Rapport Data Camp
Single-cell RNA-seq classification

Luc YAO, Léos Coutrot, Alix Madrange

26 mars 2025

Table des matières

1 Introduction

Les cellules partagent le même patrimoine génétique, mais elles n'expriment qu'une partie de leurs gènes, ce qui leur permet de se spécialiser et d'assurer des fonctions variées. Mieux comprendre et classifier ces types cellulaires est donc essentiel en biologie et en médecine.

Avec les progrès du séquençage d'ARN à cellule unique (scRNA-seq), il est maintenant possible d'observer l'expression des gènes à une échelle très fine. Cette avancée ouvre la porte à l'utilisation de l'apprentissage automatique pour analyser et classer les cellules de manière plus précise.

L'objectif est de comparer plusieurs modèles et d'améliorer les méthodes d'identification des types cellulaires, avec des applications potentielles en recherche biomédicale et en diagnostic clinique.

2 Analyse exploratoire des données

2.1 Résumé du jeu de données

Le jeu de données de ce défi de classification repose sur des données de séquençage d'ARN à l'échelle unicellulaire (scRNA-seq) et est utilisé pour prédire des types cellulaires à partir des niveaux d'expression des gènes. Il est extrait du jeu de données scMARK, qui regroupe les expressions de 100 000 cellules provenant de 10 études différentes. Ce jeu de données sera notre référence pour évaluer différentes approches de classification supervisée.

Ici nous avons un sous-ensemble avec 4 types cellulaires à classer :

— Cellules cancéreuses
— Cellules NK
— Cellules T CD4+
— Cellules T CD8+

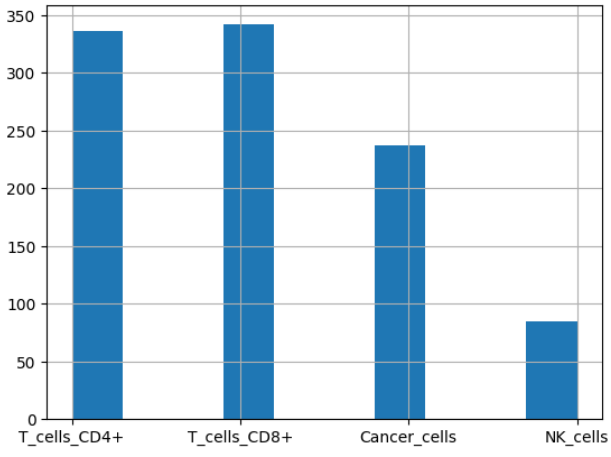


FIGURE 1 – Histogramme du jeu de données

On voit tout de suite sur le graphique 1 que nos classes sont très déséquilibrées. Ainsi, on comprend mieux le choix de la métrique de balanced_accuracy pour évaluer la performance des modèles. En effet, une métrique comme l'accuracy classique pourrait nous induire en erreur quant aux réelles performances de notre modèle (en ne prédisant presque jamais la classe minoritaire, par exemple).

2.2 Normalisation et transformation du jeu de données

Dans le cadre de notre analyse de données de type single-cell RNA-seq, il est essentiel de procéder à une transformation et une normalisation des données afin de rendre les mesures comparables entre les cellules et d'atténuer les biais techniques. Nous avons appliqué deux étapes principales : une transformation logarithmique et deux types de normalisation.

Tout d'abord, nous avons appliqué une transformation *log1p* sur les données brutes. Cette transformation est définie mathématiquement comme suit :

$$\log1\mathrm{p}(x) = \log(x+1)$$

où x représente le nombre de lectures (ou l'expression génique) pour chaque gène dans une cellule. Cette transformation permet de stabiliser la variance des données et de réduire l'impact des valeurs extrêmes ou des faibles niveaux d'expression, facilitant ainsi l'analyse des données à faible expression.

Ensuite, deux types de normalisation ont été réalisés. La première normalisation a été effectuée en utilisant le `StandardScaler` de la bibliothèque `scikit-learn`. Cette méthode consiste à standardiser chaque gène (variable) en soustrayant la moyenne et en divisant par l'écart-type de chaque gène à travers toutes les cellules. Formellement, pour chaque gène g dans la cellule c, la normalisation est donnée par :

$$\hat{x}_{g,c} = \frac{x_{g,c} - \mu_g}{\sigma_g}$$

où $x_{g,c}$ est la valeur brute d'expression du gène g dans la cellule c, μ_g est la moyenne des valeurs d'expression du gène g à travers toutes les cellules et σ_g est l'écart-type des valeurs d'expression du gène g à travers toutes les cellules. Cette normalisation permet de centrer et réduire les données, rendant les différentes échelles d'expression des gènes comparables.

