

## TP n°1: Introduction et manipulation de la base de données

*Le compte-rendu doit être rédigé en Jupyter-Notebook, Markdown ou LaTeX. Il doit comporter à la fois les codes mais également **et surtout** des commentaires et interprétations clairs et pertinents concernant (i) vos choix, (ii) vos résultats en rapport avec les notions vues en cours.*

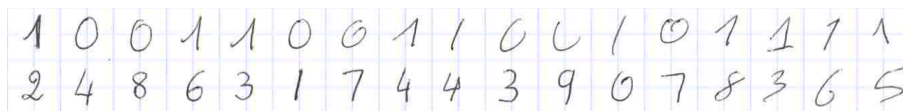
### 1 Présentation du Projet

L'ensemble des TP forme la base d'un projet long, de type "compétition Kaggle". L'objectif est en effet, au terme de l'UE, de résoudre le problème d'intelligence artificiel décrit ci-après.

#### 1.1 Objectif du projet

L'objectif du projet est le suivant: à partir du scan d'un document papier contenant une suite de chiffres manuscrits (de 0 à 9), vous allez développer un algorithme d'intelligence artificielle capable:

- de localiser la position des chiffres
- de reconnaître chacun des chiffres.



L'outil que vous allez développer vous permettra alors de créer vous-mêmes un lecteur automatique de chèques sur lesquels on essaiera de récupérer un maximum d'informations.

**BANQUE**  
 Payez contre ce chèque non endossable  
 Sauf au profit d'une banque ou d'un établissement assimilé

cent dix mille cent euros  
 somme en toutes lettres

A M. le professeur

CE-060809 Payable en France  
 15236 000036  
 PARIS 15ème  
 18 RUE DE LA BANQUE  
 75 015 PARIS  
 TEL : 08 36 22 12 18

N° de compte  
 10101 00011 01 0001111 10 11

M. DUPOND  
 123 RUE SPECIMEN  
 75 015 PARIS

€ 110100  
 A GRENOBLE  
 Le 10/01/10

Signature  
 X

Chèque N° Série BB (22)

000011 0110011111100 00100011001

Enfin, pour les plus ambitieux d'entre vous, la même opération pourra être effectuée, non plus sur une série de chiffres, mais sur une photographie sur laquelle il s'agira de détecter et reconnaître des vêtements (parmi pull, pantalon, chaussures, etc.).



## 1.2 Méthodologie générale, semaine après semaine

Afin de résoudre le problème (dont la solution n'est pas simple et loin d'être unique!), nous allons procéder ainsi:

- TP1.** nous allons utiliser et manipuler la base de données **MNIST** constituée de 70 000 images entièrement étiquetées de chiffres manuscrits: elle nous servira de **base de données d'apprentissage et de validation**
- TP2.** nous implémenterons nos premiers algorithmes de classification (kNN et SVM) sur la base de données **MNIST**.
- TP3.** nous nous pencherons ensuite sur l'utilisation de réseaux de neurones pour la classification de **MNIST** et effectuerons des comparaisons entre les méthodes ainsi développées.
- CC1.** le rapport portant sur les résultats établis des TP1 à TP3 formeront la base d'évaluation du CC1.
- TP4.** dans la 2e partie du semestre, nous allons débiter la traitement de chiffres **manuscrits et scannés par vos soins**, avec une gradation de difficulté (de chiffres bien écrits alignés sur feuille à carreaux à des chiffres désalignés de taille variée et mal écrits); à ce niveau, nous définirons manuellement la position des chiffres.
- TP5.** nous généraliserons ensuite l'approche en automatisant le scan de l'ensemble des chiffres manuscrits afin à la fois de les localiser et de les identifier.
- TP6.** la finalité du projet consistera alors à généraliser l'approche sur des chiffres rédigés de manière arbitraire; l'extension, bien plus délicate, aux images de vêtements, en s'appuyant sur la base de données **Fashion-MNIST** pourra être envisagée.
- CC2.** le rapport final comprenant l'ensemble des résultats, interprétations et méthodes développées pendant le semestre formera la note de CC2.

## 2 Préparatifs

### 2.1 Mise en place de l’environnement de travail

Deux options s’offrent à vous pour travailler sur le projet:

- via Jupyterhub (option la plus simple)
- via VSCode (que vous utiliserez de toute façon en INF203)

#### 2.1.1 Jupyterhub

*(vous pouvez passer cette section si vous souhaitez utiliser VSCode)*

Jupyterhub est très facile d’emploi:


- connectez vous à <https://jupyterhub.univ-grenoble-alpes.fr> avec votre compte Agalan
- la première fois, créez un projet via l’onglet Nouveau, puis Python 3
- aux prochaines sessions, il vous suffira de relancer votre projet qui apparait dans la liste des fichiers
- Jupyterhub ne contient pas tous les environnements disponibles sur turing (option VSCode), et il peut vous être nécessaire d’installer manuellement ces librairies en utilisant la console locale via “!”, par exemple:

```
!python -m pip install -U scikit-image
```

