  Énoncés des exercices

#### Contexte

Notre objectif sera d'entraîner un modèle qui sera capable de reconnaître les chiffres écrits sur ce type d'images.

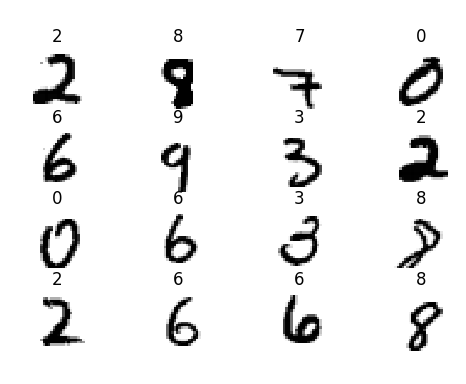
Par chance, ce jeu de données est téléchargeable directement à partir d'une fonction scikit-learn. On peut donc directement obtenir ce dataset via un appel de fonction :

from sklearn.datasets import fetch\_openml

mnist = fetch\_openml('mnist\_784', version=1)

#### Les données et la problématique

D'abord, parlons du jeu de données que nous allons utiliser. C'est un dataset très célèbre, appelé MNIST. Il est constitué d'un ensemble de 70000 images 28x28 pixels en noir et blanc annotées du chiffre correspondant (entre 0 et 9). L'objectif de ce jeu de données était de permettre à un ordinateur d'apprendre à reconnaître des nombres manuscrits automatiquement (pour lire des chèques par exemple). Ce dataset utilise des données réelles qui ont déjà été pré-traitées pour être plus facilement utilisables par un algorithme.

****

L'objet  mnist contient deux entrées principales,  data  et  target

* data contient les images sous forme de tableaux de 28 x 28 = 784 couleurs de pixel en niveau de gris, c'est-à-dire que la couleur de chaque pixel est représentée par un nombre entre 0 et 16 qui représente si celle-ci est proche du noir ou pas (0 = blanc, 16 = noir).
* target qui contient les annotations (de 1 à 9) correspondant à la valeur "lue" du chiffre.

1. Le dataset est relativement petit mais, pour le modèle k-NN, il est déjà trop gros pour obtenir rapidement des résultats. On va donc effectuer un sampling et travailler sur seulement 5000 données

Faite une restriction aleatoire des données, pour obtenir une DataFrame de 5000 données

1. Une fois notre dataset chargé, comme nous l'avons vu dans le TP précédent, nous allons séparer le jeu de données en training set et testing set.

On peut créer un premier classifieur 3-NN, c'est-à-dire qui prend en compte les 3 plus proches voisins pour la classification. Pour cela, on va utiliser l'implémentation de l'algorithme qui existe dans la librairie scikit-learn :

from sklearn import neighbors

knn = neighbors.KNeighborsClassifier(n\_neighbors=3)

knn.fit(xtrain, ytrain)

1. Testons à présent l’erreur de notre classifieur. La méthode  score  effectue exactement ça : tester les performances de prédiction d'un classifieur dans lequel on passe un jeu de données annoté.

#### Optimisation du score sur les données test

1. Pour trouver le k optimal, on va simplement tester le modèle pour tous les k de 2 à 20, mesurer l’erreur test et afficher la performance en fonction de k.
2. Tracez une courbe qui représente les erreurs des diffèrent modelés.
3. afficher les prédictions du classifieur sur quelques données.