La deuxième normalisation a été effectuée avec la fonction `scanpy.pp.normalize_total` de `Scanpy`. Cette méthode ajuste les valeurs d'expression des gènes de manière à ce que la somme des valeurs d'expression pour chaque cellule soit égale à une constante donnée, souvent 10 000 (en nombre de molécules ou de comptes de séquences normalisés). Mathématiquement, pour chaque cellule c et chaque gène g, la normalisation s'écrit :

$$\hat{x}_{g,c} = \frac{x_{g,c}}{\sum_{g'} x_{g',c}} \times \text{total\_count}$$

où $\sum_{g'} x_{g',c}$ est la somme des expressions des gènes dans la cellule c et total_count est une constante (généralement 10 000). Cette normalisation vise à ajuster l'ampleur des profils d'expression des cellules tout en conservant les relations relatives entre les gènes.

Ces transformations et normalisations sont cruciales pour réduire les biais techniques et améliorer la comparabilité des profils d'expression génétique entre les cellules, ce qui est essentiel pour des analyses de classification efficaces et robustes dans le cadre de la RNA-seq single-cell. Il existe d'autres méthodes de normalisation des données telles que RLE ou TMM, cependant nous avons décidé ici de nous concentrer sur celles présentées, cependant, il pourrait être intéressant d'approfondir ces autres pistes dans un travail futur.

3 Méthodologie : Classification supervisée

L'objectif de ce projet est de prédire le type cellulaire à partir des niveaux d'expression des gènes en utilisant des modèles de classification supervisée. Pour ce faire, nous avons testé plusieurs modèles de classification, notamment : Random Forest, Gradient Boosting, Support Vector Machine (SVM), Adaboost et Régression Logistique.

Dans le cadre de ce projet, nous n'avons accès qu'à une partie des données, pré-séparée en jeux d'entraînement et de test. C'est sur ce jeu de données que nous avons fait tous nos tests (en local). Il est donc important de garder en tête que l'élément final pour évaluer nos modèles sera les résultats sur RAMP (bien que nos résultats en local devraient être similaires à ceux obtenus sur RAMP).

Pour bien préparer les données pour la phase d'entraînement, nous avons implémenté les étapes suivantes :

- Normalisation : transformation *log1p* pour atténuer l'effet des valeurs extrêmes et réduction des données avec StandartScaler
- Séparation du jeu de donnée
- Entraînement des modèles avec validation croisée sur *train* (5-fold CV)
- Évaluation des performances sur les jeux de données *valid* et *test* en utilisant la balanced accuracy

4 Expérimentation et performances des modèles

Les données ont été préalablement divisées en un ensemble d'entraînement (*train*) et deux ensembles de test (*valid*, *test*) avant la phase d'apprentissage, afin de limiter les risques de biais et d'overfitting. Cette séparation permet d'obtenir, pour chaque modèle, une balanced accuracy objective quant aux réelles performances de ce dernier.

4.1 Test avec Filtrage par variance

Nous avons essayé d'appliquer un filtrage par variance sur les données brutes. Cela permet d'éliminer les gènes dont la variance d'expression est trop faible entre les échantillons et ainsi réduire la dimensionnalité du jeu de données. Les gènes à faible variance sont souvent considérés comme peu informatifs pour la classification, car ils ne présentent pas de différences marquées entre les différentes classes.

Nous observons dans le tableau 1 que les modèles basés sur la régression logistique et le Gradient Boosting obtiennent des scores élevés en termes de balanced accuracy. Plus particulièrement, le modèle de régression logistique atteint une performance de 0.803647 sur le jeu de test. Le modèle Gradient Boosting suit de près avec une performance de 0.807525. Le temps d'entraînement du Gradient Boosting étant plus long, la régression logistique est le meilleur modèle parmi ceux évalués.