#### 2.1.2 VSCode

*(vous pouvez passer cette section si vous souhaitez utiliser Jupyterhub)*

Il s’agit tout d’abord de se connecter sur le serveur **turing** qui dispose d’un interpréteur Python avec les modules nécessaires et d’un moteur Jupyter-Notebook. Pour cela,

- lancez Visual Studio Code
- cliquez sur le symbole  en bas à gauche et choisir “Connect to Host”
- si c’est votre première connexion, choisissez alors “Add Host” et entrez:

```
ssh -X [votre_nom_agalan]@turing.e.ujf-grenoble.fr
```

sinon, choisissez directement **turing.e.ujf-grenoble.fr**

- entrez votre mot de passe (cela peut ensuite prendre un peu de temps): vous êtes connecté!

Si c’est votre première connexion, il vous faut alors installer les extensions suivantes (icône de gauche avec les 4 carrés) sur **turing**:

- Python
- Jupyter.

Utilisez alors l’explorateur (icône “fichiers” du panneau de gauche) pour créer un dossier INF103 dans lequel vous créerez le fichier vide:

```
projet.ipynb
```

Ouvrez le fichier: vous accéderez à l’environnement Jupyter-Notebook qui vous permet d’ajouter du code Python (+Code) ou du texte (+Markdown).

**Vous êtes désormais prêt à travailler!**

*(n’hésitez pas à demander de l’aide à vos camarades, et en second recours à vos professeurs, si votre environnement de travail ne fonctionne pas)*

## 2.2 Soumission du travail en fin de TP

Le fichier `projet.ipynb` sera utilisé et mis à jour tout au long du projet et sera à la fin de chaque séance posté sur Caséine: **soyez particulièrement soigneux et utilisez de bonnes pratiques de rédaction et d'organisation!**

À la fin du TP, exportez votre document de travail au format HTML:

- sur Jupyterhub: **Fichier** → **Exporter** → **HTML**
- sur VSCode: icône en haut à droite **Export** → **HTML**. Le fichier se trouve alors dans votre répertoire local (`INF103/projet.html`) mais sur `turing`. Pour le récupérer sur votre session Windows locale, utilisez l'outil “FTP turing” disponible sur le bureau. Vous pourrez alors copier `INF103/projet.html` vers (par exemple) **Bureau**.

Déposez alors `projet.html` sur le dépôt du TP (TP1 pour ce premier TP) sur Caséine.

## 3 Découverte et manipulation de la base MNIST

### 3.1 Chargement des bibliothèques Python

En utilisant la méthode `import ... as ...`, chargez les bibliothèques:

- `numpy`: (on pourra la nommer `np`) outils NUMériques pour PYthon.

Aide: <https://numpy.org/doc/stable/index.html>

- `matplotlib.pyplot`: (on pourra la nommer `plt`) outils graphiques mathématiques

Aide: [https://matplotlib.org/stable/api/\\_as\\_gen/matplotlib.pyplot.html](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.html)

- `matplotlib.image`: (on pourra la nommer `img`) outils de traitement d'images

Aide: [https://matplotlib.org/stable/api/image\\_api.html](https://matplotlib.org/stable/api/image_api.html)

```
1 import [...] as [...]
```

Par exemple, avec `import numpy as np`, on peut alors écrire `np.sqrt(x)` pour exécuter l'opérateur `sqrt` (racine carrée) de la librairie `numpy` sur la variable `x`.

Comme très souvent par la suite, nous allons avoir besoin de librairies issues de l'important paquet `sklearn` pour l'apprentissage automatisé. Le format général est le suivant:

```
1 from sklearn.[...] import [...]
```

Aujourd'hui, nous avons juste besoin de charger des bases de données en libre accès. Pour cela, importez la librairie `fetch_openml` du sous-paquet `sklearn.datasets`.

### 3.2 Chargement des données

Commençons par charger les données des bases `MNIST` ou `Fashion-MNIST` (testez les deux!). Nous souhaitons récupérer à la fois les données, sous la forme d'une matrice `X`, ainsi que leurs étiquettes, sous la forme d'un vecteur `y`. Par défaut, le format de `X,y` sont des "dataframes" qu'on convertira en matrice et vecteur compatibles avec `numpy`. Utilisez pour cela la fonction `fetch_openml` comme suit:

```
1 X,y = fetch_openml(name='[...]', version=1, return_X_y=True)
2 X = X.to_numpy()
3 y = y.to_numpy()
```

où on pourra choisir `mnist_784` ou `Fashion-MNIST` (cherchez également d'autres bases de données si vous le souhaitez!).

Les images `MNIST` et `Fashion-MNIST` sont de taille  $28 \times 28$  pixels, stockés ligne-après-ligne sous la forme d'un vecteur de taille 784 ( $28 \times 28 = 784$ ) dans chaque ligne de `X`. En utilisant la fonction `shape` de `numpy`, déterminez le nombre d'images (= de lignes) de la matrice `X`.

