Travail 1

IFT3700
Introduction à la science des données

TRAVAIL PRÉSENTÉ À
Alain Tapp

22 Décembre 2022
Par : Hugo Carrier 20197563

1. Introduction

Nous avions comme mandat de proposer deux notions de similarités et de les tester avec divers algorithmes. Le code créer afin pour résoudre le problème est présent sur https://github.com/3Pi1416/IFT3700-A-A22-TP1.

Dans le rapport ci-présent, la base de données adulte est utilisée pour tester notre première notion de similarité. Notre analyse sur notre deuxième notion de similarité originale utilise la base de données mnist. De plus, dans ce dernier, une comparaison sera faite avec des résultats obtenus grâce à une similarité calculée avec la distance euclidienne.

Il est important de mentionner que les certains résultats obtenus tels que les calculs d'erreur et des v-mesure sont présentés dans les annexes, mais que tous les résultats obtenus sont dans des fichiers joints à la remise.

2. Méthodologie globale

2.1. Préparation des données

Pour bien réussir une analyse, il est important de bien manipuler les données. Nous utilisons des métriques « 'precomputed' » avec la librairie sklearn, nous devons donc faire attention à la manière dont nous traitons le format. Premièrement, les similarités et dissimilarités calculées sont sous forme matricielle et nous avons besoin de les séparer pour les prochaines étapes. Lorsque nous les séparons, nous utilisons la méthode train_test_split de python. L'option « shuffle » doit être mis à faux, car sinon, il sera difficile de trouver les colonnes liées aux données pour l'apprentissage du modèle avec les données pour tester les données. Ceci est causé par le fait que nous voulons utiliser des données différentes pour l'apprentissage et les tests afin d'assurer une meilleure qualité pour le rapport et que lorsque nous calculons les similarités(et les dissimilarités), nous obtenons des matrices carrées. Si

quelqu'un voulait répéter notre procédure en voulant ajouter de l'aléatoire, il doit s'assurer de mélanger les données avant de faire les matrices de dissimilarités.

2.2. K plus porche voisin

Pour cette approche, nous avons utilisé plusieurs valeurs de k pour tester notre modèle. Celle-ci nommé hyperparamètre doit être fixé par nous. Nous les avons fait varier entre 2 et 7.

Le but du projet était d'étudier la performance des algorithmes que nous avons créée. Nous avons décidé de prendre la meilleure v-mesure obtenue grâce à la fonction v_measure_score. Cependant, nous aurions pu augmenter la validité de ce choix d'hyperparamètre en séparant les données une fois de plus, mais nous croyons que la question sera répondue avec seulement un vague d'apprentissage et de test, puis une comparaison des résultats.

2.3. K Médoïdes et partition binaire

Nous avons simplement suivi la méthodologie vue en classes en s'assurant que les données d'apprentissage soient différentes de ceux du test. Nous avons donc utilisé la variation basée sur la moyenne des distances pour la partition binaire. Pour les médoïdes, nous avons fixé les médoïdes selon la quantité de solutions possible.

2.4. Isomap et PCoA

Isomap et PCoA sont deux algorithmes de réduction de dimension. Nous allons principalement utiliser leur résultat pour visualiser les données. Par curiosité, nous avons testé un k plus proche voisin afin de les regrouper une fois la dimension réduite. D'autres méthodes auraient pu être appliquées afin de les regrouper. Étant donné que les résultats sont passés dans deux modèles, nous avons dû séparer les données en 3 groupes. Une partie pour l'apprentissage du premier modèle, une partie pour l'apprentissage du deuxième modèle et finalement le reste pour tester les modèles.

2.5. V-mesure

La mesure V est la moyenne harmonique entre l'homogénéité et la complétude.[1] Elle sera utilisée pour présenter la qualité de nos prédictions.

3. Adult

3.1. Les données

Pour obtenir les données de l'ensemble adulte, nous avons téléchargé celle-ci sur https://archive.ics.uci.edu/ml/datasets/adult. Selon la description, le but de ces données est de terminer si une personne fait plus de 50 000 $ par années. L'ensemble de données adultes fournies différentes statistique sur des gens. Nous avons 14 colonnes d'information afin de déterminer la tranche de salaire. Donc 15 colonnes aux totales. Certaines colonnes sont descriptives et d'autres contiennent des valeurs numériques continues. Les données contiennent quelques trous, mais c'est décrit comme assez propre selon la source.

3.2. Notions de similarité proposée.

i. Explication de nos choix

Nous avons approché le problème avec le désir de considérer le plus d'information possible sans en inférer. C'est-à-dire que nous voulions inclure l'information des 14 colonnes, mais ne pas devoir boucher les trous s'il en a. Après quelque recherche, nous sommes tombés sur la méthode Jaccard que nous avons trouvé intéressante. Or, afin de la rendre utilisable avec nos données et en même temps, proposer une version originale de cette similarité, nous avons dû modifier l'approche.

[1] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.v_measure_score.html

ii. Explication de la méthodologie

La méthode de Jaccard peut être facilement utilisée lorsque les données sont binaires ou lorsqu'elles sont des ensembles. Les nôtres ne le sont pas, alors nous devions penser à une idée intéressante pour résoudre ce problème. Initialement, nous avions pensé transformer les colonnes descriptives en binaire. Par exemple, la colonne sexe serait devenue deux colonnes : est homme et est femme. L'étape suivante aurait été de simplement comparer les colonnes descriptives et de diviser la somme des valeurs identiques par le nombre de colonnes. Après réflexion et quelques essaies, nous nous sommes rendu que nous avions rendu le problème plus que complexe que nécessaire et que les colonnes non modifiées contenant un grand domaine de valeurs descriptives fausseraient nos résultats, pas leur transformation en de nombreuses sous-colonnes. Au lieu de cela, nous avons implémenté notre première originalité à l'approche. Nous avons simplement comparé les valeurs des colonnes pour savoir si elles sont identiques. Par la suite nous avons calculé la somme des valeurs identique puis nous avons divisé par le nombre total de colonnes.

Cette façon considère que chacune des colonnes apporte la même valeur à notre similarité. Cependant, nous n'avons pas encore bien traiter les colonnes avec des nombres continus. Afin de les considérer, nous avons pensé à une autre idée originale pour résoudre le problème. Nous avons simplement considéré la distance absolue et rapporté cette valeur en pourcentage.