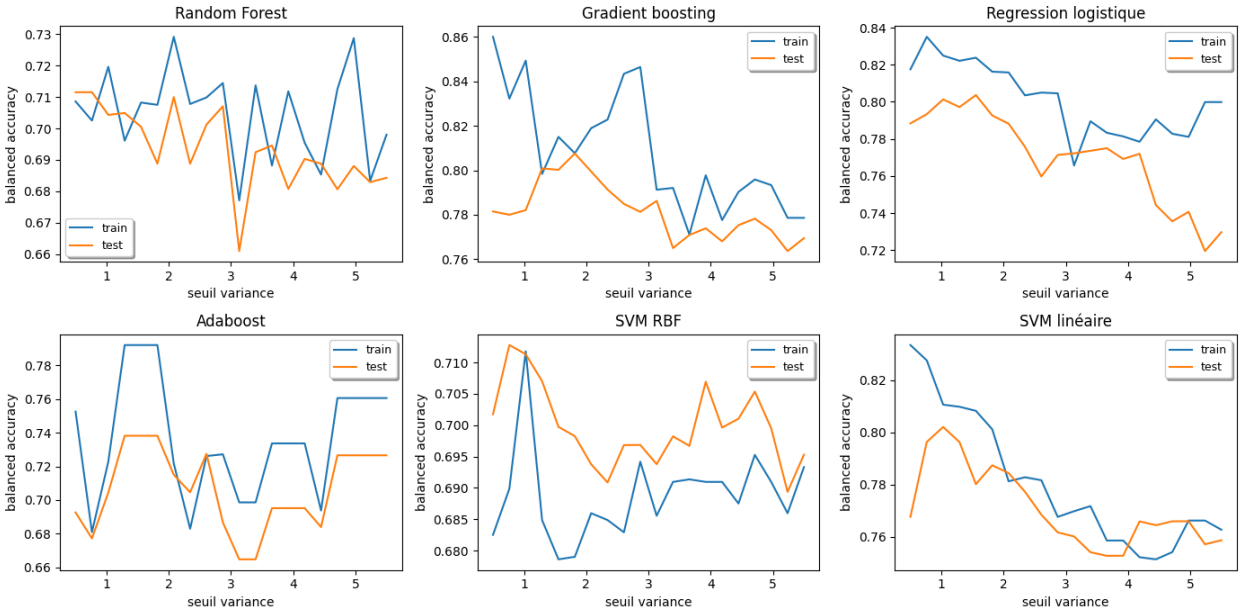


FIGURE 2 – Balanced accuracy en fonction de du seuil pour chaque modèle

Modèle	Seuil optimal	Train BA	Test BA
Random Forest	0.763158	0.702518	0.711541
Gradient Boosting	1.815789	0.807638	0.807525
Régression logistique	1.552632	0.823783	0.803647
Adaboost	1.289474	0.792060	0.738189
SVM RBF	0.763158	0.689861	0.712768
SVM linéaire	1.026316	0.810569	0.802031

TABLE 1 – Performance des modèles avec filtrage par variance

Remarque : il existe une fonction pré-implémentée dans Scanpy qui effectue un filtrage par variance, cependant cette dernière offrait des performances très mauvaises comparées à celle données par notre version. Nous avons donc décidé de nous concentrer sur notre propre version.

4.2 Test avec Régression LASSO

Nous avons utilisé la régression LASSO pour la sélection des variables, ce qui nous permet d'optimiser la robustesse et la généralisation des modèles en éliminant les caractéristiques les moins pertinentes. L'hyperparamètre α, qui contrôle la pénalisation de LASSO, joue un rôle dans cette sélection : plus α augmente, plus on restreint le nombre de gènes (figure 3). Comme le montre la figure 4, ce paramètre influence directement la performance des modèles.

