

0.1 Introduction

Je ne connais pas ton niveau précis en informatique, étant donné que tu as déjà travaillé avec python pour tes études et peut-être même durant ton stage il est possible que certaines notions te paraissent évidentes. Je ne donnerai pas beaucoup d'explication moi-même parce que le but de ce projet c'est justement que tu apprennes à t'en sortir avec des ressources extérieures.

En fait le plus important c'est que tu t'habitue à écrire du python qui manipule des données et à apprendre par toi même. A ce sujet, une compétence importante à avoir c'est de bien savoir manipuler son moteur de navigation (google je suppose) pour rapidement trouver les informations dont tu as besoin. Voilà quelques méthodes pour trouver plus facilement des résultats :

1. Ta recherche devrait toujours être faite en anglais (sauf si tu cherches quelque chose de typiquement français) ça te donnera bien plus de résultat.
2. N'écris pas de phrases écris des mots clés, ce n'est pas "Why is tensorflow slow to run" mais "why tensorflow slow". Raccourcis tes phrases au minimum syndical mais fait attention de bien garder les mots importants que tu veux voir apparaître dans les réponses.
3. Si tu veux qu'un mot apparaisse forcément dans tous tes résultats et de manière exacte, mets le entre guillemets. (why 'tensorflow' slow)
4. Au contraire si tu fais une recherche ambiguë sur deux deux domaines, élimine celui qui ne te convient pas avec le '-' (moins) qui permet d'enlever le mot suivant des résultats.
5. Enfin le plus important c'est de noter les mots que l'on veut voir dans la réponse, google est stupide, il ne sait pas répondre à des questions, mais en lui donnant suffisamment de mots clés de ta réponse, tu devrais tomber sur quelque chose d'intéressant.

Et finissons par une révélation, macOS c'est une distribution linux avec une cape donc tout ce qui marche sur linux marche "en général" sur mac. Ça devrait déjà t'être utile pour trouver des réponses à tes questions quand personne n'a répondu pour mac spécifiquement.

Chapter 1

Environnement

1.1 Le terminal

1.1.1 Qu'est ce que c'est

Le terminal tu doit connaître ça, c'est cette sorte de petite fenetre toute noire ou on peut taper des commandes et revenir à la ligne. Concrètement, ton système d'exploitation n'à pas besoin d'une interface pour fonctionner et un terminal c'est en fait une interface texte pour utiliser macos sans avoir à passer par son interface. En quoi est-ce utile ? Il se trouve que toutes les commandes s'écrivent "nom arguments..." donc dans la plupart des cas c'est bien plus court que d'aller se ballader dans une interface, et puis c'est bien connu en info on est un peu feignants donc t'as deviné pour quelle solution on a opté.

Bon voilà quelques exemples de commandes utiles dont tu vas devoir te servir :

```
git clone https://github.com/utilisateur/depot.git
mkdir Projet
mv ../depot .
./install.sh
rm -rf .
```

git, mkdir, mv, rm sont tous des programmes situés dans un endroit de ton ordinateur que tu appelles et qui font des choses pour toi. La première chose que tu dois apprendre c'est donc à te servir d'un terminal de commande, les sections suivantes te diront comment faire. Déjà comment lancer le terminal ? Appuie sur cmd+espace et tape terminal et sélectionne l'icone du petit carré noir.

Naviguer dans les fichiers

La première étape c'est de savoir comment naviguer dans les fichiers avec ta ligne de commande. J'entend par la savoir se déplacer dans tes dossiers, en créer de nouveau, savoir les copier les supprimer... Pour cela je te recommande

les ressources suivantes :

```
<https://www.digitalocean.com/community/tutorials/basic-linux-navigation-and-file-management>
```

Celui là est un tutoriel détaillé pour naviguer dans tes directories.

```
<https://askubuntu.com/questions/232442/how-do-i-navigate-between-directories-in-terminal>
```

celui ci c'est une réponse à une question (ne regarde que la première réponse) elle est pour ubuntu mais comme je t'ai dit c'est pareil sur macOS. Ca t'informera un peu plus sur des particularités du système.

Executer des programmes

La deuxième chose la plus importante sera de savoir executer des commandes (des programmes en fait) et de savoir ce que représente leur arguments...

Pour cela retiens simplement que tu devras utiliser la syntaxe suivantes :

nomduprogramme options1, option2, option3

Maintenant il faut que tu t'habitue à certains programmes qui marchent de cette manière à savoir git et conda. Ils vont former la base de ton environnement. Attend un peu je te les présente bientôt.

