université PARIS-SACLAY

UC 2A - LE MACHINE LEARNING EN PRATIQUE

INTRODUCTION AUX OUTILS (NUMPY, SCIKIT-LEARN...)

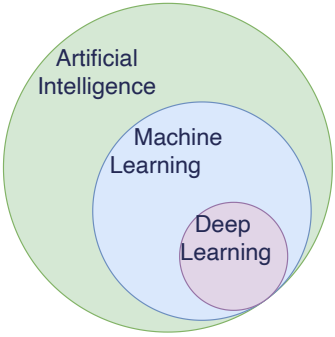
Vincent Guigue, Romain Thoreau
vincent.guigue@agroparistech.fr

MIA PARIS-SACLAY
EKINOCS

AGROPARISTECH
Institut des Sciences et Industries du Vivant et de l'Environnement

Intelligence Artificielle, Machine Learning et Programmation

Input (**X**)	Output (**Y**)	Application
email	spam? (0/1)	spam filtering
audio	text transcript	speech recognition
English	Chinese	machine translation
ad, user info	click? (0/1)	online advertising
image, radar info	position of other cars	self-driving car
image of phone	defect? (0/1)	visual inspection



IA : programmes informatiques qui s'adonnent à des tâches qui sont, pour l'instant, accomplies de façon plus satisfaisante par des êtres humains car elles demandent des processus mentaux de haut niveau.

Marvin Lee Minsky, 1956

N-AI (Narrow Artificial Intelligence), dédiée à une tâche

$\neq$ **G-AI (General AI)** qui remplace l'humain dans des systèmes complexes.

Andrew Ng, 2015

Ingrédients de l'Intelligence Artificielle

www

Données Capteurs

Cadre logiciel

Stockage & Calcul

Modèles

python jupyter

NVIDIA CUDA

PyTorch

TensorFlow

APACHE Spark

scikit learn

GitHub

amazon web services

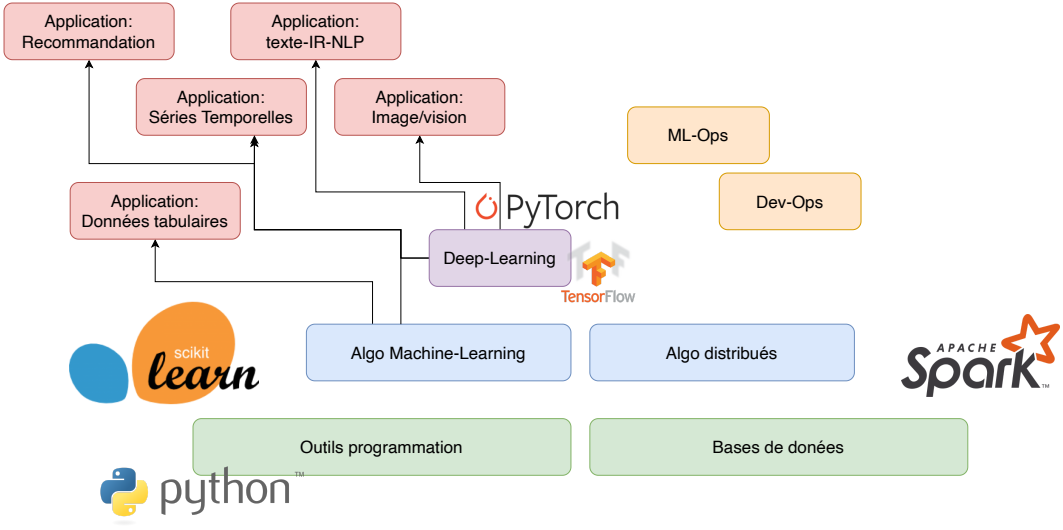
Microsoft Azure

Google Cloud

NVIDIA

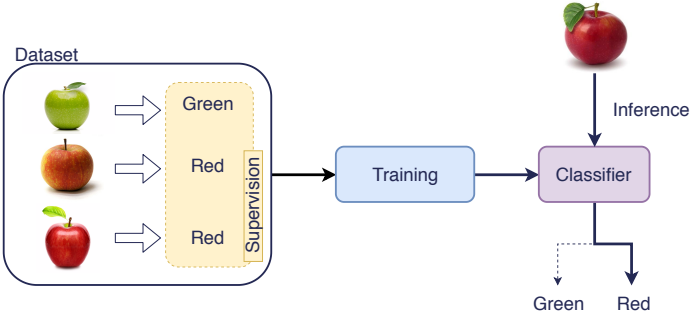
Enseignement de l'IA

- Différents niveaux d'accès
- Différentes branches: types d'outils, application thématiques, ...