Pour y arriver, nous avons trouvé le maximum et le minimum de chacune des colonnes pour calculer la distance maximale. Par la suite, nous calculons la distance absolue entre nos points puis nous divisons par la distance des extremums de cette colonne. Ce résultat donne une dissimilarité. Si deux chiffres sont identiques, la valeur obtenue sera 0. Pour trouver une similarité, il suffit de faire 1 moins la valeur obtenue. Ainsi nous avons une valeur qui se

retrouve entre 0 et 1, ce qui nous permet de considérer l'information de chacune des colonnes à parts égales.

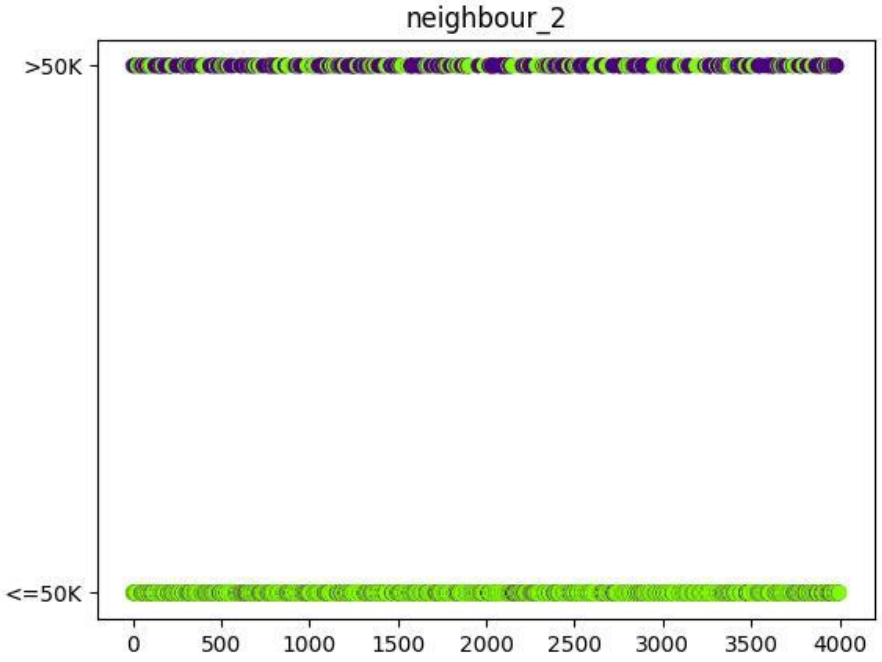
Au final, le résultat total sera entre 0 et 1 dû à la division du nombre de colonnes. Si deux valeurs sont identiques, nous obtenons 1 comme similarité, plus elles sont différentes, plus leur similarité tendra vers 0. Pour calculer la dissimilarité, nous avons fait 1 – la similarité. Avec les similarités et les dissimilarités calculées, nous avons par la suite poursuivi l'analyse en utilisant les résultats précédents avec les diverses méthodologies demandées dans le devoir.

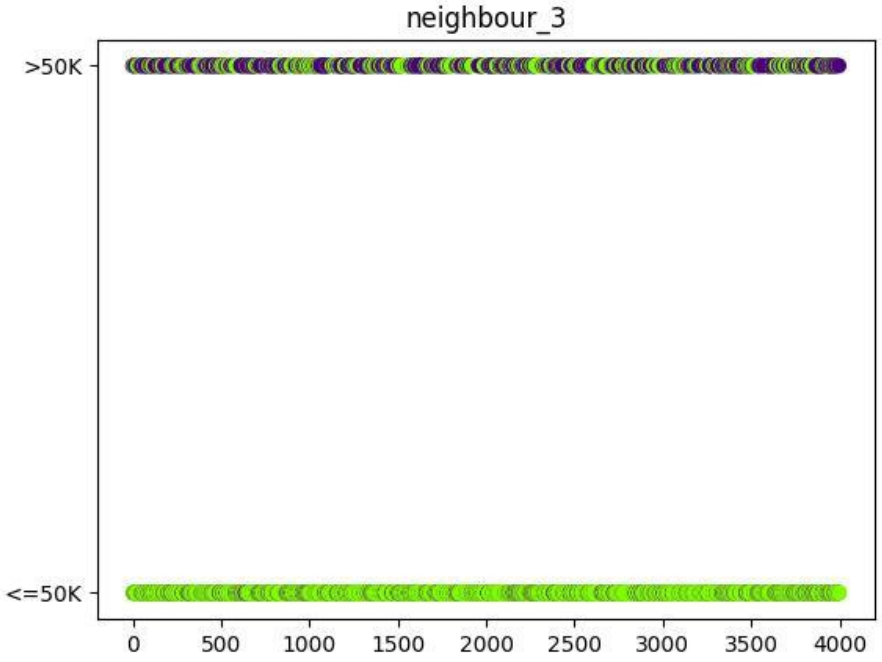
3.3. Présentation des résultats

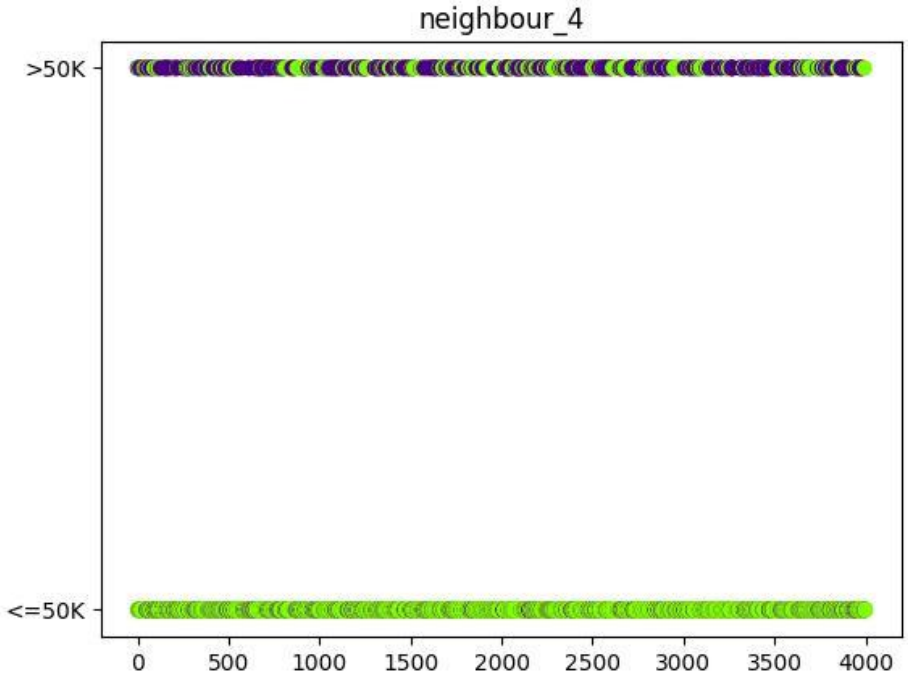
Pour tous les résultats qui suivent, vert signifie que la vraie valeur est"<=50K" et mauve est ">50K". Nous avons aussi fait l'analyse en prenant seulement 5000, 80 % de celle-ci pour tester et 20 % pour l'apprentissage des modèles.