1.2 Git(hub)

1.2.1 Pourquoi utiliser un CVS

Je te dis d'utiliser git mais pourquoi ? Bah tu admettras que ce sera plutôt sympa quand tu écris du code de pouvoir revenir en arrière autant que tu le veux, et si en plus tu pouvais accéder à ton code depuis n'importe où tu diras pas non. Tout ça c'est le but d'un **C**entralised **V**ersion **C**ontrol. En bref c'est un programme qui se souvient de chaque étapes dans ton code et qui te permet de retourner en arrière, de publier ton code où de collaborer. Git est un CVS, c'est pas important de savoir ce qu'il a de spécial. Mais alors qu'est ce que github ? github est à git ce que google drive est à tes dossiers. C'est juste un serveur qui se propose de conserver ton "dépot" git pour que tu puisse le télécharger sur un autre ordi et donc travailler de n'importe où avec n'importe qui.

1.2.2 Les bases

Il faut que tu comprenne d'abord la logique de travail avec git. Pour cela tu peux lire ces ressources :

```
https://www.freecodecamp.org/news/learn-the-basics-of-git-in-under-10-minutes-da548267cc91/>
```

```
https://guides.github.com/introduction/git-handbook/
```

Voilà pour git, avant de passer à la suite tu dois être capable de :

1. Créer un repository (dépot)
2. Ajouter du code et des fichier
3. Faire un commit
4. Envoyer tout ça sur un remote sur github

Pour cela il te faudra un compte sur github aussi. Une fois que tu sauras faire tout ça tu seras bon pour utiliser git dans 99% des cas

1.3 Conda et python

1.3.1 Pourquoi utiliser conda

Bon maintenant tu te demandes peut être ce que c'est que conda et pourquoi tu en aurai besoin pour utiliser python. C'est plus simple pour ça de regarder quel problème conda résout : sur python tu peux rapidement télécharger beaucoup de librairies, mais tu vas te rendre compte que il y a certaines librairies que tu n'utiliseras jamais sur le même projet. Alors pourquoi s'embêter à avoir un seul python sur ton ordi avec toutes les librairies alors que tu pourrai avoir un python par catégorie de librairie ? Bah voilà le concept de conda, te permettre de gérer des environnements différents. Et vu que c'est un socle à python même on commence par là ! Bon au passage il se trouve que cpnda sert aussi de package manager à python (comme pip si tu veux) en gros tu peux lui demander de télécharger des librairies et de les installer automatiquement. Il ne reste plus qu'à apprendre à s'en servir !

1.3.2 Utiliser conda

Bon tout d'abord il va falloir installer conda, pour cela suit le tutoriel suivant : <https://docs.conda.io/projects/conda/en/latest/user-guide/install/macos.html>

Il se trouve que ce site c'est celui de la documentation de conda, il est plutôt bien fait et t'expliquera tout, mais comme on est au départ du projet je vais encore te donner les liens qui correspondent à chaque étape. Retiens néanmoins que tu seras très souvent confronté à des sites très similaires voir plus aride. Bon voici maintenant un tutoriel qui t'explique les bases de conda : <https://docs.conda.io/projects/conda/en/latest/user-guide/getting-started.html#managing-envs>

Je veux qu'avant d'avancer tu saches :

1. Créer un environnement
2. Ajouter des packets(librairies)
3. Supprimer des packets
4. Lister tes librairies installées
5. Naviguer entre tes environnements

Ca c'est aussi 99% de ce que tu devras faire avec conda donc maintenant te voilà près à passer au concret.

1.3.3 Préparer l'environnement

Tu vas donc maintenant créer un nouvel environnement appelé "datascience" et tu y installera les librairies suivantes :

- pandas
- numpy
- scipy
- matplotlib
- openpyxl
- tensorflow
- keras
- requests

Voilà avec ça tu devrais pouvoir réaliser tout le projet normalement. Il faut aussi que tu te sentes à l'aise avec les erreurs provoquées par l'absence d'une librairie. Essaie de voir quel message s'affiche si tu ne télécharge pas la librairie et que tu essayes quand même de l'utiliser, histoire que quand ça t'arrivera vraiment tu sois au taquet.

1.4 Pycharm

1.4.1 Intégration avec conda

Pycharm fonctionne à merveille avec conda, quand tu crée un nouveau projet tu as un petit onglet "python interpreter" la choisi "existing interpreter" et tu choisi parmi tes environnements anaconda existants.

Integration avec Git

Là non plus ce n'est pas bien compliqué, tu as un onglet VCS et un sous onglet git qui te permettra d'accéder à toutes les fonctionnalités que tu as utilisées.

