp and one ground electrodes. However, there could be additional electrodes for recording eye movements, heart activities, muscle activities, or photic stimulation.



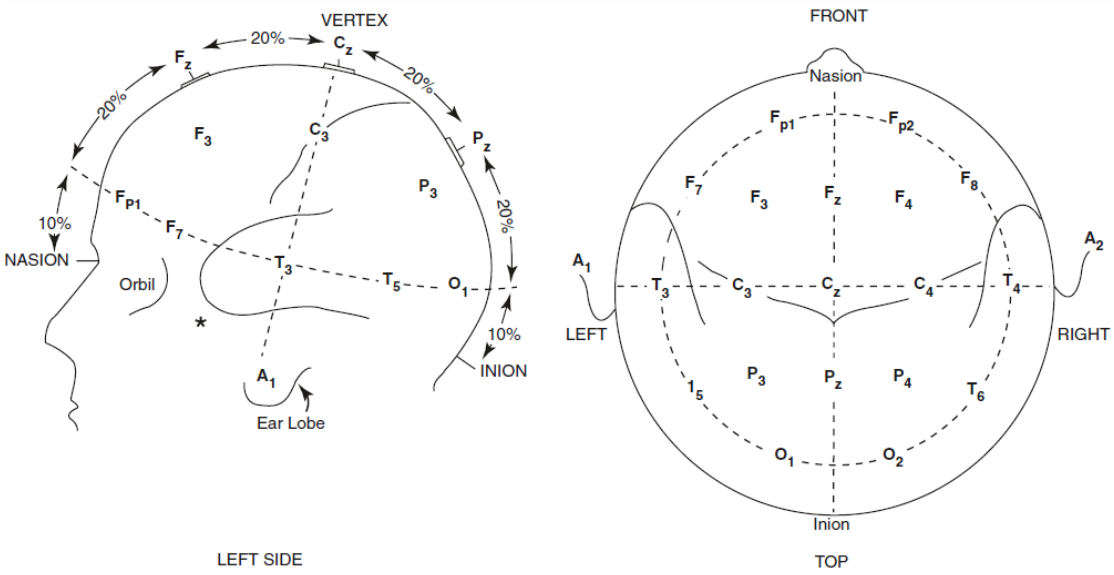
Electroencephalography (EEG)

All electrodes placements rely on the cerebral cortex with respect to skull shape and size. The brain is separated into four regions by lobe, including frontal, temporal, parietal, and occipital.



Electroencephalography (EEG)

Four main landmarks are 10% adjacent distance from (1) front to back skull (nasion), (2) posterior to anterior skull (inion), (3) left to right skull (left ear lobe), and (4) right to the left skull (right ear lobe). Other electrodes locate at 20% of such distances. Each electrode has a label that indicates its location by brain lobe (first digit) and side.



EEG analysis

EEG analysis is exploiting mathematical signal analysis methods and computer technology to extract information from electroencephalography (EEG) signals.

The targets of EEG analysis are to help researchers gain a better understanding of the brain; assist physicians in diagnosis and treatment choices; and to boost brain-computer interface (BCI) technology.

EEG analysis

There are many ways to roughly categorize EEG analysis methods. If a mathematical model is exploited to fit the sampled EEG signals, the method can be categorized as parametric, otherwise, it is a non-parametric method.

Traditionally, most EEG analysis methods fall into four categories: time domain, frequency domain, time-frequency domain, and nonlinear methods.