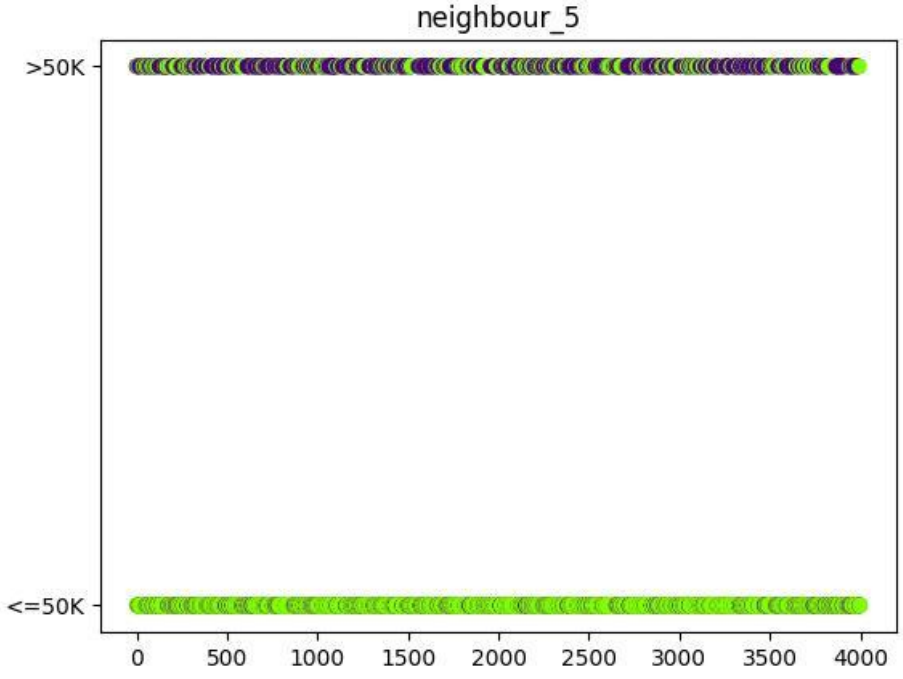
i. K- voisin :

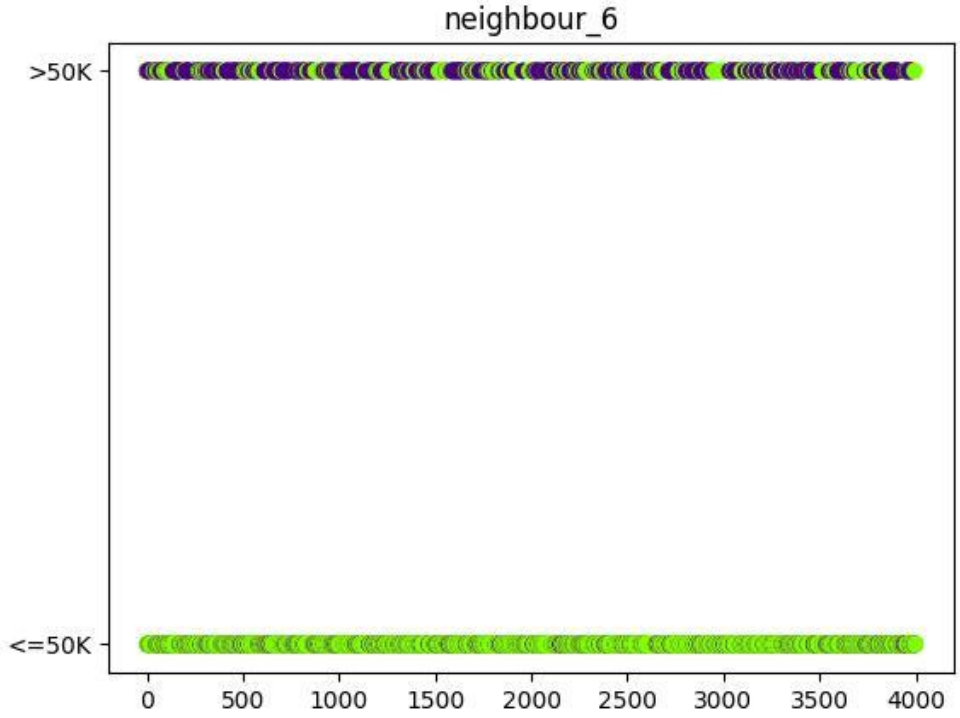
Voici la présentation de différent résultat obtenu pour les k voisins.