Chapter 2

Projet

2.1 Description du projet

Le projet consistera à prédire la qualité du vin en fonction de caractéristiques chimique de ce dernier. Pour cela il faudra extraire des données les caractéristiques utiles à la prédiction. Le dataset devra être récupéré sur internet, nettoyé et analysé.

2.2 Récupération des données

2.2.1 Enoncé du premier projet

La première étape va consister en la récupération du set de données sur internet. Alors tout d'abord une rapide définition de set de données pour qu'on soit d'accord :

Un set de données est un ensemble de données faisant correspondre les caractéristiques d'un élément à une forme de note. Ton dataset est une sorte de tableaux excel communément constitué de n colonnes dites de "features" (nos caractéristiques) et de m colonnes de "label" qui désignent les notes correspondantes (ces termes sont importants). Chaque colonne représente donc un élément (un vin ici) auquel tu associe des propriétés et une (ou plusieurs) note(s).

Par ma définition de dataset tu comprend que la plupart des choses que tu as appris en algèbre linéaire te seront utiles pour exploiter tes données. Mais ça on en discute plus tard. En attendant voilà l'url de ton dataset :

`https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data` `https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.names`

Tu remarqueras qu'il y a deux URL, le premier est tes données, le deuxième est un fichier intéressant qui te donnera des informations sur les caractéristiques que l'on va étudier. Donc sans plus attendre voici la description de ta première étape :

- Créer un nouveau projet sur pycharm avec toutes les librairies requises précédemment.

- Etre capable de télécharger le dataset dans le sous dossier "data" de ton projet (crée le). Je te demanderai évidemment de faire ça avec python et pas à la main.
- Faire en sorte que toutes les fonctionnalités du programme soit séparées en fonctions réutilisable et pas spécialisée (pense que tu voudras réutiliser ces fonctions dans l'année mais avec d'autres valeurs)
- Le tout devra être documenté et annoté comme il se doit

Pour faire tout ça il va te falloir un minimum de compréhension du concept de requêtes web et de la librairie requests. Pour ce faire la sous section suivante t'y introduira. Il faudra évidemment aussi maîtriser la gestion des données sur ton disque dur (ouvrir/fermer un fichier, créer un répertoire...) pour cela utilise le module Pathlib (qui est contenu dans la bibliothèque standard du python donc rien à installer.

2.2.2 Requetes web

Je vais te faire une sorte de micro-cours sur les requêtes web, pour plus de détail je te mettrai à disposition des liens à la fin de mon blabla. Et sache que c'est une semaine de ton cours les requêtes et la récupération de données sur internet donc tu t'avance là.

Internet c'est juste énormément d'ordinateurs comme les autres (un peu plus gros mais rien de plus) avec lesquels tu peux parler de manière très basique. En gros il existe que très peu de verbes (5) dans ce langage mais il existe une infinité d'objets. Les objets il se trouve que ce sont des pages web, et les verbes décrivent ce que tu veux faire avec (la voir/télécharger, envoyer des données comme pour un mot de pass...) En pratique on utilise que ces deux "verbes". Une requête c'est simplement un verbe et un objet. Si tu veux un exemple très concret, quand tu tape sur google la recherche "chat mignon", ce que tu fais vraiment c'est :

GET (verbe) <https://www.google.com/search/chat+mignon> (objet)

En fait tu demande littéralement de télécharger une page qui a été créée par google. Maintenant pour le concret, GET c'est le "verbe" qui permet de télécharger la page. Donc tu comprend maintenant qu'il te faudra un moyen d'envoyer une requête GET à l'adresse de ton dataset. Pour se faire il existe la jolie librairie "requests" qui se propose de te permettre de faire ça facilement. Mais avant de plonger dans la librairie, voilà quelques ressources plus formelles sur les requêtes :

<https://learn.onemonth.com/understanding-http-basics/> <http://www.steves-internet-guide.com/http-basics/>

Ce protocole de communications c'est ce qu'on appelle le HTTP au fait et c'est ce qui est marqué devant chacune de tes url dans ton navigateur parceque ça indique le protocole à utiliser.

