Université de Strasbourg

Intelligence artificielle

Licence 3 - UFR Mathématique - Informatique

TP : Python – bases, numpy, pandas

Langage de programmation à utiliser : Python3 (documentation : `https://www.python.org/doc/`). Un ensemble de ressources est listé sur moodle.

L'objectif de ce TP est de vous permettre de vous rafraîchir la mémoire concernant l'utilisation du langage Python et de commencer à vous familiariser avec les bibliothèques Numpy et Pandas.

Pourquoi python pour ces TP ?

En cursus informatique, vous avez très certainement été habitué à des langages plus bas niveau comme le C. Python, lui, est un langage très accessible à l'ensemble des scientifiques non informaticiens et est assez largement utilisé dans de nombreuses disciplines scientifiques et de nombreuses bibliothèques, pour la plupart ouvertes, sont accessibles. On peut citer par exemple :

- Matplotlib : visualisation de données `https://matplotlib.org/`
- Biopython : biologie computationnelle et bioinformatique `https://biopython.org/`
- DEAP : calcul évolutionnaire `https://github.com/deap`
- PsychoPy3 : sciences comportementales `https://www.psychopy.org/`
- SciPy : calcul scientifique `https://scipy.org/`
- etc. ! il existe vraiment *beaucoup* de bibliothèques python.

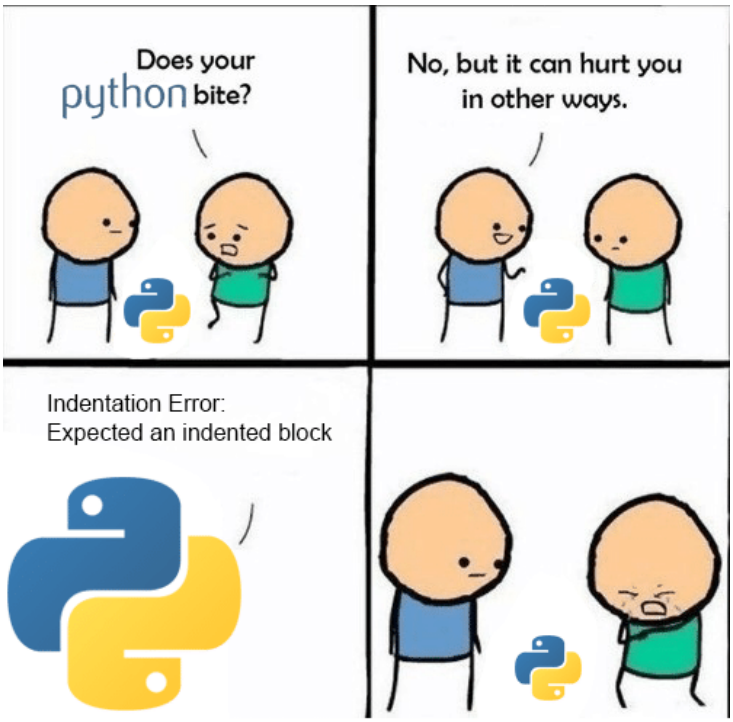
Pour tout ce qui traite de l'apprentissage automatique, c'est aussi en python que l'on trouvera les bibliothèques les plus utilisées telles que :

- scikit-learn : apprentissage automatique en général `https://scikit-learn.org/stable/`
- pandas : analyse de données `https://pandas.pydata.org/`
- tensorflow / Keras : apprentissage automatique, *deep learning* `https://www.tensorflow.org/` et `https://keras.io/`
- PyTorch : apprentissage automatique, *deep learning* `https://pytorch.org/`
- Theano : apprentissage automatique, *deep learning* `https://github.com/Theano/Theano`
- etc.

Dans le cadre de ce cours, nous n'allons pas utiliser de bibliothèques qui contient des modèles clé en main. L'objectif est de mettre en œuvre *from scratch* des modèles d'apprentissage automatique, ce qui permettra de bien comprendre comment ceux-ci fonctionnent et ainsi être capable d'émettre des hypothèses sur les résultats produits, qu'ils soient cohérents ou non, ou encore de mieux comprendre comment aborder des données en amont (pré-traitement, choix d'un modèle, ...)

Ça ne veut pas dire que l'on ne va pas s'aider un peu : les bibliothèques `numpy` et `pandas` seront utilisées pour avoir accès à quelques opérations de haut niveaux (manipulations de tableaux et matrices, imports des données et leur manipulation).

Pour celles et ceux qui s'inquiètent des performances de Python : il n'y a pas de quoi s'en faire. Déjà, parce qu'il faut relativiser la complexité des données et modèles que nous allons utiliser et développer ☺ ; secundo parce que dans toutes ces bibliothèques, Python n'est souvent rien de plus qu'une API permettant un accès à des routines codées en langages plus performants comme C, C++, Fortran, ou encore CUDA (pour le calcul scientifique sur cartes graphiques).


Does your python bite?
No, but it can hurt you in other ways.
Indentation Error:
Expected an indented block

He doesn't.

Environnement de développement

Il n'y a pas de contrainte concernant l'environnement de développement à adopter. La seule chose qu'il faut, c'est que cela tourne sous **Python 3**.