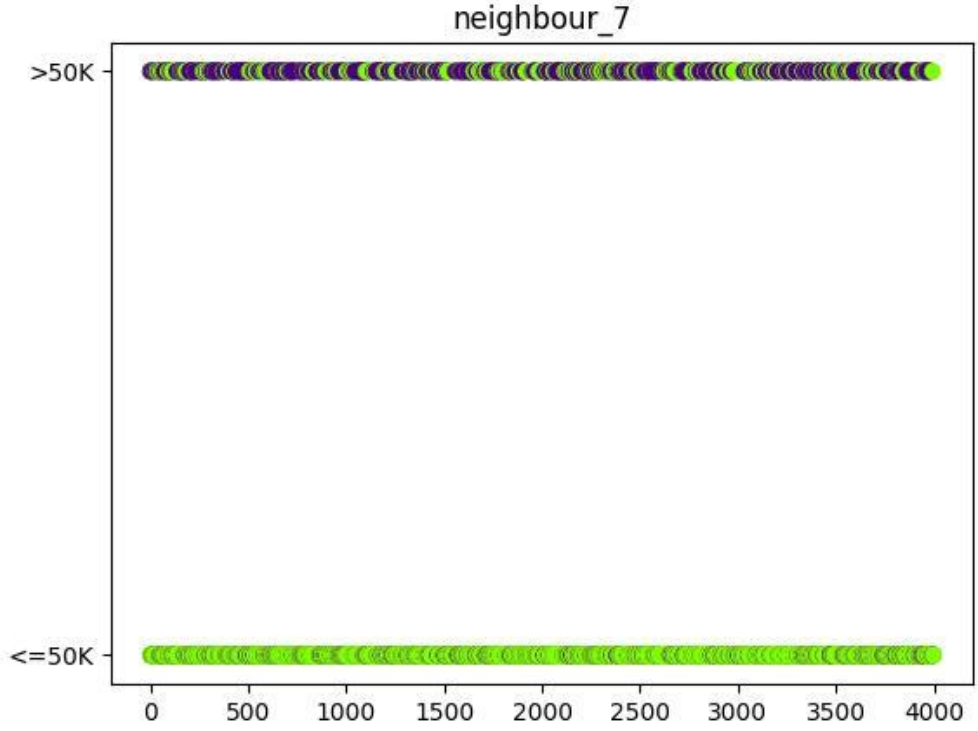





neighbour_4
>50K
<=50K
0
500
1000
1500
2000
2500
3000
3500
4000

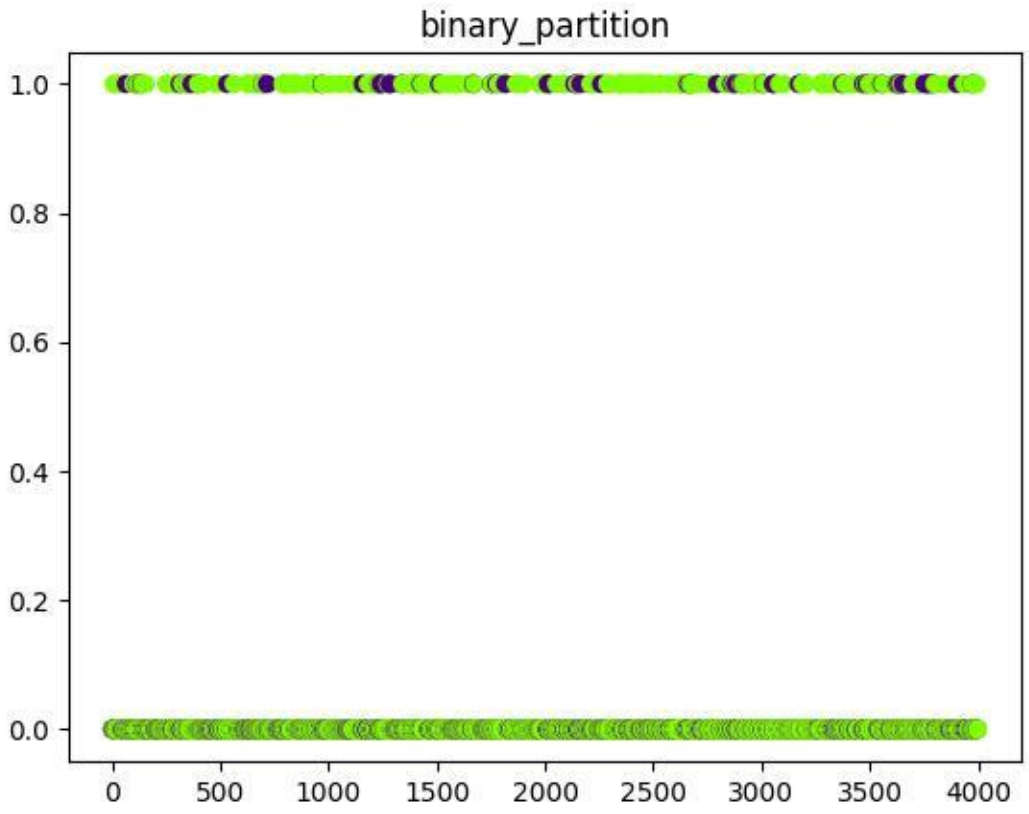

neighbour_5
>50K
<=50K
0
500
1000
1500
2000
2500
3000
3500
4000

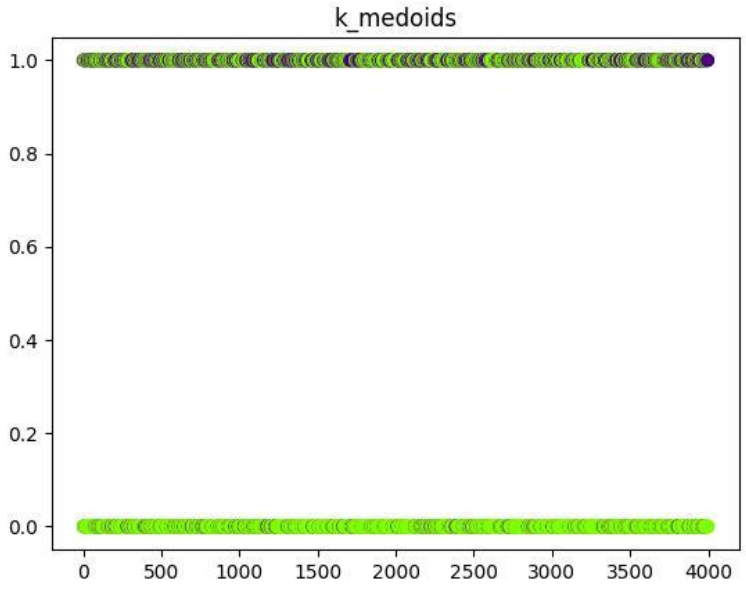

neighbour_6
>50K
<=50K
0
500
1000
1500
2000
2500
3000
3500
4000


neighbour_7
>50K
<=50K
0
500
1000
1500
2000
2500
3000
3500
4000

Visuellement nous pouvions penser que nous avons quand même bien réussi à séparer les données. Cependant, lorsque nous faisons l'analyse avec les chiffres, on se rend compte que notre modèle n'apporte pas beaucoup de valeur (vois annexe 6.1 pour tous les résultats). Nous avons calculé la v-mesure et le nombre d'erreurs. Selon la v-mesure. Le modèle avec k = 7 est le meilleur pour une valeur de la v-mesure d'environs 0.17. Nous avons un total de 768 erreurs sur 4000 valeurs.