2.2.3 Requests

C'est donc la librairie qui va te permettre d'émuler ces requêtes et de télécharger des choses, vu que c'est la librairie que tu utilisera en cours je vais me permettre de simplement te donner le tutoriel présent sur ton cours et la documentation de la librairie pour t'en sortir. Mais ça devrait le faire et c'est surtout très enrichissant. Si tu bloques vraiment inutile de gacher du temps et envoie moi un message.

```
https://github.com/epfl-ada-2018/ADA2018-Tutorials/blob/master/06%20-%20Data%20From%20The%20Web/ADA%20Tutorial%20-%20Data%20from%20the%20Web.ipynb
```

```
https://2.python-requests.org/en/master/user/quickstart/
```

Le tutoriel de l'EPFL couvre quelque chose de légèrement différent à savoir récupérer des données présentes sous la forme de texte mais déjà se servir de requests pour télécharger quelque chose c'est la première étape. Voilà un tutoriel spécifiquement pour télécharger un fichier avec requests :

```
https://www.codementor.io/aviaryan/downloading-files-from-urls-in-python-77q3bs0un
```

2.2.4 Dossiers et fichiers

Je te fais ici un mini topo sur le système de fichier et la manière de manipuler des dossiers/fichiers.

Si tu maîtrise bien ce que tu as lu au départ sur la navigation dans les dossiers, il ne te manque plus qu'à connaître comment réaliser les différentes actions que tu connais. Ainsi, si tu souhaite créer un nouveau dossier tu vas devoir utiliser la fonction *os.makedirs*, pour créer un fichier, ouvre le avec la fonction *open*, sinon de manière générale les objets de type Path (*pathlib.Path*) offre de nombreuses méthodes très pratiques pour la manipulation des chemins de fichiers (extraire le nom du fichier depuis un chemin, passer de chemin relatif à chemin absolu, vérifier si le chemin existe, si c'est un fichier, un dossier...).

2.3 Préanalyse des données

2.3.1 Objectifs

C'est maintenant que commencent les choses plus amusantes. Il va s'agir de la phase la plus importante du projet, en effet, les performances de tes modèles de machine learning dépendent directement de cette phase. Voici donc les objectifs de cette section :

- Maîtriser les différents mécanismes de nettoyage des données.
- Maîtriser les techniques de préparations des données en vue d'une analyse plus poussée.

- Maîtriser la génération de graphiques utiles à la compréhension des liens entre les données.

2.3.2 Charger les données

Après avoir téléchargé et enregistré tes fichiers, tu devras recharger les données en mémoire, évidemment, si tu lance ton programme pour la deuxième fois il semble logique de mettre en place un mécanisme sautant la première partie.

Maintenant il s'agit de savoir comment charger ces données. Il va te falloir ouvrir le fichier, le traverser, et construire un *dictionnaire de listes* de cette forme : Tu donnera une entrée dans le dictionnaire pour chaque colonne de tes données (pense à une feuille excel). Puis dans chaque entrée tu créera une liste (tes lignes) qui accumulera les valeurs correspondantes ligne après lignes. Je reste volontairement flou parce que je veux que tu réfléchisses à quoi ressemblent les données et que tu trouves toi même un moyen de construire cette structure. Une fois que ce sera fait tu vas pouvoir construire une *Dataframe* (*pandas.DataFrame*) à partir de ces données. Une dataframe permet de manipuler de manière bien plus puissante les données que tu viens d'extraire. Pour construire une Dataframe à partir d'un dictionnaire il faut utiliser la méthode *fromdict* de pandas.

2.3.3 Vérification des fichiers

Une chose importante à retenir c'est que les fonctions sont conçues pour un type de variables très précis, si une valeur n'a pas le type attendu, tout explose en général. En python, c'est compliqué d'arriver à cette situation sauf quand tu télécharge des données comme ça, en effet il se pourrait qu'une case soit vide, pas dans le sens 0, dans le sens absolument vide, ou juste invalide. Heureusement, pandas te permet de vérifier tout ça et plus encore. Pour travailler avec ces valeurs embêtantes je te renvoie à cet article :

https://pandas.pydata.org/pandas-docs/stable/user_guide/missing_data.html

Une fois que tu aura réglé cela tu vas pouvoir commencer la vraie préanalyse.

2.3.4 Un premier regard

Il s'agit dans cette étape de te familiariser avec les données et de comprendre comment elle pourraient être liées et à quoi as tu vraiment à faire. Pour cela une première toute petite étape serait de vérifier les dimensions de tes données (*pandas.DataFrame.shape*) cette méthode de ta dataframe te permettra donc de connaître le nombre de colonnes et de lignes qu'elle possède. Ensuite tu peux porter un premier regard avec la méthode *head* de Dataframe qui te montreras les premières lignes. Enfin, *describe* t'offrira des valeurs utiles sur la répartition des valeurs de chaque colonne. Je te conseille de regrouper toutes ces informations dans une fonction pour la faire tourner à chaque fois.