M2DS data-camp : classification supervisée de type cellulaire

Prof. Nicolas Jouvin - nicolas.jouvin@inrae.fr

M2 data science Évry

Challenge du 21 janvier au 1er avril 2025

Program for today

Organisation

Présentation rapide des données et du challenge

Github classroom

La plateforme RAMP

Quelques outils utiles pour le développement Python

À vous !

Organisation

Agenda

3 sessions + soutenance

- **Horaires:** Mardi 8h30-11h45
- **Lieu:** Salle 113, IBGBI (this room)

- Session 1 : aujourd'hui
- Session 2 : mardi 4/02
- Session 3 : mardi 18/02
- Soutenance : mardi 1er avril

Bien envoyer son rapport avant la soutenance

Règles du challenge

- Collaboration : équipes de 2 minimum et 3 maximum
- Pas d'utilisation de données auxiliaires
- Créer un compte Github (si pas d'existant)
- Créer un compte RAMP.studio + demandé l'accès à l'évènement *Challenge on single-cell type classification (jan 2025)*
- Le suivi se fera via le fichier `README.md` de votre repository d'équipe.

Notation globale

1. **5 points** de gestion du projet et de travail en équipe via le suivi README du github de l'équipe : bilans hebdomadaire, répartition du travail équitable, etc.
2. **8 points** pour le rapport écrit avec prise en compte du classement RAMP
3. **7 points** pour la soutenance orale (points individuels)

Attention vous êtes évalués sur la qualité de l'ensemble votre démarche.

Sont valorisés

- une revue de littérature rapide sur les données singe-cell RNAseq
- l'analyse de données : statistiques descriptives, nettoyage, pre-processing, etc.
- l'interprétation des sorties de votre solution : points forts, points faibles, idées d'amélioration possibles, etc.
- La qualité du code

Ne sont pas valorisés : les solutions qui performent bien sans aucune tentative d'analyse.

Pour aujourd'hui

A la fin de cette journée, tout le monde devra avoir

- installé Git
- un environnement Python nommé `data-camp` avec python>=3.10 installé
  `conda create -n data-camp python=3.11`
- rejoins une équipe sur l'assignement Github Classroom
- exécuté `git clone <le-repo-de-son-équipe>`
- exécuté `pip install -r requirements.txt`
- exécuté `ramp-test` sans erreur

Présentation rapide des données et du challenge

Données single-cell RNAseq

Pour 1 cellule unique $\rightarrow$ comptage de niveau d'expression des gènes

(Hyp) : $\neq$ gènes exprimés $\implies$ $\neq$ types cellulaire

But : prédire le type cellulaire à partir des niveaux d'expressions de gènes

Formalisation statistique : classification supervisée

- n cellules et p gènes (variables).
- Observations :$\mathbf{X} \in \mathbb{N}^{n \times p}$
- Label : types cellulaire $\mathbf{y} \in \{1, \dots, K\}^n$

On souhaite apprendre un classifieur : $f(\mathbf{X}) = \mathbf{y}$

Le jeu de données scMARK

Jeu de données *benchmark* créé dans le but d'avoir un équivalent de MNIST pour les données single-cell

- Meta dataset : plusieurs dataset compilé en un seul
- $100,000$ cellules et $p \approx 14,000$ gènes
- Les types cellulaires (labels) sont harmonisé (28 au total)
- https://www.biorxiv.org/content/10.1101/2021.12.08.471773v1.full.pdf

Extraction pour le challenge

- on travaillera avec $K = 4$ types cellulaires

  1. Cancer cells 2. NKcells 3. T cells CD4+ 4. T cells CD8+

- dataset public : $n = 1500$ séparées en $\frac{2}{3}$ train / $\frac{1}{3}$ test.

Github classroom

Inscriptions

Deux inscriptions à faire (mêmes équipes) :