ii. Partition binaire et k médoïdes





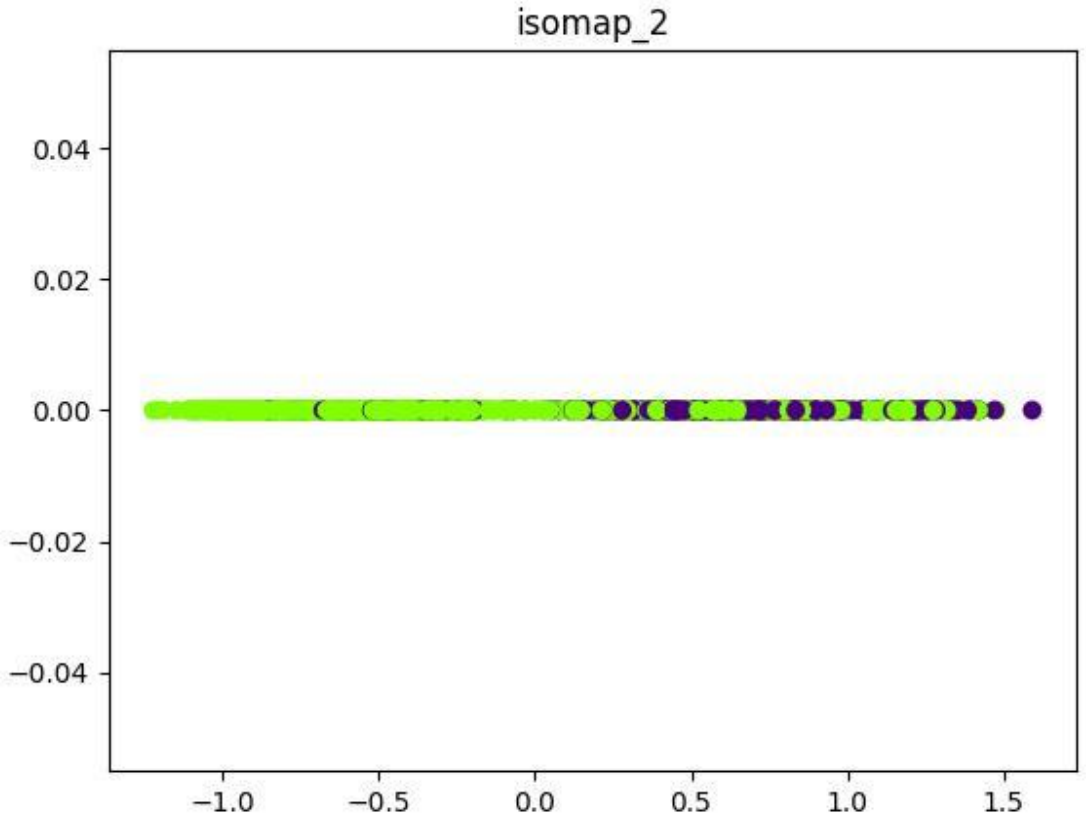
Visuellement, les résultats des deux modèles ne semblent pas concluants. Comparativement à k voisin, ils ont l'air d'avoir moins bien réussi. La ligne du haut représentant probablement ceux qui ont plus de 50 000 et elle est parsemée de couleur vert. Rappelons-nous que la couleur verte signifie moins de 50 000.

Lorsqu'on compare avec la v-mesure, nous obtenons environs 0.0001 pour la partition binaire et environs 0.13 pour k médoïdes. L'approche de partition binaire ne semble pas du tout concluante pour ce type de similitude. K médoïdes semble apporter une légère amélioration lorsqu'on évalue sa v-mesure. En revanche, sa quantité d'erreurs totales est de 2828 sur 4000. Ainsi, les k plus proches voisins semblent être une bien meilleure méthodologie pour nos données.

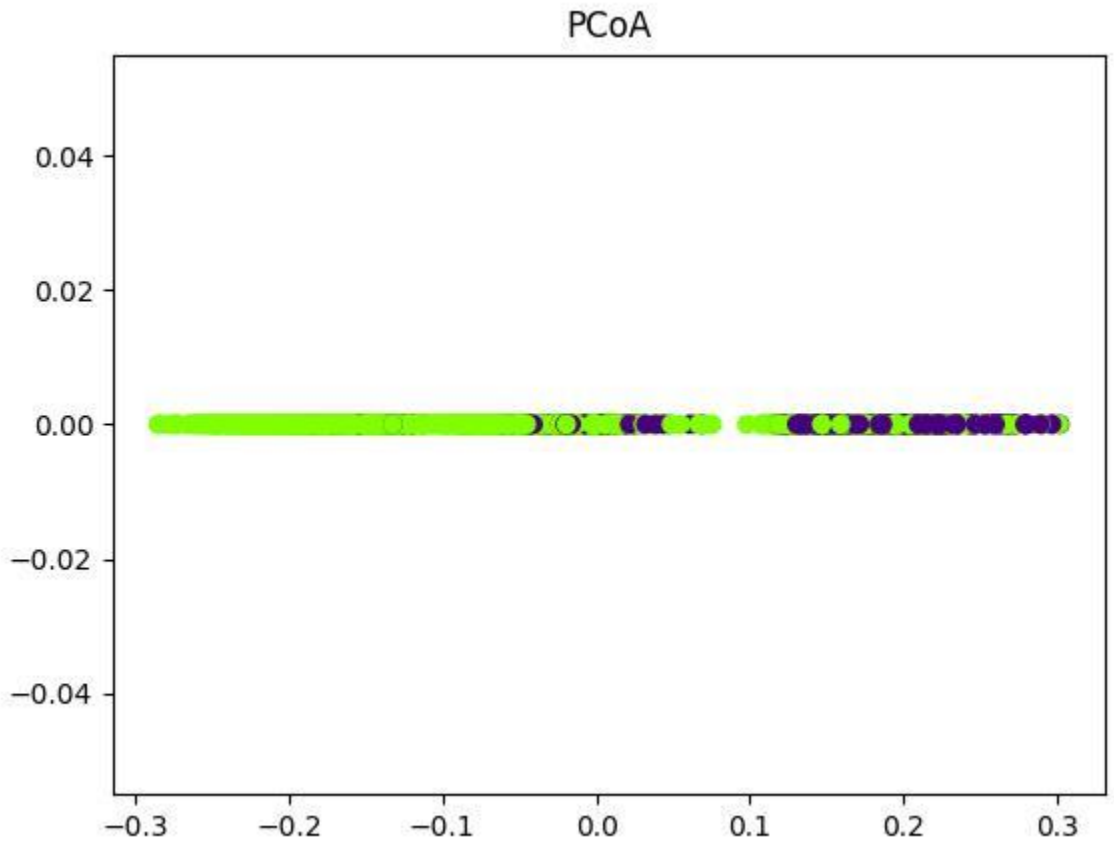
iii. Isomap et PCoA

Isomap et PCoA sont deux approches qui permettent de visualiser les similitudes grâce à une réduction de dimensionnalité.

Nous avons pour isomap :



Pour PCoA :



Visuellement, on remarque quand même une séparation de couleur. PCoA semble avoir moins de couleurs mélangées, mais cette analyse est assez subjective.

Lorsque nous avons mis ces réductions de dimensionnalité dans un k plus proche voisin, nous n'avons pas obtenu de bon résultat. Les meilleures valeurs de la v-mesure tournent autour de 0.026 (voir annexe 6.1). Ainsi on peut conclure que notre similitude ne sépare pas tout à fait assez bien les données dans leur groupe respectif. Visuellement, nous remarquons beaucoup de points verts mélangés aux mauves, donc il y a une mauvaise séparation.