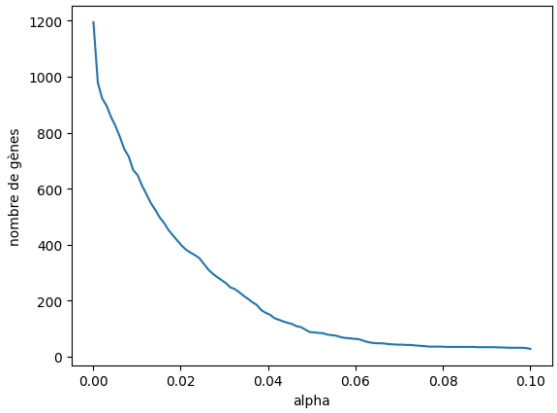


FIGURE 3 – Évolution du nombre de gènes sélectionnés en fonction de α

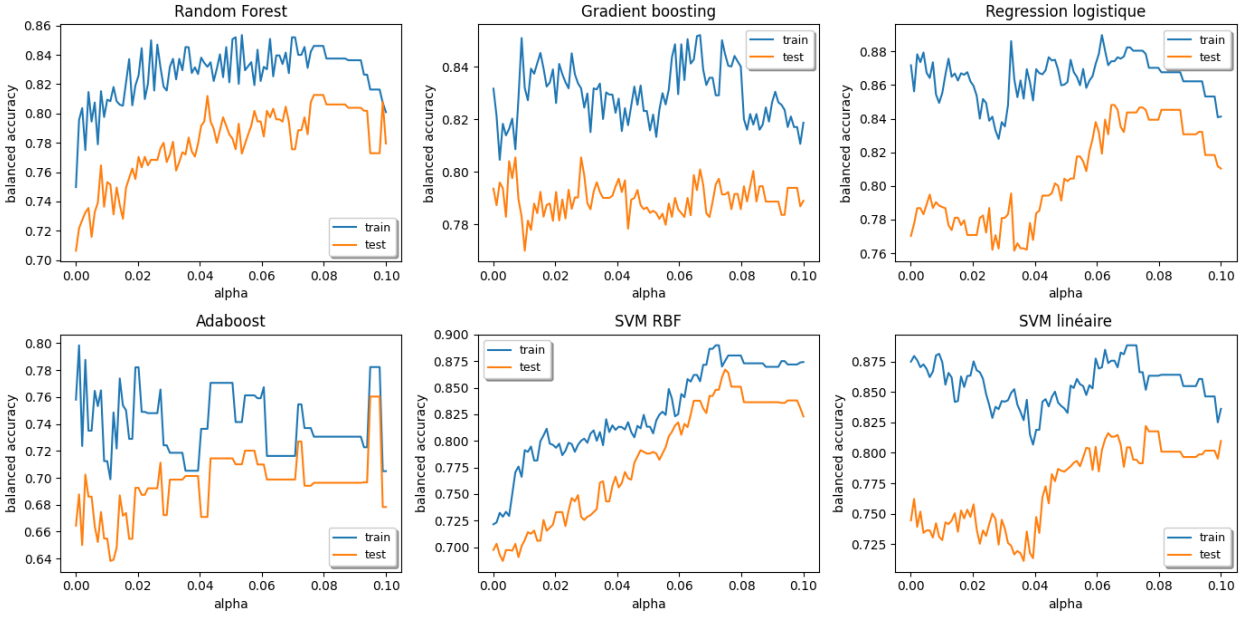


FIGURE 4 – Évolution de la balanced accuracy en fonction de α pour chaque modèle.

Le tableau 2 présente les performances, la balanced accuracy (BA), des différents modèles après sélection du α optimal.

Modèle	**α optimal**	**Train BA**	**Test BA**
Random Forest	0.076791	0.846119	0.812646
Gradient Boosting	0.007164	0.808643	0.805519
Regression logistique	0.064682	0.874163	0.848152
Adaboost	0.094955	0.782285	0.760364
SVM RBF	0.074773	0.875308	0.867072
SVM linéaire	0.075782	0.851672	0.822069

TABLE 2 – Performance des modèles après sélection du meilleur alpha via LASSO

Nous observons que les modèles basés sur les SVM et la régression logistique obtiennent des scores élevés en termes de *balanced accuracy*. Plus particulièrement, le modèle SVM avec noyau RBF atteint une *balanced accuracy* de 0.867072 sur le jeu de test, ce qui en fait le meilleur modèle parmi ceux évalués.

