FACULTÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE

IFT799 - Science des données

TP-3&4

préparé par

Gabriel Gibeau Sanchez (gibg2501)

présenté à

Shengrui Wang

11 décembre 2020

Table des matières

Table des figures

1 Préambule

La figure 1 montre les informations contenues dans le fichier README.md fourni ci-joint. Il s'agit d'informations pertinentes pour l'exécution des scripts python.

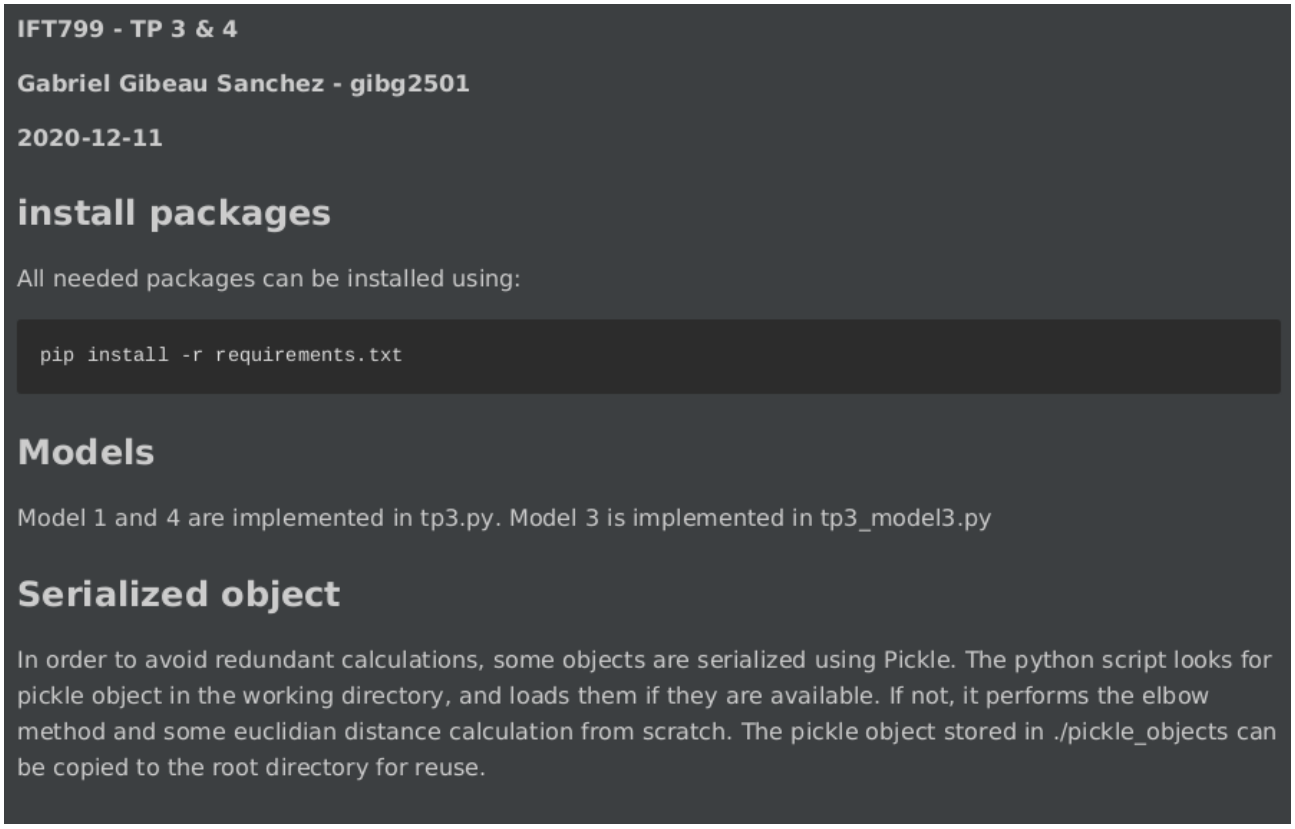
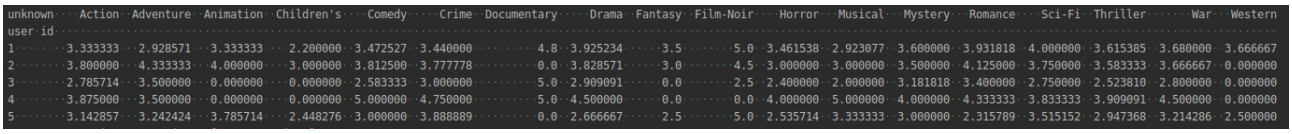