Principalement, vous vous porterez soit vers un IDE "classique" type Spyder [1] soit vers des *notebooks* : en ligne, avec l'outil Colab de Google [2] ou en local, en utilisant Jupyter [3]. Celles et ceux qui ont l'âme plus *roots* peuvent simplement éditer leur code dans un éditeur de code et exécuter leur programmes `.py` en ligne de commande.

Si vous travaillez en local, une solution "propre" est de vous créer un environnement python virtuel pour les TP d'IA, dans lesquels vous pourrez installer les différentes bibliothèques que nous allons utiliser. Pour la création d'environnements virtuels python, c'est par ici : `https://docs.python.org/3/tutorial/venv.html`. Le gestionnaire de paquets python sur les machines de l'UFR est pip.

1 Manipulation de vecteurs / matrices

L'idée ici n'est pas de faire les choses à la main, mais de trouver les bonnes fonctions dans la doc ! Vous trouverez la documentation de l'API Numpy ici : `https://numpy.org/doc/stable/reference/index.html`.

1. Créer un vecteur `v` de 10 éléments entiers aléatoires de 0 à 10.
   (a) Sélectionner tous les éléments de ce vecteur jusqu'aux 4ème (inclus).
   (b) Sélectionner tous les éléments de ce vecteur après le 4ème.
   (c) Sélectionner le dernier élément de ce vecteur.
   (d) Sélectionner tous les éléments de ce vecteur sauf le dernier.
2. Créer une matrice `a` de 4×3 éléments entiers aléatoires de 0 à 10 inclus.
   (a) Sélectionner les 2 premières ligne de la matrice.
   (b) Sélectionner les 2 premières colonnes de la matrice.
   (c) Sélectionner les 2 premières lignes de la 3ème colonne de la matrice.
   (d) Sélectionner le dernier élément de la matrice.
   (e) Afficher la forme de la matrice.
   (f) Afficher le nombre de lignes de la matrice.
   (g) Afficher le nombre de colonnes de la matrice.
   (h) Afficher le nombre total d'éléments dans la matrice.
   (i) Afficher la dimension de la matrice.
3. Manipulation++ de vecteurs et matrices
   (a) Transposer le vecteur précédemment créé.
   (b) Applatir la matrice précédemment créée. Comment est arrangé le vecteur issu de la transformation ?
   (c) Redimensionner `v` de façon à l'afficher comme une matrice de 2 lignes et 5 colonnes.
   (d) Créer une seconde matrice `m` de mêmes dimensions que la première et contenant des valeurs aléatoires distribuées normalement autour de 2 et avec un écart type de 1. Vous aurez auparavent fixé la graine du générateur de nombre pseudo-aléatoire de numpy.

1. `https://www.spyder-ide.org/`
2. `https://colab.research.google.com/` — nécessite d'avoir un compte sur Google
3. Guide de démarrage : `https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/`

(e) Est-ce possible de multiplier les matrices `a` et `m` ? Pourquoi ?

(f) Créer la matrice `m2` qui sera la transposée de `m`.

(g) Multiplier les matrices `a` et `m2`.

(h) Calculer la moyenne et l'écart type de `m2`.

(i) Créer deux matrices carrées `a1` et `a2` de dimensions 3×3 et contenant des entiers aléatoires entre 0 et 10 non inclus.

(j) Sommer `a1` et `a2`.

(k) Soustraire `a1` et `a2`.

2 Manipulation d'un `Dataframe`

Vous trouverez sur moodle un fichier nommé `abalone.csv`. Ces données sont issues d'une étude sur des populations d'ormeaux, dont vous pouvez consulter la description en suivant le lien `https://archive.ics.uci.edu/ml/datasets/Abalone`. Nous n'allons pas faire d'apprentissage sur ces données, mais simplement faire quelques manipulations avec la bibliothéque `Pandas` pour vous familiariser avec celles-ci (lien vers la documentation : `https://pandas.pydata.org/docs/reference/index.html`)

1. Importer les données dans un dataframe.
2. Afficher les 10 premières instances du jeu de données.
3. Quels sont les noms des attributs du jeu de données ?
4. Combien d'instance ce jeu de données comporte-t-il ? Manque-t-il des données ?
5. Quels sont les types des attributs ?
6. Considérons que ce que nous souhaitons prédire le nombre d'anneaux des individus selon leurs différentes caractéristiques.
   (a) Combien y'a-t-il de valeurs d'anneaux différentes ?
   (b) Quels sont les 3 valeurs les plus représentées en nombre d'instances ?
7. Considérons maintenant que nous souhaitons prédire le sexe d'un individus selon ses caractéristiques.
   (a) Quelles sont les valeurs possibles de l'attribut correspondant ?
   (b) Combien d'instances existe-t-il pour chaque valeur possible ?
   (c) La répartition des instances parmis ces valeurs possibles vous semble-t-elle équilibrée ?
   (d) Transformer les valeurs possibles de l'attribut en un code entier.
   (e) Séparer les données en deux ensembles `X` (les attributs) et `y` (les cibles)
   (f) Transformer votre dataframe `X` et votre série `y` en tableaux numpy.