3.4. Discussion de l'approche

Notre approche est loin d'être parfaite. Nous le voyons par les résultats un peu désastreux. Il y a plusieurs problèmes. Premièrement, en python, notre méthodologie prend du temps à être calculée. Nous avons utilisé un ensemble de 5000 données, 1000 pour l'apprentissage et 4000 pour tester le modèle. Il nous prit 30 minutes pour faire la matrice de dissimilarités. Elle pourrait probablement être optimisée dans des langages plus performants, mais pour python ceci est un gros problème. Nous avons, plus tard, testé le programme avec l'ensemble des données offert par l'importation. Le code prit plus d'une journée pour s'exécuter pour finalement obtenir des résultats similaires. Par exemple, avec cinq voisins et toutes les données incluses dans l'analyse, nous avons obtenu comme résultat une v-mesure de 0.19.

Un autre problème c'est qu'elle traite chacune des colonnes égales. Il est possible de voir cela comme une qualité, car le modèle reste simple, mais c'est aussi un défaut. Il est probable que certaines colonnes apportent plus d'information que d'autres et notre modèle ignore cette possibilité. Il serait aussi possible d'augmenter la qualité des résultats en augmentant la complexité des ressemblances. Dans notre cas, les valeurs descriptives sont égales ou non. Or, certaines valeurs descriptives ont des ressemblances entre elles. Par exemple le niveau l'éducation ont un ordre. La différence entre une maitrise et un bac n'est pas la même qu'entre une maitrise et une quatrième année,

La plupart de ces problèmes sont causés par un choix que nous avons fait. Nous avons désiré un modèle simple et facilement réplicable. Notre méthodologie peut être facilement répliquée et même utilisée sur d'autres types de données. En science de données, c'est un gros avantage, car il en devient facile de vérifier les conclusions que nous avons apportées.

Un autre avantage c'est que nous n'avons pas nécessairement besoin de corriger les données. Le modèle fonctionne correctement lorsqu'une certaine information est manquante. Les informations manquantes sont vues comme des différences, mais étant donné que le modèle traite les colonnes à part égales, les résultats sont moins influencés par ces erreurs.

Pour conclure, la qualité des résultats montre que le modèle doit être modifié avant d'être réellement utilisable.

4. Mnist

4.1. Les données

Les données de mnist ont été obtenues grâce à la librairie keras.datasets de python. Elle nous donne accès à 70 000 données pour construire et tester notre modèle.

Une information importante à mentionner pour la présentation des prochaines mesures est que les algorithmes regroupent les chiffres en groupe allant de 0 à 9, mais la valeur associée ne correspond pas toujours au groupe. La vraie valeur connue va de 0 à 9. Les couleurs dans cet ordre sont indigo, noire, rouge, chocolat, or, chartreuse, turquoise, bleu royal et argent.

4.2. Notion de similarité proposée

i. Explication de nos choix

Pour bien réussir le devoir, nous devions nous assurer qu'une légère translation n'affecte pas trop les résultats. Personnellement, nous trouvions que comparer chacune des images entre elles pouvait apporter beaucoup de bruit. Il y a beaucoup de façons d'écrire un même chiffre. Ainsi, nous ne passons pas que la distance euclidienne entre les images était la meilleure idée.

ii. Explication de la méthodologie

Afin de résoudre le problème des translations, nous avons eu comme idée de comparer la distance euclidienne à une moyenne d'image. Nous avons donc commencé notre algorithme en utilisant 10 000 données pour créer des images moyennes de chacun des 10 chiffres. On obtient alors une image floue de chacun des chiffres. Par la suite, nous comparons le reste de nos images à celles-ci. Nous obtenons donc des vecteurs de distance entre les images et des images moyennes. Par la suite, nous comparons ces vecteurs entre eux pour obtenir une dissimilarité. L'idée derrière ce principe est que chacune des images ressemble à plusieurs chiffres. Pour visualiser cette idée, voici un exemple fictif : Un 7 peut être vu comme un 7, mais aussi comme un mélange de 2 et 1. Ce qui permet donc de différentier un 7 et 1 mal écrit. Ils pourraient avoir comme différence que le 7 à tendance à ressembler plus à 2 que le 1. Ainsi, cette dissimilarité utilise le fait qu'un nombre peut ressembler à un autre nombre pour mieux les distinguer.

Comme pour les données de la base de données adulte, nous avons utilisé 80 % des valeurs pour tester les modèles puis 20 % pour l'apprentissage sur les 60 000 valeurs restant après la moyenne calculée. En d'autres mots, les données ont été séparées en trois pour éviter tout surapprentissage.