Figure 1 – Extrait du fichier README.md

2 Modèle 1 - Approche par *clustering*

2.1 Conception du modèle

Les données fournies dans les fichiers u#.base et u#.test se prêtent mal au *clustering*. Afin d'augmenter le nombre de dimensions à traiter, les informations contenues de ce fichier ont été recoupées avec les informations de u.item, à savoir les genres attribués à chaque film. De cette manière, une matrice contenant la moyenne des évaluations pour chacun des 18 genres de chaque usager à été générée et utilisée pour le *clustering*. À noter que la catégorie *unknown* n'est pas prise en compte puisqu'elle a seulement 2 occurrences dans l'ensemble des évaluations.



unknown	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary	Drama	Fantasy	Film-Noir	Horror	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western
user id																		
1	3.333333	2.928571	3.333333	2.200000	3.472527	3.440000	4.8	3.925234	3.5	5.0	3.461538	2.923077	3.600000	3.931818	4.000000	3.615385	3.680000	3.666667
2	3.800000	4.333333	4.000000	3.000000	3.812500	3.777778	0.0	3.828571	3.0	4.5	3.000000	3.000000	3.500000	4.125000	3.750000	3.583333	3.666667	0.000000
3	2.785714	3.500000	0.000000	0.000000	2.583333	3.000000	5.0	2.909091	0.0	2.5	2.400000	2.000000	3.181818	3.400000	2.750000	2.523810	2.800000	0.000000
4	3.875000	3.500000	0.000000	0.000000	5.000000	4.750000	5.0	4.500000	0.0	0.0	4.000000	5.000000	4.000000	4.333333	3.833333	3.909091	4.500000	0.000000
5	3.142857	3.242424	3.785714	2.448276	3.000000	3.888889	0.0	2.666667	2.5	5.0	2.535714	3.333333	3.000000	2.315789	3.515152	2.947368	3.214286	2.500000

Figure 2 – Extrait du tableau des moyennes d'évaluations par genre pour chaque usager

L'avantage de procédé de la sorte et qu'on travaille alors avec une matrice pleine. Par la suite, une technique de réduction de dimensionnalité peut être appliquée afin de ne conserver qu'un nombre pertinent de dimensions en regard de leurs variances. Dans notre cas, une *Single value decomposition* (SVD) révèle qu'au-delà de 8 composantes, les composantes supplémentaires ne causent même pas 5% de variance. Elles peuvent donc être rejetées. Un procède donc à un *clustering* sur les 8 premières composantes de la SVD. Évidement, il n'y aura pas de représentation graphique des *clusters* dans un espace à 8 dimensions.

2.2 Choix du nombre de *clusters*

Le choix du nombre optimal de *clusters* s'est fait à l'aide de la technique du coude (*elbow method*) et du score silhouette. La technique du coude consiste à calculer itérativement un score de "qualité" de *clustering*, en l'occurrence la somme de la distance des points à leur centroïde au carré (SSE), pour différentes valeurs de nombre de *clusters* et de tracer le graphique des scores successifs. On choisi alors le nombre de *clusters* associé au "coude" de la courbe, c'est-à-dire la où la pente change subitement pour s'aplanir, ce qui signifie qu'un nombre plus élevé de *clusters* n'influence plus le score de qualité du *clustering*. Cependant, la figure 3 montre qu'il n'est pas évident de trouver un "coude". La méthode de la silhouette a donc également été employée. Il s'agit d'un autre score de *clustering* qui prend en compte la distance aux autres centroïdes (valeurs allant vers -1) et aux frontières de décision (valeurs allant vers 0), en plus de la distance au centroïde du cluster (valeurs allant vers 1). La figure 5 illustre ces scores pour chaque *cluster* d'une itération avec le $4^{\text{ième}}$ jeu de données. Ici, le nombre de *clusters* optimal d'après le score silhouette est de 11.

