Data Camp

Soutenance de projet

Yao Luc & Coutrot Léos & Madrange Alix

Chargé de Projet : Nicolas Jouvin

Université d'Evry

1 avril 2025

Sommaire

Introduction

- Challenge de Machine Learning (apprentissage supervisé)
- Données de type single-cell RNA-seq
- Jeu de données de test privé
- Métrique du classement cachée

Notre jeu de données en quelques chiffres

- Jeu de données extrait de scMARK : 100.000 cellules
- 13551 colonnes et 1000 observations
- Aucune donnée manquante
- 4 types de cellules possibles : T_cells_CD4+, T_cells_CD8+, Cancer_cells, NK_cells

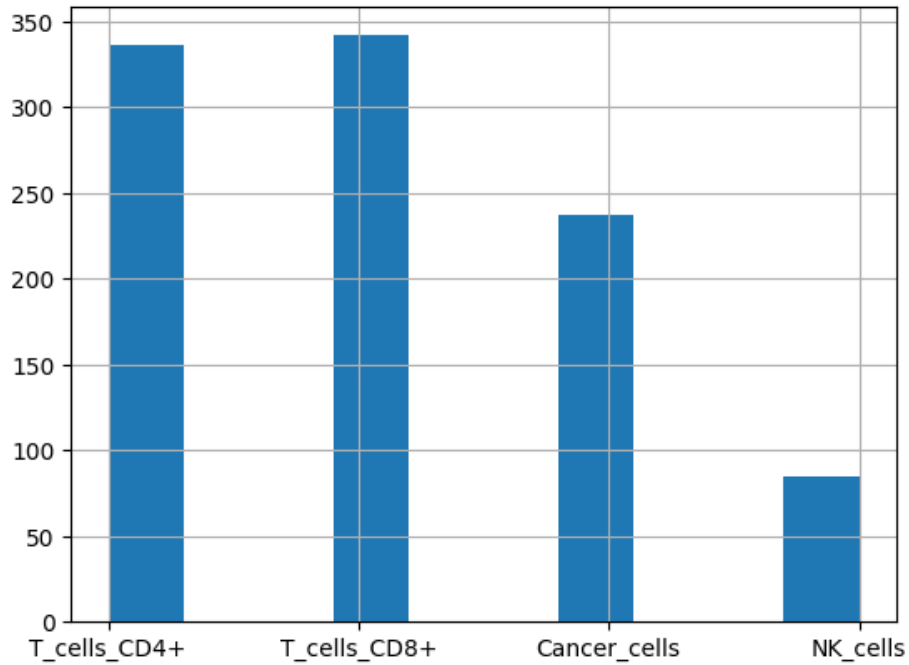

Figure 1 – Histogramme du jeu de données

Transformation des données

- Transformation $log1p(X) = \log(X + 1)$
- Réduire l'impact des valeurs extrêmes

Première idée de normalisation des données

Utilisation de `StandardScaler` de `scikit-learn`

- Pour chaque gène g dans la cellule c :

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

- $x_{g,c}$: valeur brute d'expression du gène g dans la cellule c
- μ_g : moyenne des valeurs d'expression du gène g à travers toutes les cellules
- σ_g : écart-type des valeurs d'expression du gène g à travers toutes les cellules

Deuxième idée de normalisation des données

Utilisation de `scanpy.pp.normalize_total` de Scanpy

- Pour chaque cellule c et chaque gène g :

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

- $\sum_{g'} x_{g',c}$: somme des expressions des gènes dans la cellule c
- total_count est une constante (généralement 10 000)

Modèles de classification supervisée

- Random Forest
- Gradient Boosting
- Support Vector Machine (SVM)
- Adaboost
- Régression Logistique

Méthode de test

Jeu de données pré-séparé en *train set* et *test set*.