Programmation *orientée données*

- **Python** : langage unificateur (codage *vs* wrapper)
  - *Calcul scientifique* : `numpy`
  - *Machine-learning*: `scikit-learn`, `pandas`, `matplotlib`
  - *Deep-learning*: `pytorch`
  - Environnement de développement: Visual Studio Code / jupyter-notebook

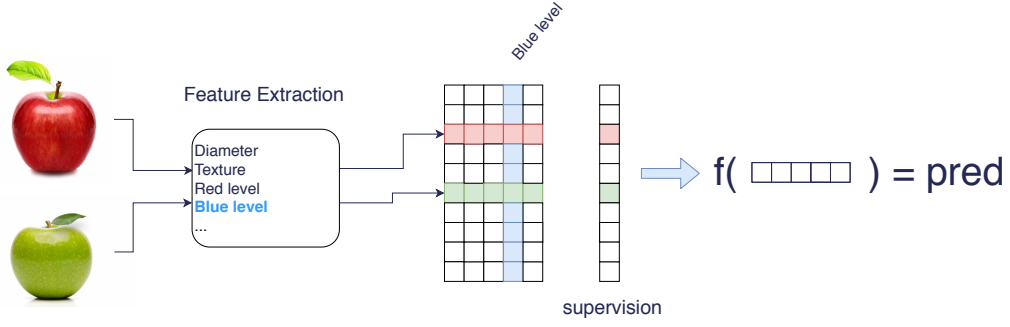


Où se trouve les leviers de performance?

Dans les modèles...
Mais surtout dans les chaînes de traitements !

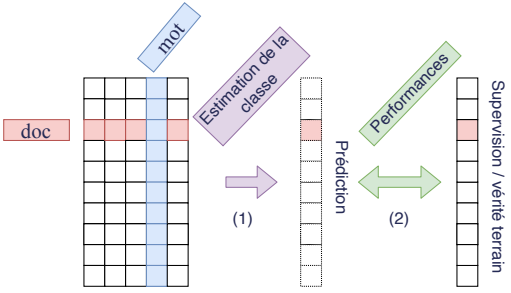
Programmation *orientée données*

- **Python** : langage unificateur (codage *vs* wrapper)
  - *Calcul scientifique* : `numpy`
  - *Machine-learning*: `scikit-learn`, `pandas`, `matplotlib`
  - *Deep-learning*: `pytorch`
  - Environnement de développement: Visual Studio Code / jupyter-notebook



Programmation *orientée données*

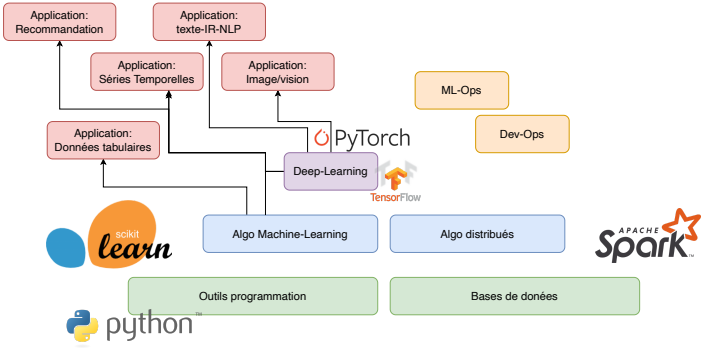
- **Python** : langage unificateur (codage *vs* wrapper)
  - *Calcul scientifique* : `numpy`
  - *Machine-learning*: `scikit-learn`, `pandas`, `matplotlib`
  - *Deep-learning*: `pytorch`
  - Environnement de développement: Visual Studio Code / jupyter-notebook



Organisation

Organisation

- **2 séances - Numpy (=5h)**
  - Mise à niveau en python, `numpy`, `matplotlib`
  - Naïve Bayes (à la main)
- **6 séances - Scikit-Learn (=6x3h30)**
  - Classifieurs Scikit-Learn : syntaxe, possibilités offertes
  - Apprentissage supervisé et non-supervisé
  - Évaluation
  - Chaine de traitements, sélection de modèles (grid-search...)
  - Visualisation & post-traitements

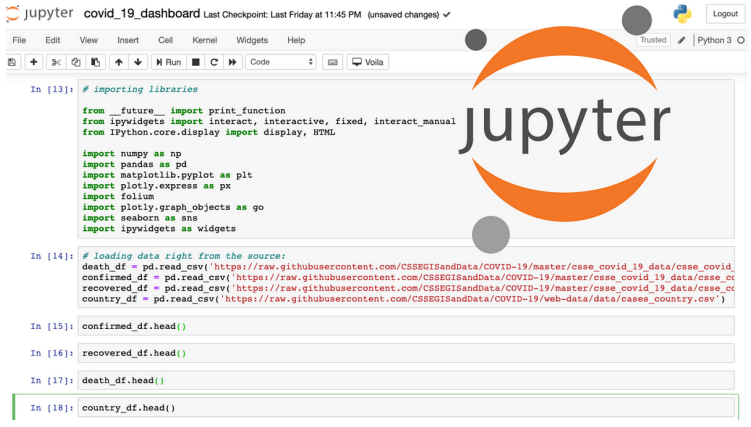


- **6 séances - Bouclage & projet (≈5.5x3h30)**

Jupyter Notebook

- Du code dans un navigateur web????
  - Principe de textes à trous
  - Bel outil pédagogique...
  - ... avec des risques (contemplation)

et des limites (organisation de code sous-optimale)



Conclusion : passer à un nouveau langage...

- **Cout faible**
  - une fois que vous avez compris la logique générale
- **Cout non négligeable:**
  - Comprendre les forces et les faiblesses du langage
    - ... Et des environnements de développement
  - Adapter sa manière de programmer (e.g. calculer un décile)
  - Reprendre les bons reflexes (=aller vite)

⇒ Devenir *data-scientist* n'a jamais été aussi facile... Mais il reste quelques savoir-faire et quelques pièges à éviter!