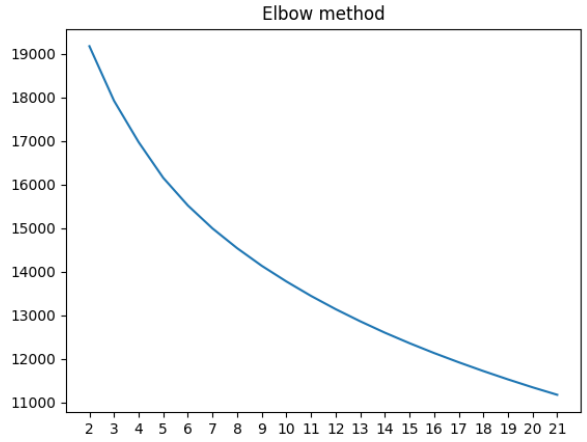


Figure 3 – Courbe des SSE en fonction du nombre de *cluster*

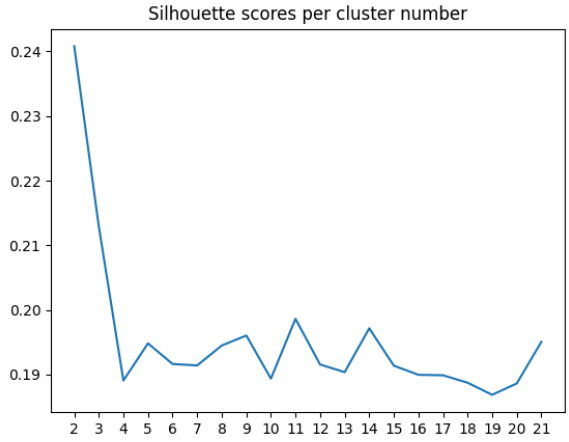


Figure 4 – Courbe des scores silhouette en fonction du nombre de *clusters*

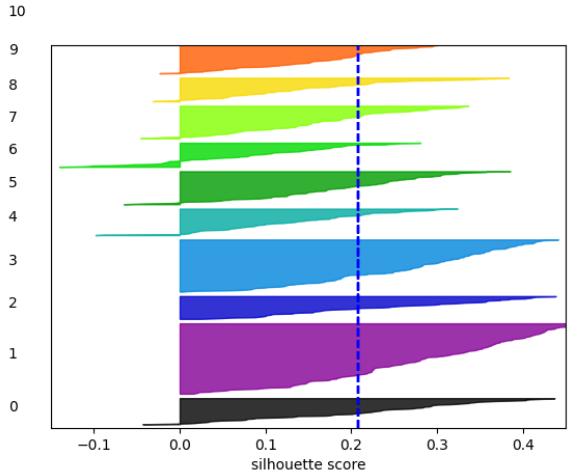


Figure 5 – Représentation graphique des scores de silhouette par *cluster* pour 11 *cluster*

Des valeurs allant jusqu'à 62 *clusters* ont été testées, mais n'ont pas produit de meilleurs résultats. La valeur du score silhouette de 3 à x *cluster* monte et se stabilise rapidement autour de 0.2 après 5 *clusters*. À 11 *clusters*, on maximise le score silhouette sur un intervalle de cluster raisonnable, et la pente de la courbe de SSE devient assez faible. C'est donc cette valeur qui a été retenue.

2.3 Résultats

Une fois les prédictions effectuées, il est possible de calculer l'écart entre ces dernières et les vraies évaluations des ensembles de tests en effectuant le calcul d'erreur RMS. Afin de valider la qualité des prédictions, ces valeurs d'erreur RMS ont ensuite été comparées aux erreurs RMS du pire cas possible, c'est-à-dire de tirer aléatoirement un score entre 1 et 5 pour chaque film et comparer le résultat avec les prédictions. La figure 6 montre les résultats des erreurs RMS des prédictions, tandis que la figure 7 montre les résultats de l'erreur causés par des évaluations aléatoires.