1. Suivi-challenge https://classroom.github.com/a/zEWT5-4M
2. Challenge : https://classroom.github.com/a/kmKHlPqy

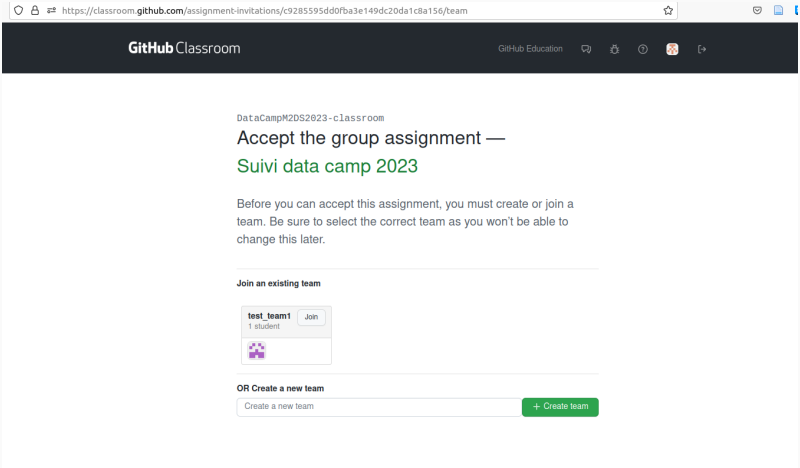
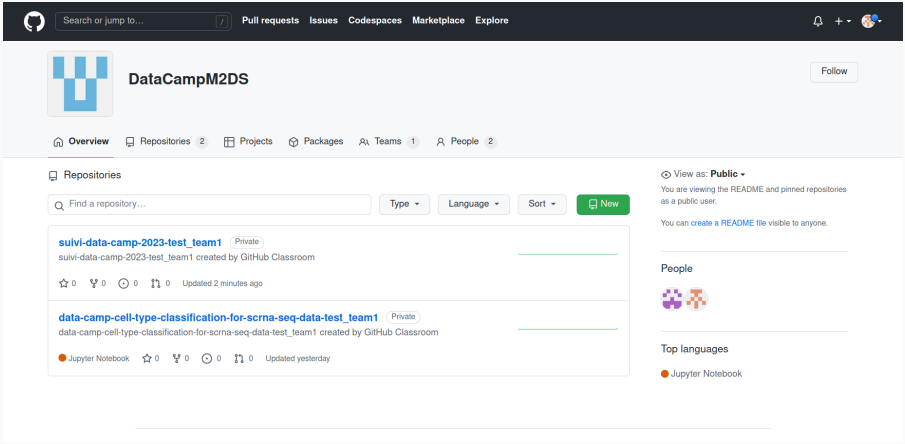
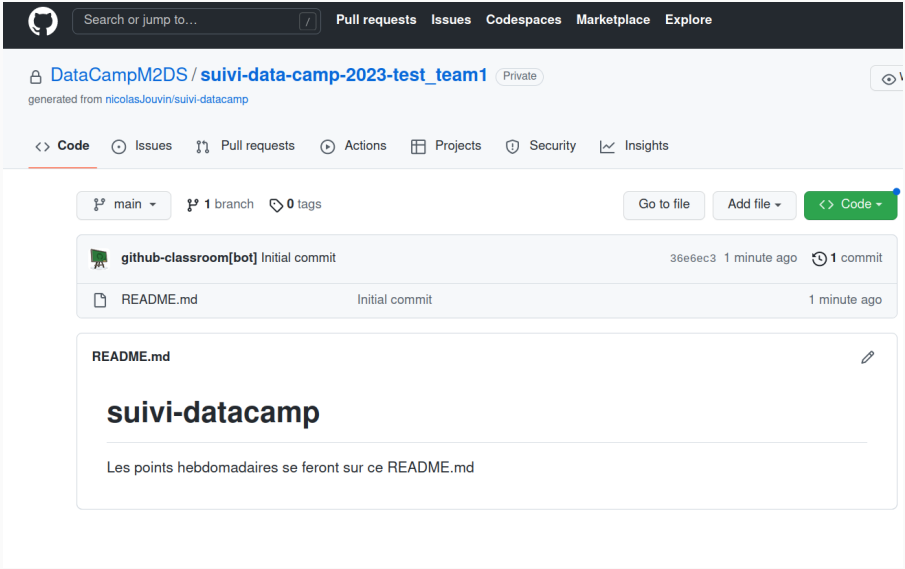