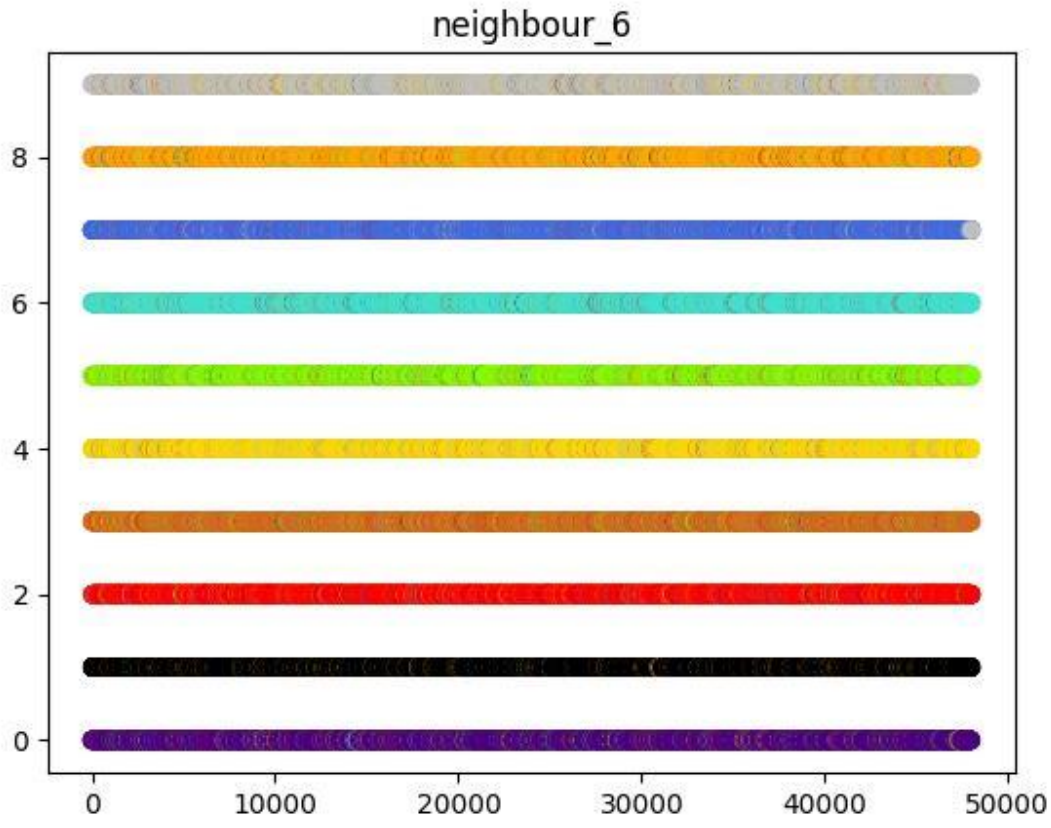
iii. Distance euclidienne

Afin de suivre les directives du devoir, nous avons précalculé les distances euclidiennes et nous les avons posées dans une matrice de dissimilarité. Nous avons utilisé les mêmes 60 000 valeurs qu'avec notre dissimilarité personnelle pour l'apprentissage et les tests.

4.3. Présentation des résultats

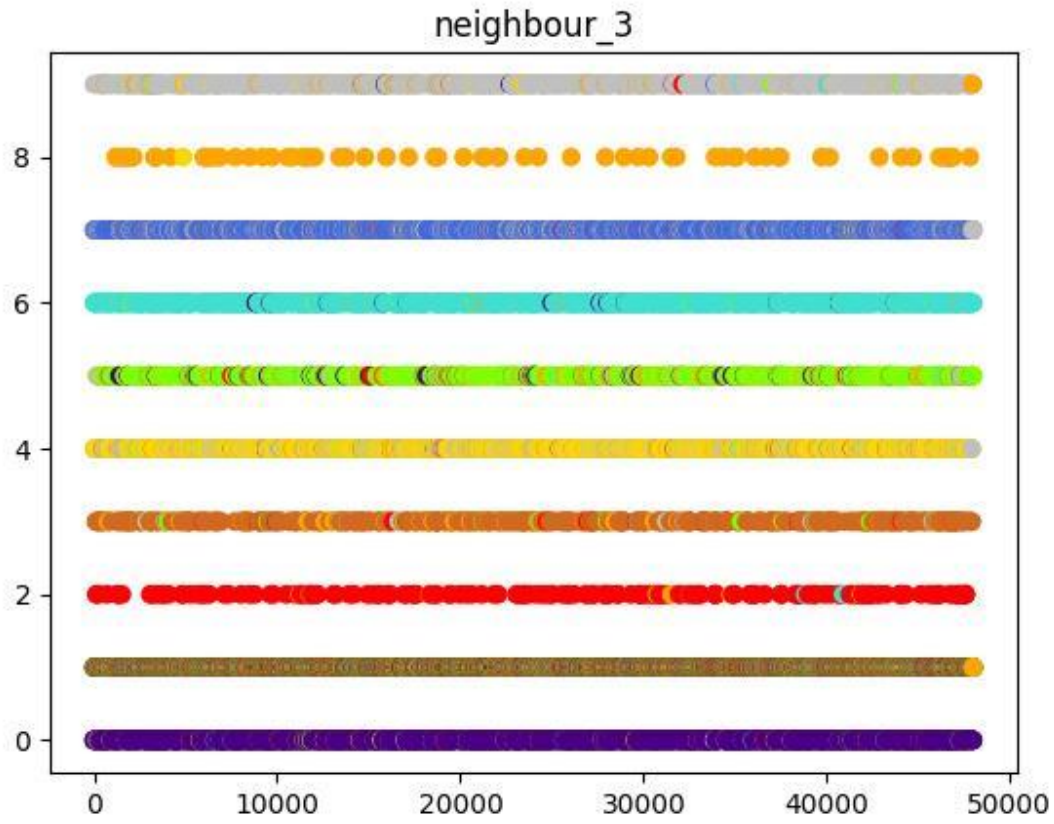
i. K- voisin :

Encore une fois, nous avons utilisé plusieurs valeurs de k. Pour notre algorithme personnel, la meilleure valeur obtenue comme v-mesure provenait d'un nombre de voisins égal à 6 avec une valeur d'environs 0.75.



Visuellement, il est facile de remarquer la qualité de la prédiction, nous avons 10 groupes de couleurs distinctes. Nous voyons aussi qu'elle n'est pas parfaite, mais on peut dire que notre algorithme à tout de bien fonctionner, surtout si on le compare avec les résultats de la distance euclidienne.

Nous avons refait la même procédure pour les distances euclidiennes. Le meilleur résultat est une v-mesure d'environs 0.28 avec 3 voisins.



On remarque tout de suite que certaines lignes manquent des données comparativement à notre dissimilarité et d'autre semble regrouper plusieurs valeurs différentes telles que le groupe 1, expliquant du fais même la piètre valeur pour la v-mesure de la distance euclidienne.