5 Résultats sur RAMP

Classement	Modèle	Méthode	Hyperparamètres	BA
1	SVM RBF	LASSO	$\alpha = 0.005$	0.87
2	SVM RBF + LDA + Régression logistique	LASSO	$\alpha = 0.005$	0.86
3	SVM linéaire	LASSO	$\alpha = 0.005$	0.85
3	Régression logistique	LASSO	$\alpha = 0.005$	0.85
4	Gradient Boosting	LASSO	$\alpha = 0.01$	0.84
5	Régression logistique	Filtrage variance	seuil = 1.5	0.83
5	Gradient Boosting	Filtrage variance	seuil = 1.5	0.83

TABLE 3 – Classement des modèles avec leur méthode et leur performance

Le meilleur score obtenu sur RAMP est de 0.87 de balanced accuracy avec la méthode SVM avec un noyau RBF pour un $\alpha = 0.005$ (tableau 3). Comme les données sur RAMP ne sont pas accessibles, nous avons fait une recherche manuelle des hyperparamètres en les testant un à un sur le dépôt RAMP et en prenant pour point de départ les hyperparamètres optimaux obtenus localement.

6 Méthodes n'ayant pas abouties

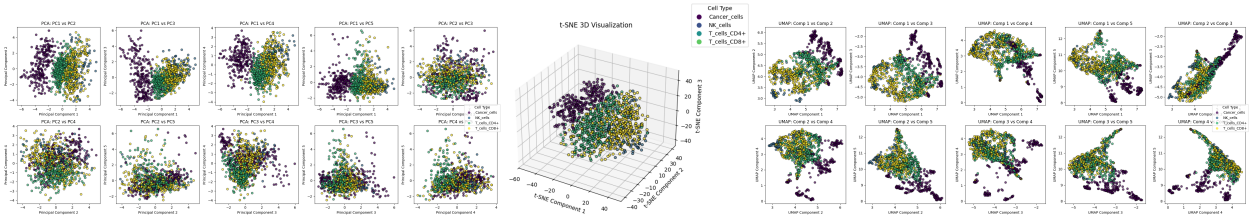
Cette partie va prendre le temps de détailler plusieurs pistes que nous avions envisagées mais qui n'ont malheureusement pas été à l'origine de nos meilleurs résultats. Cependant, l'approche étant tout de même intéressante, nous avons souhaité la détailler.

6.1 Réduction de dimensions

Bien que nous soyons dans un contexte d'apprentissage supervisé, et que la réduction de dimensions soit de manière générale plutôt rattachée au non supervisé, la piste de la réduction de dimensions a été très vite envisagée. Dans de nombreux articles scientifiques décrivant des frameworks pour la classification de données Single-cell RNA seq, la réduction de dimensions était un aspect central (cf cette article).

Les méthodes phares de la réduction de dimension sont l'Analyse en Composantes Principales (ACP et KACP), t-SNE et UMAP. Malheureusement, nous n'avons pas réussi à exploiter les résultats de la réduction de dimensions de sorte à ce qu'elle nous permette d'atteindre d'excellents scores sur le projet, mais puisque nous avons pris le temps d'explorer cette piste et que nous la trouvons intéressante, nous avons décidé de, malgré tout, la détailler ici.

Grâce à la log transformation des données et à une normalisation via scany, nous avons pu obtenir 5 ces trois projections différentes.



(a) ACP (b) t-SNE (c) UMAP

FIGURE 5 – Exemple de trois images côte à côte.

Malgré des résultats plutôt prometteurs (surtout pour t-SNE 3D qui semblait avoir une projection très efficace), l'utilisation de ces méthodes de réduction de dimension couplées avec des algorithmes de machine learning (SVM principalement) n'ont pas donnés d'excellents résultats. Cependant, il est bon de souligner que ces méthodes ont tout de même pu atteindre jusqu'à 0.80 de précision dans le meilleur des cas.

7 Conclusion

Au final, nous avons réussi à atteindre la quatrième place du classement RAMP à ce jour, avec une balanced accuracy affichée très proche des équipes devant nous. Un aspect qui n'a pas été beaucoup développé est la complexité temporelle de nos modèles. En effet, notre meilleur modèle en termes de score sur RAMP est un modèle SVM qui a mis 155 secondes à s'entraîner, alors que notre second meilleur modèle (un mélange de SVM, régression logistique et LDA) a une performance comparable et s'entraîne en seulement 26 secondes. En conclusion, bien que les méthodes d'analyse traditionnelles aient permis de faire des avancées significatives, l'exploration de l'analyse de données topologique serait une piste intéressante pour ce projet.