Extrayons et visualisons la  $i$ -ème image de `X`: pour ce faire, isolez tout d'abord le vecteur de la ligne  $i$  de `X` que l'on remettra sous la forme d'une matrice  $28 \times 28$  grâce à l'opération `.reshape((28,28))` (attention à la double parenthèse: `reshape` appelle un couple). On pourra ensuite imprimer, grâce à `print()`, la matrice obtenue et tenter de visualiser le chiffre (il peut être nécessaire de transposer la matrice en utilisant l'opérateur `.T`). Que remarque-t-on sur les valeurs prises par les différents pixels? À quelles valeurs correspondent le noir et le blanc?

**Attention:** Lors de l'utilisation des classifieurs, au cours des TPs suivants, les grandes valeurs prises par les pixels poseront souvent un problème de stabilité: il nous faudra penser à *normaliser*

l'ensemble des données de sorte à ce que les valeurs des pixels soient comprises dans l'intervalle  $[0, 1]$ . Effectuez la normalisation de `X` afin de vérifier cette propriété. N'oubliez pas de répéter cette opération au cours des TPs à venir!

Revenons à la visualisation des images: il est en fait plus simple d'utiliser la fonction `imshow` de `matplotlib.pyplot` pour représenter l'image  $i$  en dégradé de couleurs plutôt qu'en valeurs chiffrées. L'option `cmap='gray'` permettra d'avoir une représentation en niveaux de gris plus claire (`cmap` est la carte de couleur: "color map").

Confirmez que le chiffre correspond bien à l'étiquette correspondante du vecteur `y`.

Il peut être vite pénible de trouver une image d'une étiquette donnée (par exemple récupérer plusieurs 3). Pour cela, on peut filtrer les lignes de `X` grâce à:

```
1 X[y=='3']
```

Cela retourne une matrice (en général) ne contenant que les lignes pour lesquelles l'étiquette associée vaut '3' (attention, c'est le caractère '3' et non pas le chiffre 3). Quelle est la taille des différentes sous-matrices pour chaque chiffre? (on pourra utiliser une boucle et l'opérateur `str()` qui transforme un nombre en chaîne de caractères) La base de données est-elle équilibrée?

On peut généraliser l'idée en opérant sur les conditions booléennes. Par exemple

```
1 X[(y=='3')+(y=='4')]
```

filtrera à la fois les 3 et les 4. Que signifie en effet l'opérateur `+` ici? Et l'opérateur `*`? Créez ainsi une base de données `X_bin` ne contenant que les chiffres 0 et 1. Vérifiez que cela fonctionne bien. À quoi cela peut-il servir?

Vérifiez bien sûr que les images filtrées sont bien celles souhaitées. Pour cela, on pourra représenter dans 4 graphes successifs les premières images de la matrice filtrée. On pourra utiliser notamment:

```
1 fig, (ax1, ax2, ax3, ax4) = plt.subplots(1, 4, figsize=(16, 4))
2
3 ax1.imshow([...])
4 ax1.set_title('Image 1')
5 ...
6 ax4.imshow([...])
7 ax4.set_title('Image 4')
```

qui permet de représenter plusieurs images à la suite (vous pouvez bien sûr jouer avec les paramètres de `.subplots` pour comprendre son fonctionnement). Il sera très pratique d'utiliser cette fonctionnalité plus tard dans le projet.

### 3.3 Préparation des bases de données d'entraînement et test

Afin de pouvoir entraîner nos algorithmes de classification, il est important de diviser la base de données en une partie "entraînement" et une partie "test" (ou validation). Coupez les ensembles `X` et `y` afin de créer un sous-ensemble d'entraînement et un sous-ensemble de test:

```
1 X_train, y_train = ...
2 X_test, y_test = ...
```

Justifiez le choix des tailles de chaque ensemble. Quel est l'intérêt d'avoir une grande base d'entraînement? Une grande base de test? Quel est selon vous le meilleur compromis?

### 3.4 Manipulation des images

L'opérateur `roll` de `numpy` permet de faire "tourner" les entrées d'un vecteur (décaler à droite de manière cyclique). Observez ce qu'il se passe lorsque l'on fait tourner l'image  $i$  avec un paramètre `shift` différent. Expliquez ce comportement. On gardera cette astuce en mémoire pour les derniers TPs de l'UE!

### 3.5 Récapitulons

Avec tout ce que vous avez vu jusque là, créez une base de données composées de 3 classes:

- la classe '0' est l'ensemble des chiffres 0
- la classe '1' est l'ensemble des chiffres 1
- une classe 'z' constituée de chiffres "0 et 1" mal centrés

Pour cela, on pourra utiliser

- le raccourci `['z']*n` qui génère un vecteur de taille `n` constitué uniquement de 'z'
- la fonction `concatenate([ y1,y2,... ])`, `concatenate([ X1,X2,...])` de `numpy` afin d'étendre des vecteurs ou matrices.