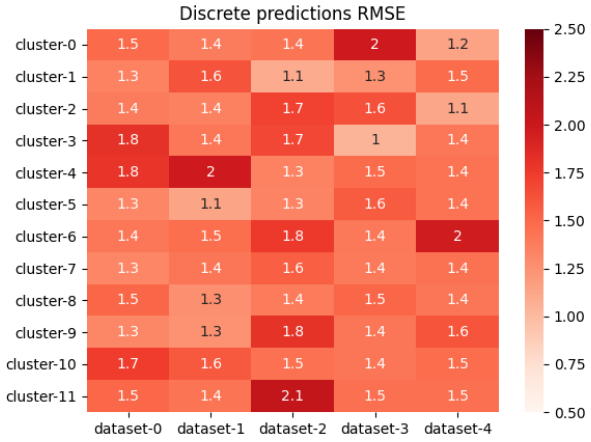


Figure 6 – Erreur RMS des prédictions par cluster et par *dataset*

Figure 7 – Erreur RMS d'un vecteur aléatoire par cluster et par *dataset*

On constate donc que la technique de *clustering* utilisée et meilleure pour suggérer des films que de simplement faire des suggestions au hasard. La figure 8 permet de visualiser la différence de performance moyenne entre ces deux modalités pour chaque *dataset*.

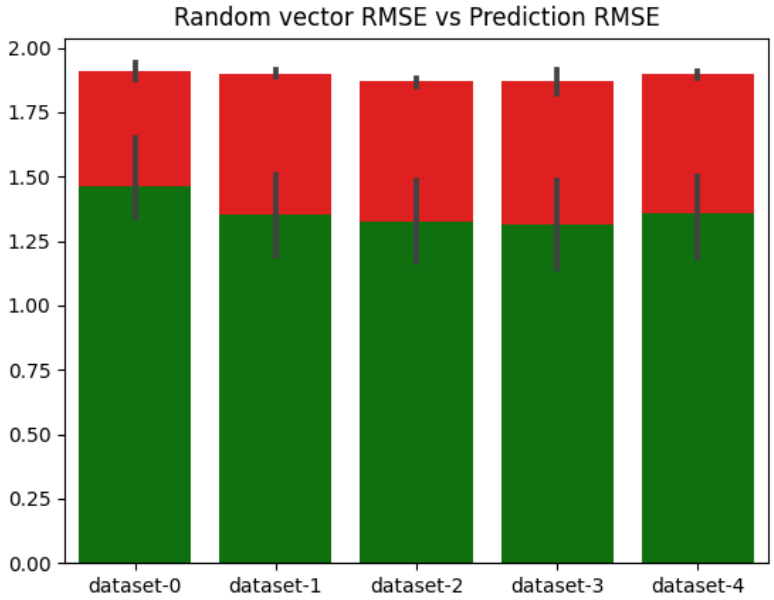


Figure 8 – KMean - Comparaison des erreurs RMS pour chaque *dataset*. En rouge : RMSE du vecteur aléatoire. En vert : RMSE des prédictions

3 Modèle 3 - Approche par *clustering* étendu

3.1 Conception du modèle

Ce modèle hérite directement plusieurs choix de conception du premier modèle. En particulier, le prétraitement des données est également conservé, à savoir la table des moyennes de genres de films par usager (voir fig. 2).

Les poids sont initialisés à 1 et sont mis à jour sur 100 itérations avant d'effectuer la prédiction finale. Ils sont décrémentés proportionnellement à la variance des dimensions d'intérêts, et pondèrent certaines évaluations à la baisse. Le coefficient C de normalisation a été choisi à 1.

3.2 Résultats

Les figures 9 et 10 montrent, encore une fois, les erreurs RMS moyennes par *cluster* et par *dataset*. On constate également qu'après 100 itérations, certains centroïdes ne sont jamais attribués dans certain *dataset* : il s'agit des cases blanches dans les *heatmaps*.

