Comment installer la bilbiothèque :

pip3 install scikit-learn

Dès que vous souhaitez mettre en place un apprentissage automatisé, vous avez besoin de manipuler plusieurs ensembles. Vous allez découper votre ensemble labellisé en plusieurs sous-ensembles :

* un ensemble d'entraînement pour entraîner vos modèles de machine learning ;
* un ensemble de test pour mesurer la performance de vos modèles sur des données non apprises. La performance d'un modèle sur l'ensemble de test correspond à une mesure de ce modèle sur des données réelles, qui permettent d'évaluer la capacité de généralisation du modèle ;
* vous pouvez éventuellement utiliser un ensemble de validation pour déterminer les meilleurs hyper-paramètres de vos modèles. Vous allez ainsi rechercher le meilleur paramétrage de vos modèles sans pour autant vous servir de l'ensemble de test.

Le découpage d'un dataset en ensemble d'entraînement et ensemble de test se fait par Scikit-learn:

from sklearn.model\_selection import train\_test\_split

# L'ensemble de tests aura 20 % des éléments de départ.

# L'ensemble d'entraînement contiendra les 80 % restants.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2)

C'est pourquoi on choisira souvent d'utiliser une validation croisée pour affiner les hyper-paramètres d'un modèle. La validation croisée consiste en une suite de découpages distincts en un ensemble d'entraînement et un ensemble de validation. Chaque découpage permet de mesurer une performance du modèle et la moyenne des performances de chaque découpage permet de trouver les meilleurs hyper-paramètres sans jamais avoir validé le modèle sur une donnée déjà apprise.

from sklearn.model\_selection import cross\_val\_score

model = ...

scores = cross\_val\_score(model, X, Y, cv=5) #cv est le nombre de découpages à réaliser

score = scores.mean()

À part avec beaucoup d'intuition, il n'est pas possible de deviner les meilleurs hyper-paramètres pour un ensemble d'entraînement donné. Scikit-learn propose deux méthodes de recherche :

* GridSearchCV : qui va tester le modèle avec toutes les combinaisons d'hyper-paramètres, pour trouver les meilleures valeurs ;
* RandomizedSearchCV, qui va tester au hasard quelques combinaisons d'hyper-paramètres, pour trouver les meilleures.

Le principe est de définir un dictionnaire de paramètres et de valeurs à tester :

from sklearn.model\_selection import GridSearchCV

parameters = {"n\_neighbors": [2,3,4,5]}

model = GridSearchCV(KNeighborsClassifier(), parameters)

model.fit(...)