Étapes :

- Transformation log1p et normalisation `StandardScaler`
- Séparation des données du *train set* en *train* et *valid set*
- Entraînement des modèles avec validation croisée sur le *train set*
- Évaluation des performances sur *valid* et *test* en utilisant la balanced accuracy

Filtrage par Variance

Élimination des gènes dont la variance d'expression est trop faible entre les échantillons

$\Longrightarrow$ Réduction de la dimensionnalité du jeu de données

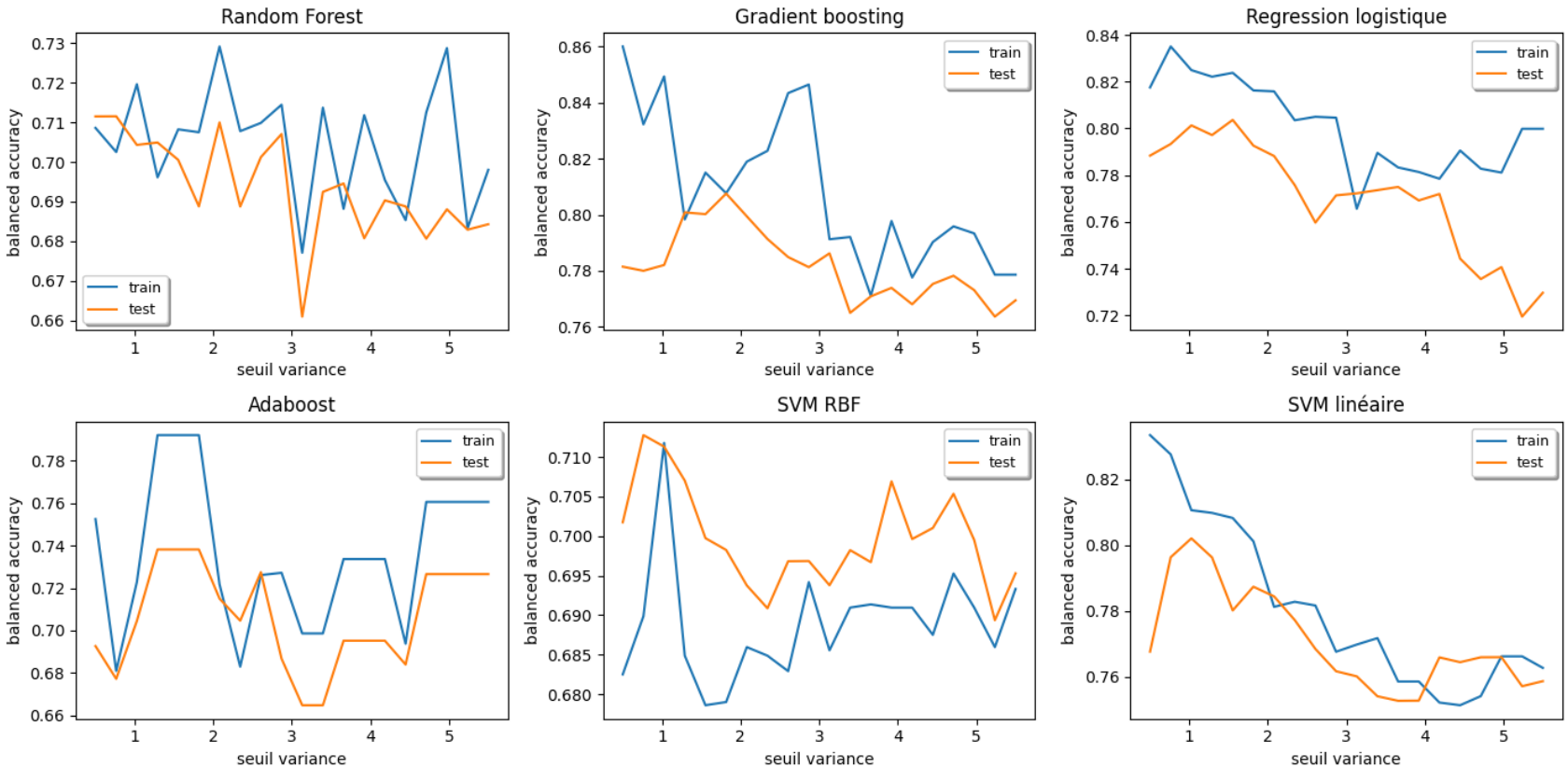

Figure 2 – Balanced accuracy en fonction de du seuil pour chaque modèle

Résultats de nos modèles

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

Régression LASSO

Optimisation de la robustesse et de la généralisation des modèles en éliminant les caractéristiques les moins pertinentes