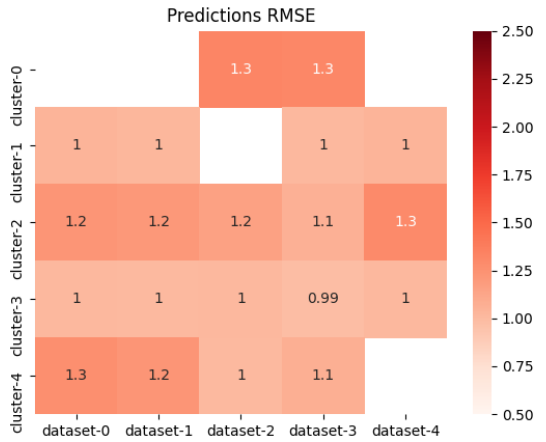


Figure 9 – Erreur RMS des prédictions par cluster et par *dataset* - kmean étendu

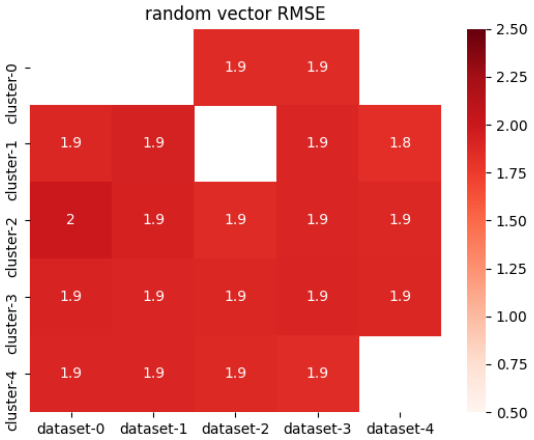


Figure 10 – Erreur RMS d'un vecteur aléatoire par cluster et par *dataset* - kmean étendu

Notons que le *heatmap* des erreurs RMS avec un vecteur de suggestions aléatoires est également troué puisqu'il se fit sur les mêmes index que les prédictions. Il en ressort néanmoins que, même en laissant tomber certains *clusters*, la méthode du *KMean* étendu semble non seulement plus performante que des suggestions aléatoires, mais également plus performantes que la méthode du *clustering* sur des dimensions réduites du 1er modèle. La figure suivante montre une comparaison graphique des performances du *KMean* étendu vs des suggestions aléatoires. Il est également intéressant de comparer les figures 8 et 11 pour constater le gain de performance avec le 3 ^{ième} modèle.

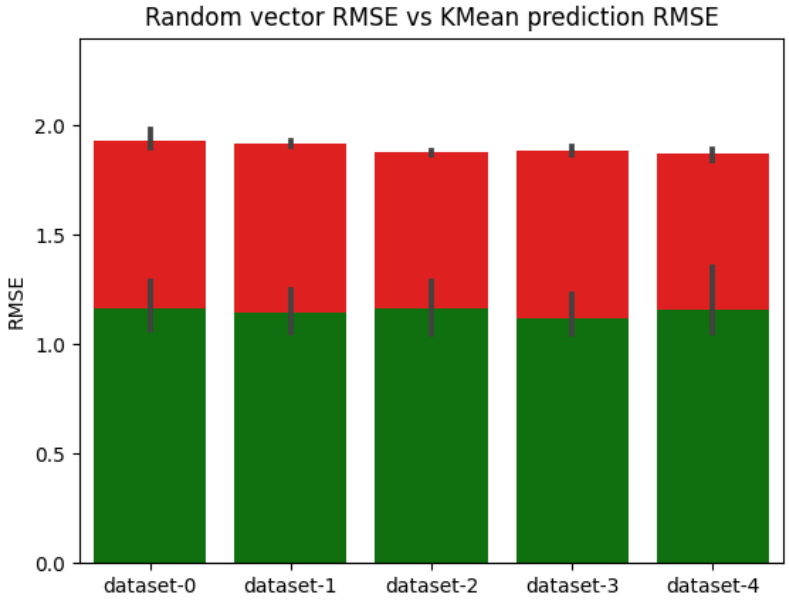


Figure 11 – KMean étendu - Comparaison des erreurs RMS pour chaque *dataset*. En rouge : RMSE du vecteur aléatoire. En vert : RMSE des prédictions

4 Modèle 4 - Approche par filtrage collaboratif

Pour la 4e approche, la méthode du classement basé sur les usagers (*user-based classification* a été retenue. Elle présume que des usagers "semblables" auront des goûts "semblables".