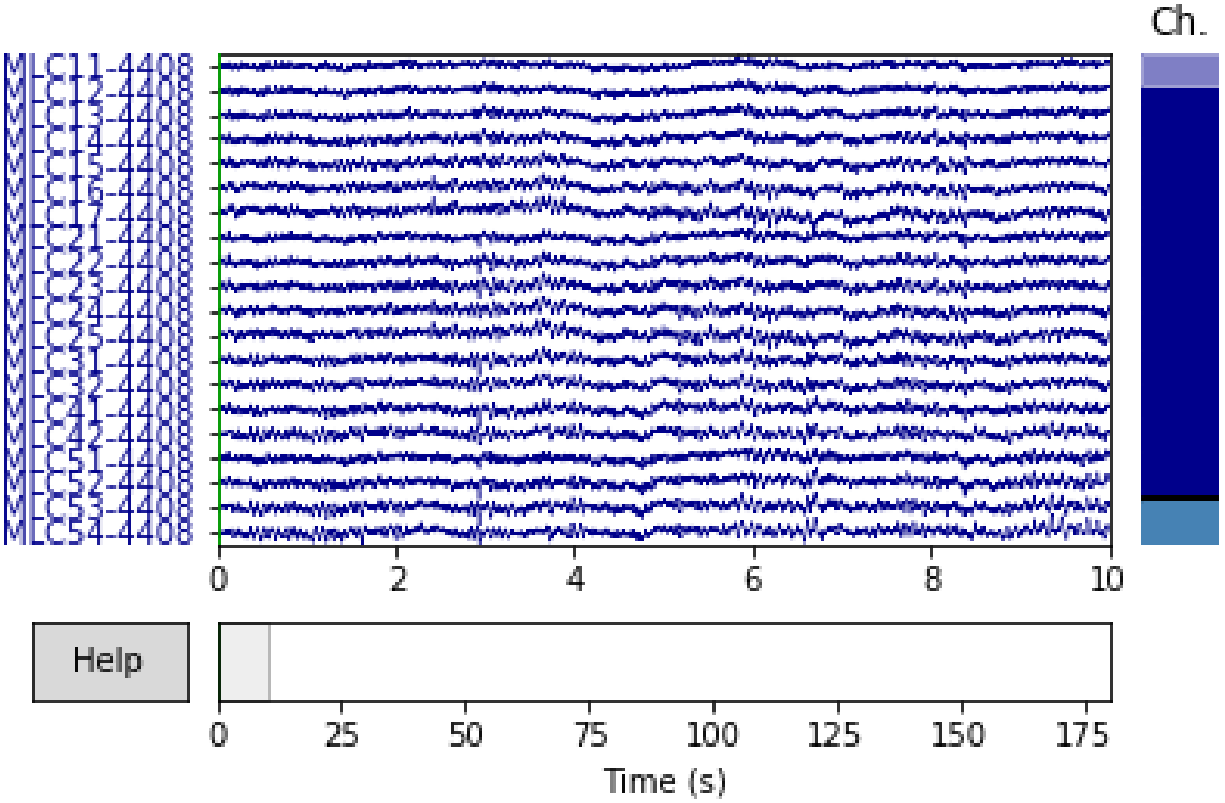
EEG analysis with MNE-Python

https://mne.tools/dev/auto_examples/datasets/plot_brainstorm_data.html#ex-brainstorm-raw

EEG analysis with MNE-Python

```
import numpy as np
import mne
from mne.datasets.brainstorm import bst_raw
from mne.io import read_raw_ctf
print(__doc__)
tmin, tmax, event_id = -0.1, 0.3, 2  # take right-hand somato
reject = dict(mag=4e-12, eog=250e-6)
data_path = bst_raw.data_path()

raw_path = (data_path + '/MEG/bst_raw/' +
        'subj001_somatosensory_20111109_01_AUX-f.ds')
# Here we crop to half the length to save memory
raw = read_raw_ctf(raw_path).crop(0, 180).load_data()
raw.plot()
```