Fonction de coût de la régression LASSO

$$J(\beta) = \frac{1}{2n} \sum_{i=1}^{n} (y_i - X_i \beta)^2 + \alpha \sum_{j=1}^{p} |\beta_j|$$

- $X \in \mathbb{R}^{n \times p}$: Matrice des variables explicatives.
- $y \in \mathbb{R}^n$: Vecteur des valeurs cibles.
- $\beta \in \mathbb{R}^p$: Vecteur des coefficients du modèle.
- $\alpha \geq 0$: Paramètre de régularisation qui contrôle la pénalisation des coefficients.

Hyperparamètre α

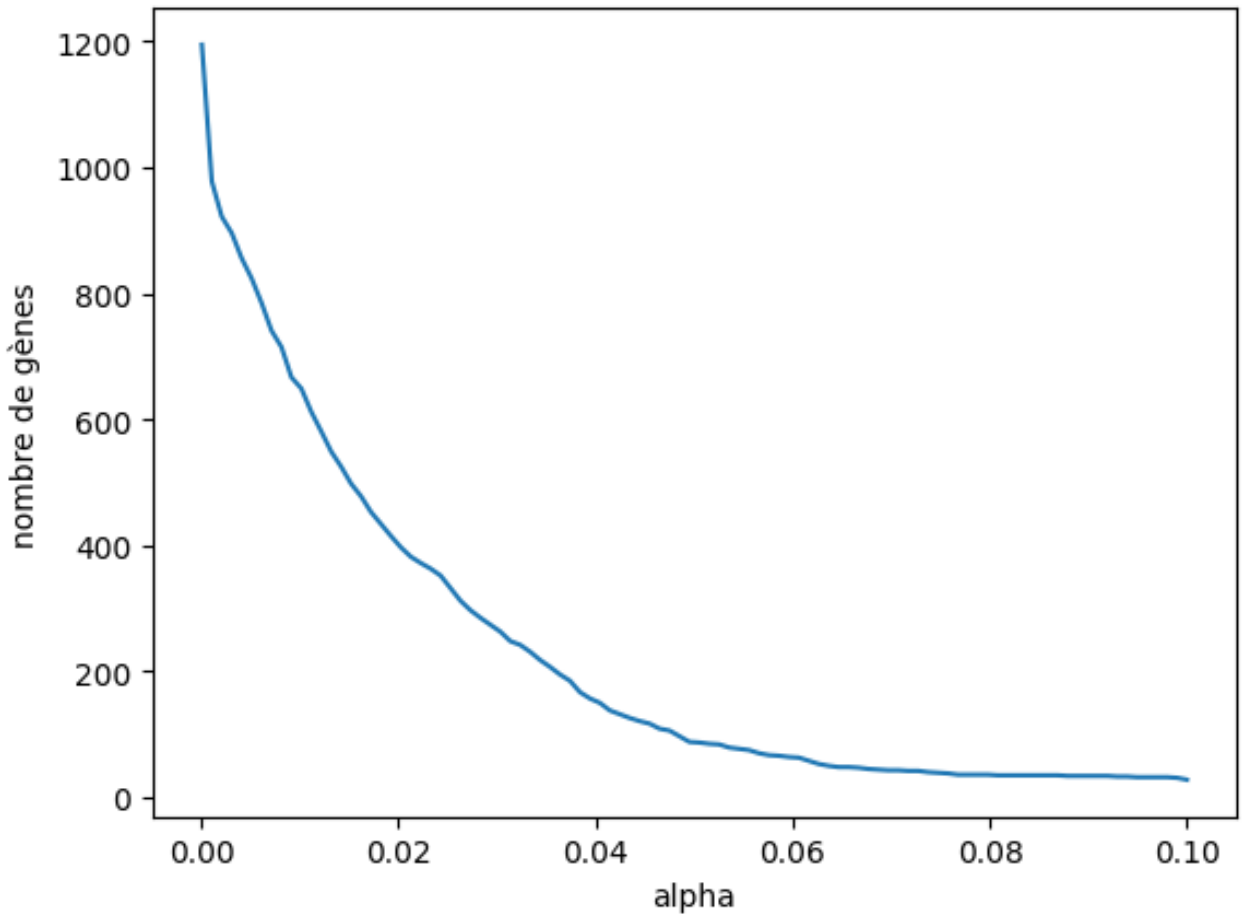

Figure 3 – Évolution du nombre de gènes sélectionnés en fonction de α

Performance des modèles

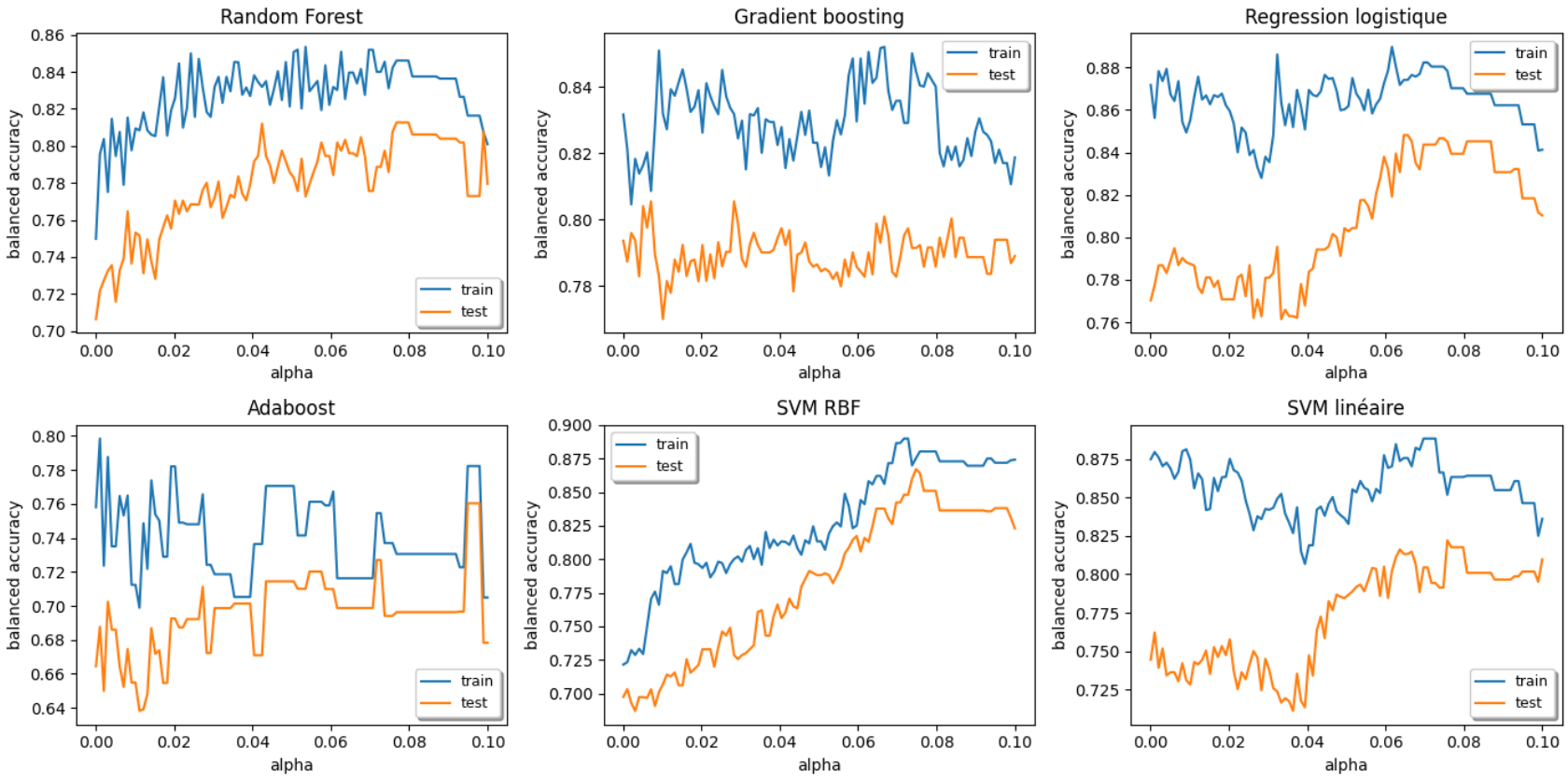

Figure 4 – Évolution de la balanced accuracy en fonction de α pour chaque modèle.

Résultats de nos modèles

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

Classement des méthodes sur RAMP

Classement	Modèle	Méthode	Hyperparamètres	BA
1	SVM RBF	LASSO	$\alpha = 0.005$	0.87
2	SVM RBF + LDA + Rég. log.	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

Réduction de dimension

$\longrightarrow$ Approche plutôt orientée classification non supervisée

Pourquoi ?

- **Approche largement étudiée** : La réduction de dimension est une technique largement utilisée dans la littérature scientifique.
- **Frugalité computationnelle** : Réduire la dimensionnalité permet d'accélérer l'entraînement des algorithmes.
- **Facilitation des visualisations** : En projetant des données en 2D ou 3D, il devient plus facile d'explorer et d'interpréter des structures sous-jacentes.
- **Réduction du bruit** : En éliminant les dimensions non pertinentes ou redondantes, on peut obtenir une représentation plus pertinente des données.