4.1 Conception du modèle

La première étape pour la technique du filtrage collaboratif consistait à réorganiser les données dans une matrice dont les lignes sont les usagers, les colonnes sont les films et les éléments sont les évaluations. La figure suivante montre cette réorganisation.

	rating																		
item id	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
user id																			
1	5.0	3.0	4.0	3.0	3.0	5.0	4.0	1.0	5.0	3.0	2.0	5.0	5.0	5.0	5.0	5.0	3.0	4.0	5.0
2	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	4.0	4.0	0.0	0.0	0.0	0.0	3.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	4.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0

Figure 12 – Tableau réorganisé des évaluations

Pour ce modèle, il a été décidé de normaliser les évaluations pour chaque usager. Ce puisque,

si un usager donné n'a jamais coté un film plus haut que 4, et que son voisin cote des films jusqu'à 5, on peut alors estimer que le niveau d'appréciation d'une évaluation de 4 du premier usager correspond à une évaluation de 5 du second usager. Par la suite on calcule la distance euclidienne entre usager pour chaque paire d'usagers de test et d'usager de base. Parallèlement, une table contenant les indexes des x plus proches voisins de chaque usager de test et conservée. Cette étape est lourde au niveau computationnel, et afin d'accélérer le processus ces tables sont sauvegardées et réutilisées par le script Python fourni avec ce rapport.

Avant de procéder au vote, il faut déterminer les poids associés à chaque plus proche voisin. À cet effet, la fonction de similarité de cosinus a été choisie. Ainsi, pour chaque usager de test, sa similarité cosinus avec ses x plus proches voisins est calculé. Cette similarité sert ensuite de coefficient pour pondérer le vote de son usager respectif.

Finalement, afin de refléter l'importance du nombre d'évaluations, un dernier ajustement est fait sur chaque poids : il est multiplié par un ratio entre 0 et 1 déterminé par un nombre seuil minimal γ d'évaluations commune entre l'usager de test et chaque voisin. Par exemple, dans notre cas ce seuil est fixé à 5. Si un voisin a seulement 3 films évalués en commun avec l'usager de test ($I_{uv} = 3$), alors le poids de ses votes (jusque là déterminé par sa similarité cosinus) sera ajusté comme suit : $w' = w * min(I_{uv}, \gamma)/\gamma = w * min(3, 5)/5 = w * \frac{3}{5}$. On remarque donc qu'en dessous de ce seuil, le poid est pondéré à la baisse.

Enfin, les votes peuvent être calculés. Pour les fins de tests, seulement les votes sur des films cotés dans les données de test sont considérés. Les votes sont faits sur les données normalisées, et son reconverti dans la plage de valeur d'évaluation initiale, soit de 1 à 5. Des tests ont été effectués avec 10 et 40.

4.2 Résultats

À l'instar de la première approche, l'erreur RMS a été choisie pour évaluer la justesse des prédictions. Les figures 13 et 14 montrent les distributions des erreurs RMS des prédictions pour chacun des *dataset* pour 10 et 40 k voisins respectivement. On constate que pour 10 k voisins, la moyenne des erreurs RMS est supérieure à la moyenne pour 40 k voisins.

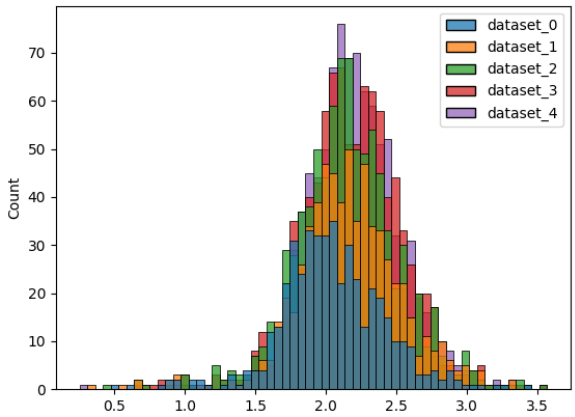


Figure 13 – Histogramme des erreurs RMS des prédictions pour chaque *dataset* - 10 k voisins