Tableau de bord d'équipe



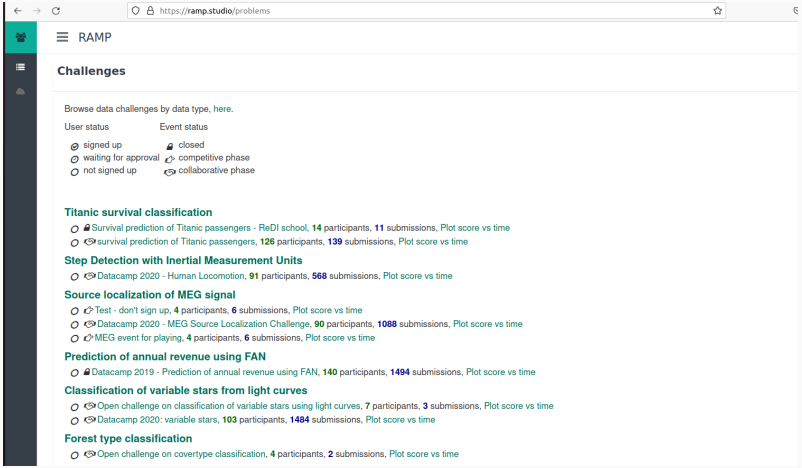
Suivi via README



La plateforme RAMP

Inscription à l'évènement

Inscriptions : https://ramp.studio/sign_up



Fonctionnement

En local la commande `$ramp-test --submission ma_soumission`

- `ma_soumission` contient au moins un fichier `classifier.py` (nom obligatoire)
- `classifier.py` défini obligatoirement une classe Python `Classifier` avec méthodes
  - `.fit(X, y)` : ne retourne rien
  - `.predict_proba(X)` : retourne une matrice avec les probas d'appartenance à chaque classe.
  - vous pouvez définir des fonctions/classe supplémentaires qui seront utilisées par `fit()` et `predict_proba()`

Sur le serveur

- **Soumission :** on soumet `ma_soumission` via la sandbox
- **Exécution :** le code tourne sur un serveur distant
- **Classement :** score de classement calculé sur une jeu de donnée test privé (non-accessible). Seul le score sur le test public est accessible.

La sandbox

Demo en live

Format d'une soumission :

- toute soumission commence **forcément** par le nom de l'équipe.
- vous pouvez ajouter une courte description de votre soumission, par exemple : `team_XXX_PCA50_SVC`.

Quelques outils utiles pour le développement Python

Sur le code que vous produisez

Au fur et à mesure de vos expériences, vous aller produire beaucoup de code Python : analyse de données, petit scripts d'essais, etc.

On peut distinguer 2 niveaux

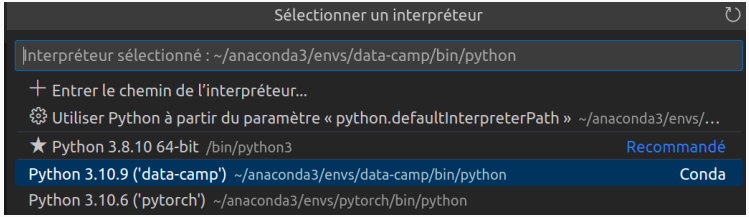
- Local (votre machine) : vous êtes totalement libre sur l'organisation.
- Repository d'équipe : il faut qu'il soit organisé de manière claire avec du code propre.

Attention à ne pas stocker de trop gros fichiers sur github (« 1Mo) : utilisez le `.gitignore` !

J'ai accès au code de vos soumissions sur RAMP

Visual Studio Code (VSCode)

- Python Extension : syntax color, autocomplete, line-by-line execution, debugger, etc.
- Gestion environnements : `ctrl + maj + P >python: select interpreter`



- Auto-formatage du code à la sauvegarde avec Black: `$pip install black`
- Terminal intégré (ou `conda prompt` pour Windows)
- Intègre Git

Jupyter-lab pour les IPython Notebooks

```
$ conda activate data-camp
$ jupyter-lab
```

Conseils :

- (Pros) très utiles pour communiquer vos résultats de manière reproductible à l'équipe.
- (Cons) Attention ce n'est pas un bon environnement de développement ! Séparer les classes/fonctions utiles (`.py`) et l'expérience (`.ipynb`) !

Flake8 et black pour *linter* son code

Linter = ”analyse statique du code”

- variable non déclarée
- variable inutilisée
- non respect des bonnes pratiques du code → flake8 + black

Exemple : `$ flake8 ugly_script.py`

```
ugly_script.py:1:80: E501 line too long (121 > 79 characters)
ugly_script.py:5:1: E302 expected 2 blank lines, found 1
ugly_script.py:6:1: W191 indentation contains tabs
ugly_script.py:7:1: W191 indentation contains tabs
```

Pour gagner du temps : VSCode + Black pour auto-formatage à la sauvegarde.

Deux prochaines séances

- Retours sur votre progression et conseils/feedback

- Conseils généraux pour développer

À vous !