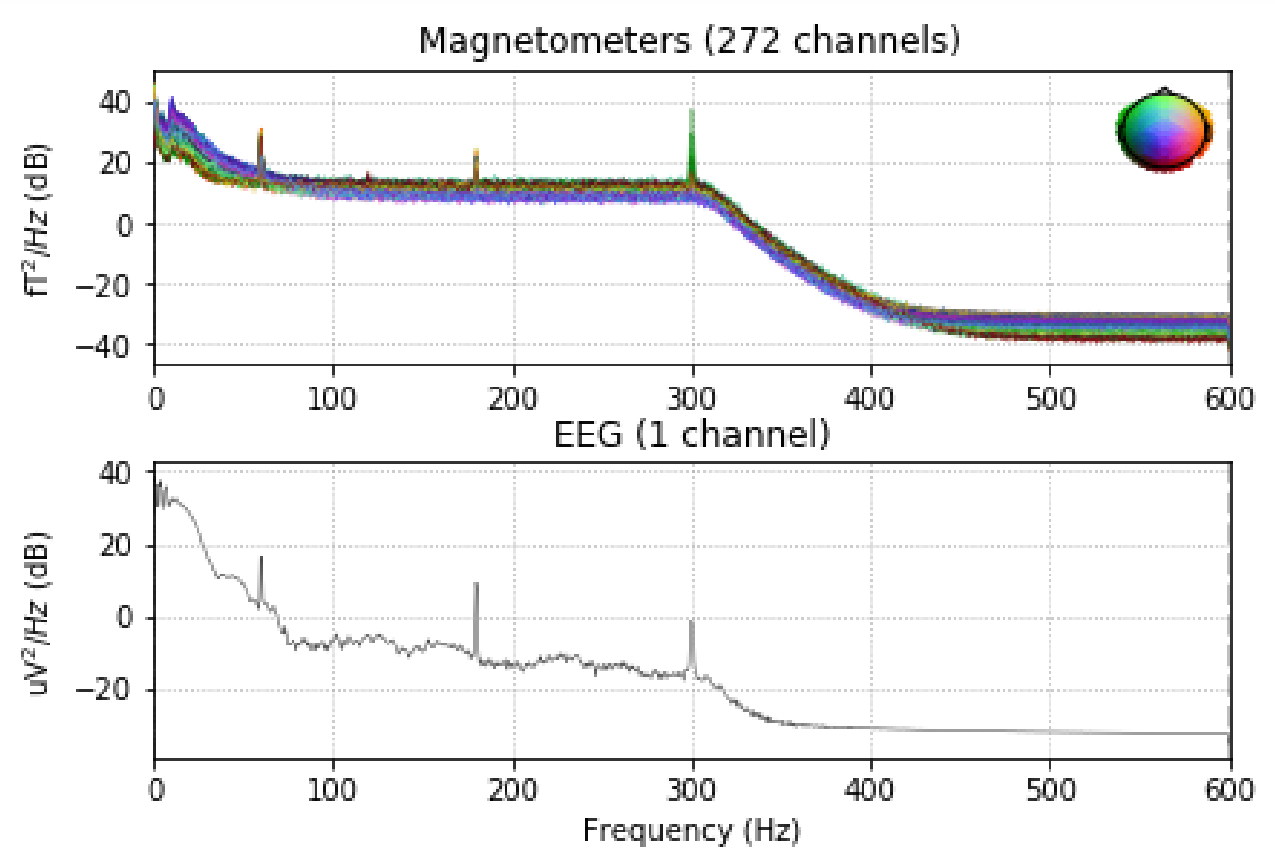
EEG analysis with MNE-Python



EEG analysis with MNE-Python

```
# set EOG channel
raw.set_channel_types({'EEG058': 'eog'})
raw.set_eeg_reference('average', projection=True)

# show power line interference and remove it
raw.plot_psd(tmax=60., average=False)
raw.notch_filter(np.arange(60, 181, 60), fir_design='firwin')

events = mne.find_events(raw, stim_channel='UPPT001')
```

EEG analysis with MNE-Python



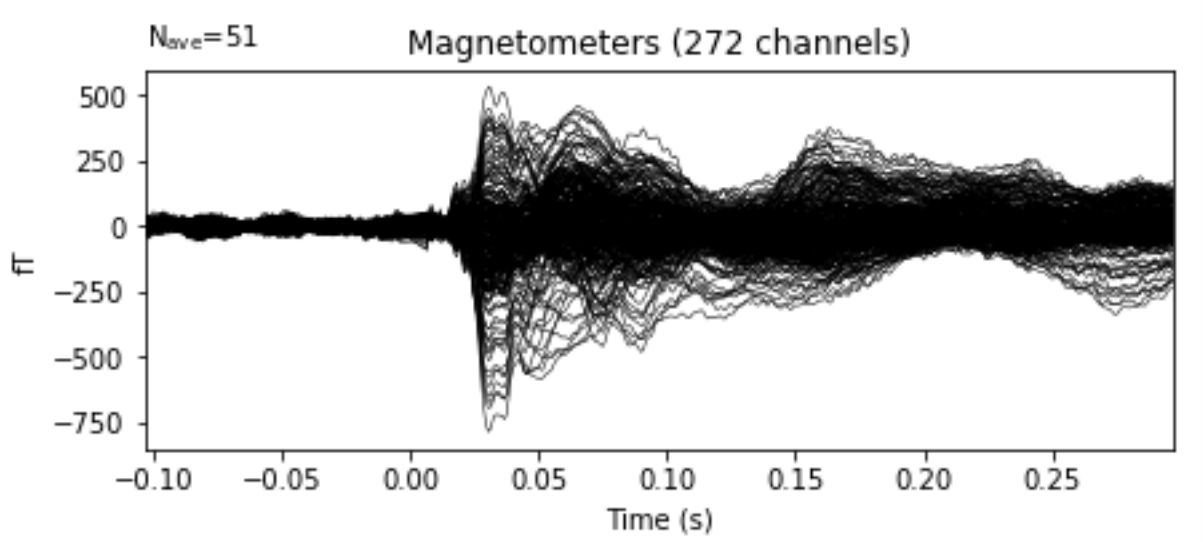
EEG analysis with MNE-Python

```
# pick MEG channels
picks = mne.pick_types(raw.info, meg=True, eeg=False, stim=False, eog=True,
                       exclude='bads')
# Compute epochs
epochs = mne.Epochs(raw, events, event_id, tmin, tmax, picks=picks, baseline=(None, 0), reject=reject, preload=False)

# compute evoked
evoked = epochs.average()

# remove physiological artifacts (eyeblinks, heartbeats) using SSP on baseline
evoked.add_proj(mne.compute_proj_evoked(evoked.copy().crop(tmax=0)))
evoked.apply_proj()

# fix stim artifact
mne.preprocessing.fix_stim_artifact(evoked)
```

EEG analysis with MNE-Python

```
# correct delays due to hardware (stim artifact is at 4 ms)
evoked.shift_time(-0.004)

# plot the result
evoked.plot(time_unit='s')

# show topomaps (EEG topographic maps)
evoked.plot_topomap(times=np.array([0.016, 0.030, 0.060, 0.070]), time_unit='s')
```

EEG analysis with MNE-Python



EEG analysis with MNE-Python