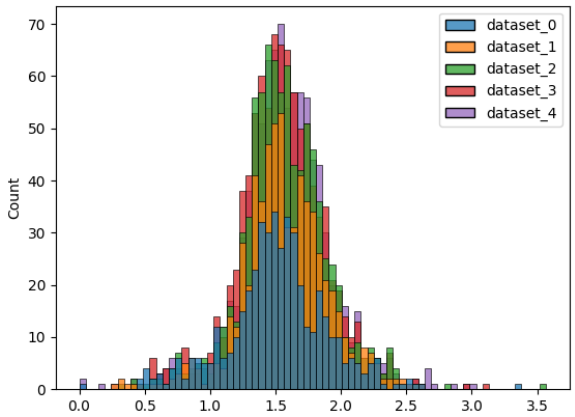


Figure 14 – Histogramme des erreurs RMS des prédictions pour chaque *dataset* - 40 k voisins

La comparaison avec les prédictions aléatoires nous permet d'avoir une meilleure perspective sur la performance du modèle utilisé avec 10 ou 40 k voisins. Ce qu'illustre les figures 15 et 16.

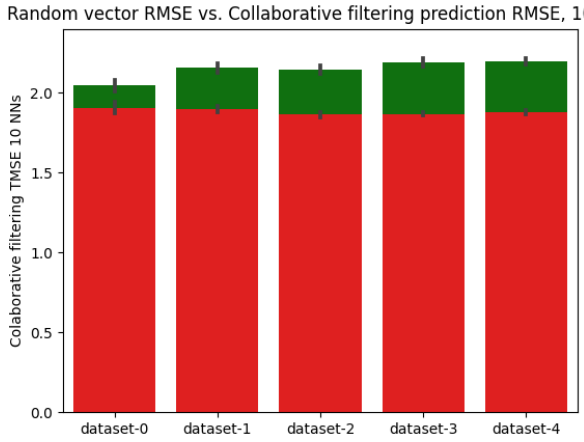


Figure 15 – Filtrage collaboratif 10 k voisins - Comparaison des erreurs RMS moyennes pour chaque *dataset*. En rouge : RMSE du vecteur aléatoire. En vert : RMSE des prédictions

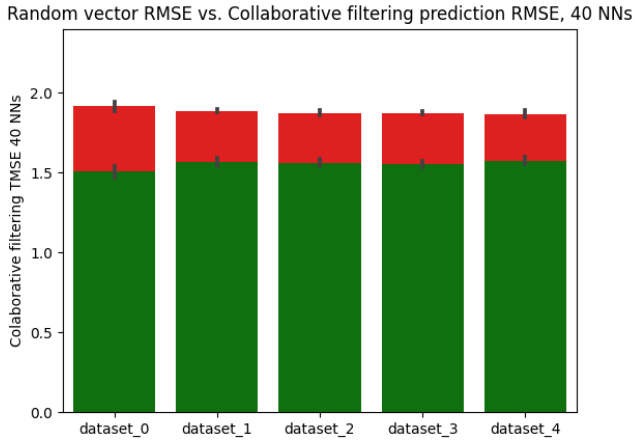


Figure 16 – Filtrage collaboratif 40 k voisins - Comparaison des erreurs RMS moyennes pour chaque *dataset*. En rouge : RMSE du vecteur aléatoire. En vert : RMSE des prédictions

On constate ainsi que pour 10 k voisins, la performance de prédictions et comparable à des

prédictions aléatoires, alors que pour 40 k voisins la performance est comparable à celle de l'approche par *clustering*.

5 Conclusion

Bien que le 3$^{\text{ième}}$ modèle semble plus performant, je serais hésitant à recommander cette méthode puisque son comportement me laisse un peu perplexe. J'aurais été tenté de le tester davantage si j'avais eu plus de temps, notamment à l'aide d'une méthode de détermination du nombre de *clusters* automatique. Je suis satisfait des modèles 1 et 4, qui réagissent selon mes attentes et me semblent assez instinctifs à comprendre. Je crois qu'ils pourraient facilement être utilisés pour faire de vraies recommandations.

Je crois cependant que d'autres indices que qualité de *clustering* que le score silhouette pourrait être employé pour la méthode du coude pour le modèle 1. Également, faute de capacité de calcul je n'ai pas pu faire toutes les *cross-validation* que j'aurais voulu sur le modèle 4. J'ai été en mesure de tester adéquatement seulement 2 valeurs de k voisins, alors qu'il aurait fallu en tester davantage. Par ailleurs il aurait été possible de manipuler d'autres hyperparamètres ou le nombre d'itérations, comme le seuil I_uv, pour améliorer la performance du modèle.