Méthodes testées

- Analyse en compasantes principales (ACP)
- t-SNE
- UMAP

ACP

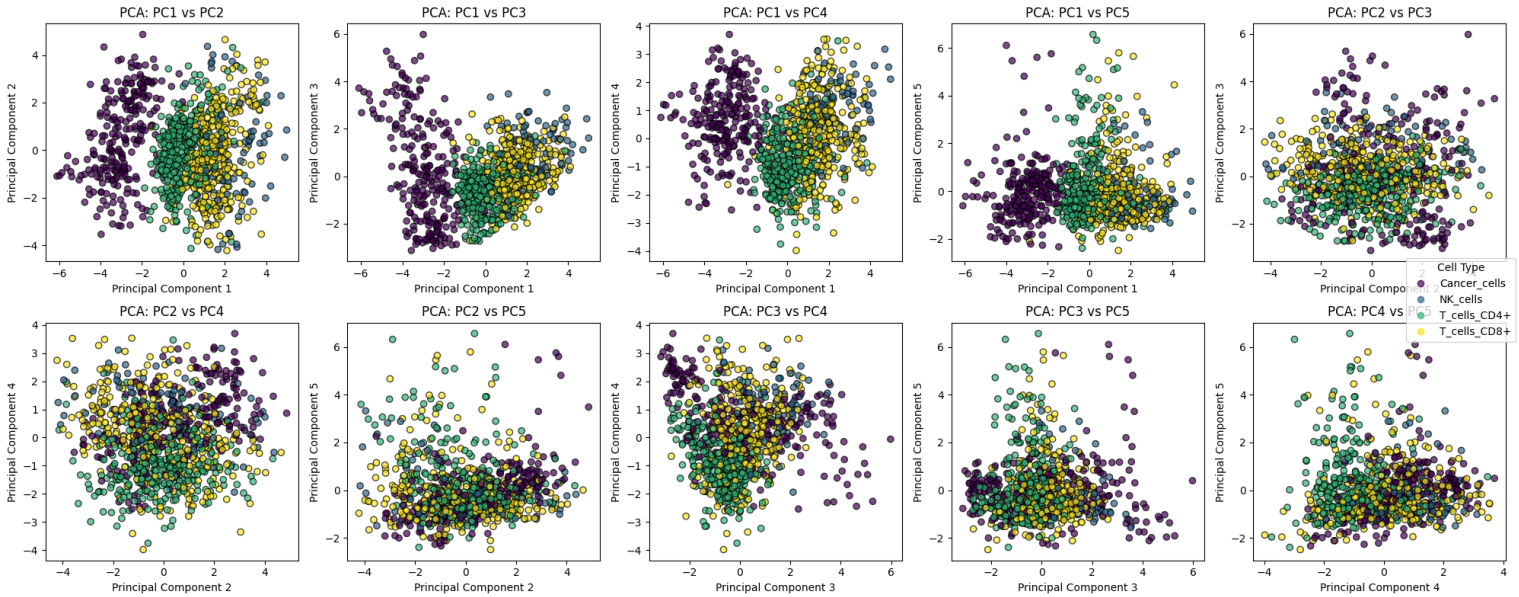

Figure 5 – Résultat de l'analyse en composantes principales

t-SNE 3D

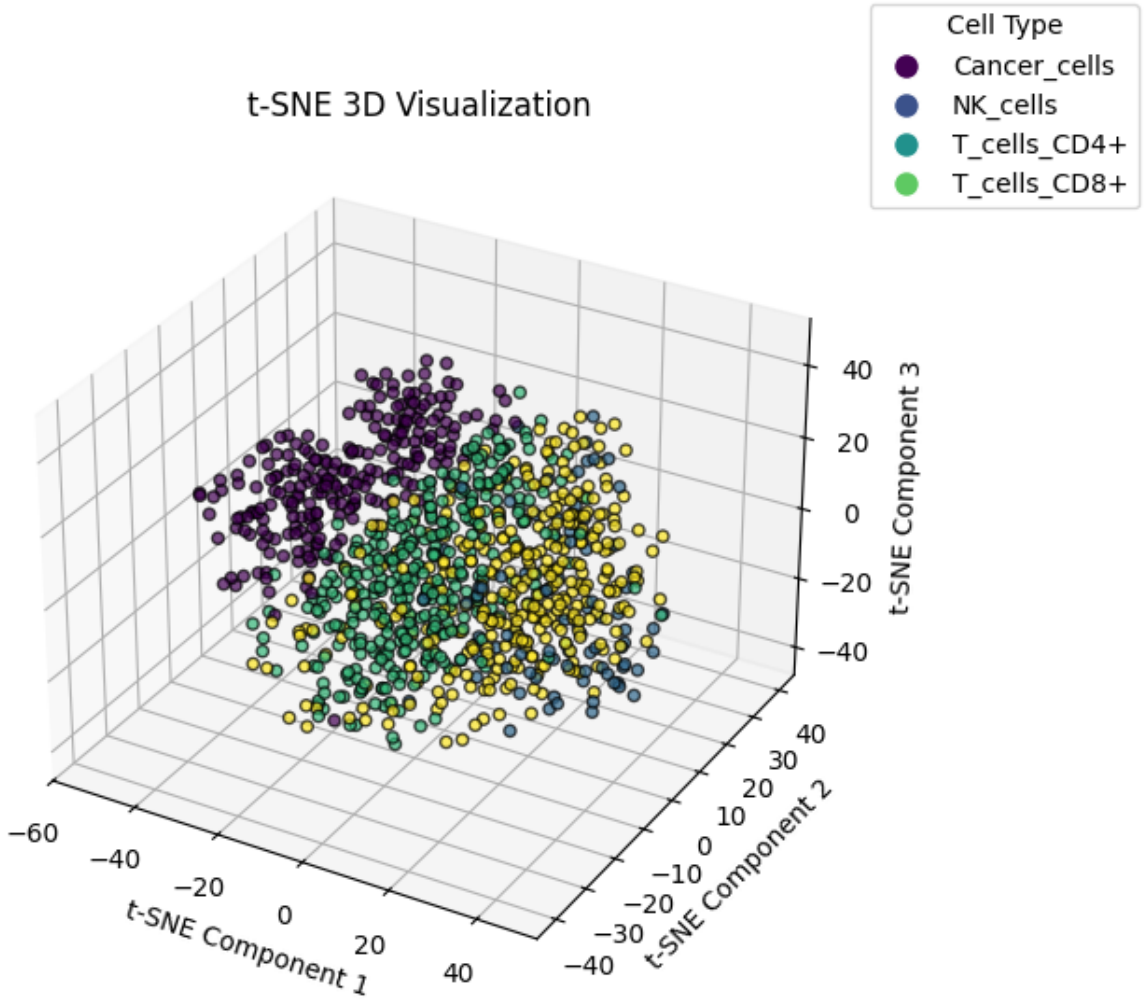

Figure 6 – Résultats de t-SNE 3D

UMAP

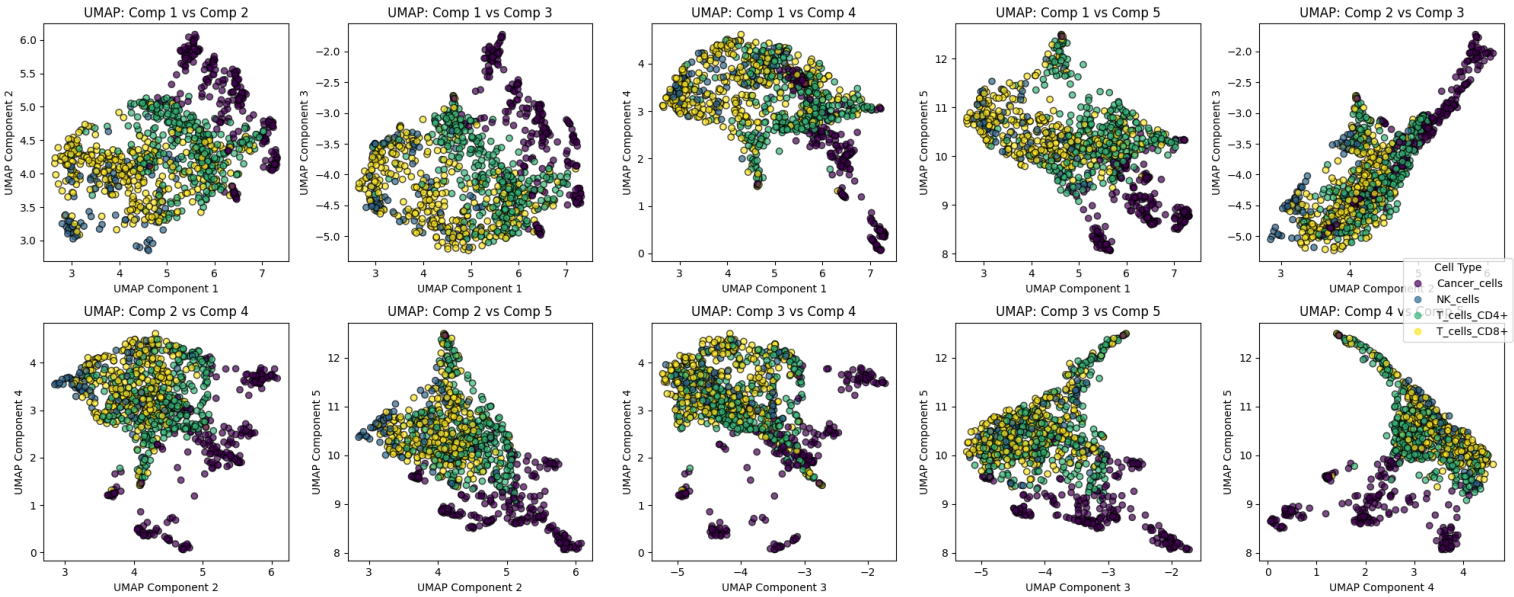

Figure 7 – Résultats de UMAP

Résultats

Malheureusement bien que les projections semblaient plutôt prometteuses, les modèles perdaient beaucoup en précision en utilisant ces méthodes de réduction de dimension.

$\longrightarrow$ Perte d'information inévitable avec ces méthodes.

Remarque : Certaines modèles ont malgré tout atteint des performances honorables allant jusqu'à 0.80 d'accuracy.

Conclusion

- Résultats satisfaisants
- 4ème place au classement final
- Des modèles performants et rapides
- Des pistes qui pourraient être intéressantes à explorer par la suite (TDA)