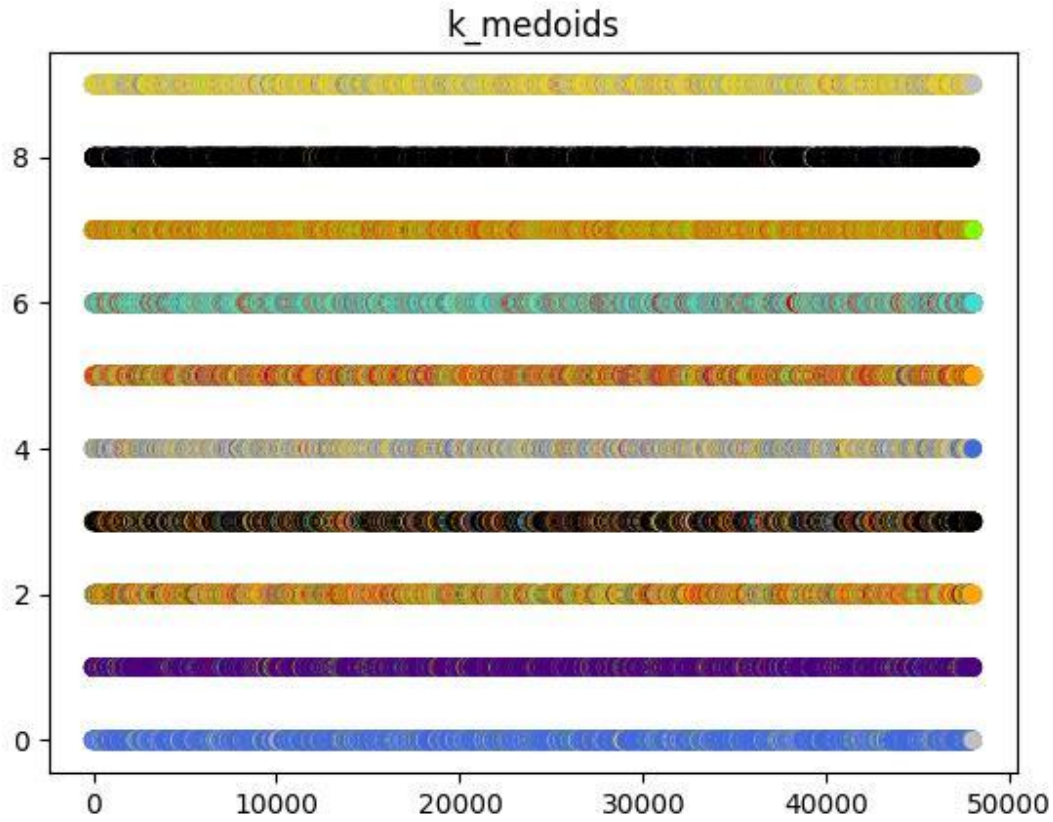
ii. Partition binaire et k médoïdes

Encore une fois, les résultats de dissimilarité sont plutôt bons comparativement à la distance euclidienne. Cependant, la partition binaire et le k-médoïdes donnent des prédictions de moindre qualité que k-voisin.

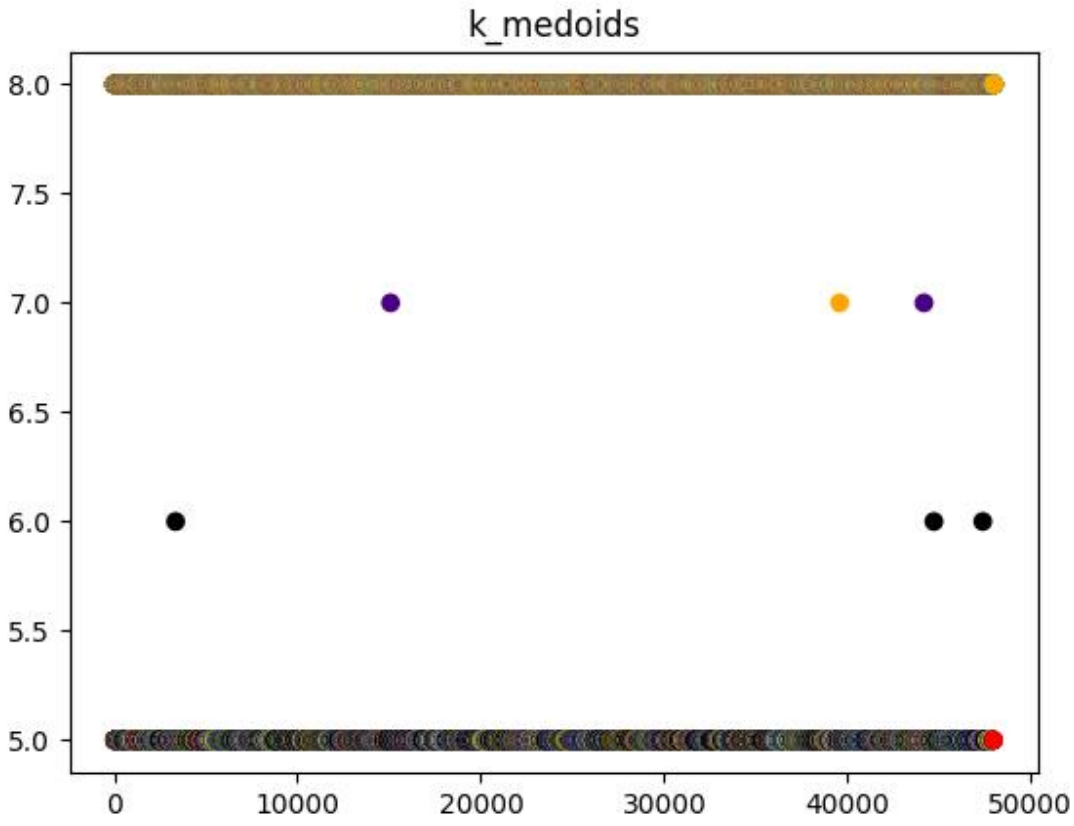
Notre dissimilarité utilisée avec k-médoïde a obtenu une v-mesure de 0.45 comparativement à environ 0.08 pour la distance euclidienne.

Visuellement, on remarque tout de suite que la distance euclidienne n'a pas du tout fonctionné.

Pour notre dissimilarité :

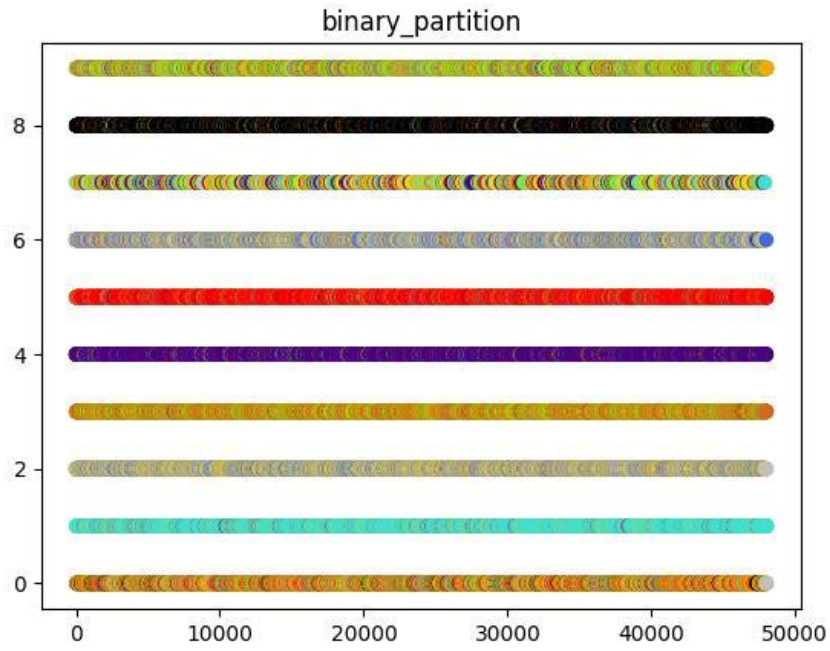


Comparativement à la distance euclidienne :

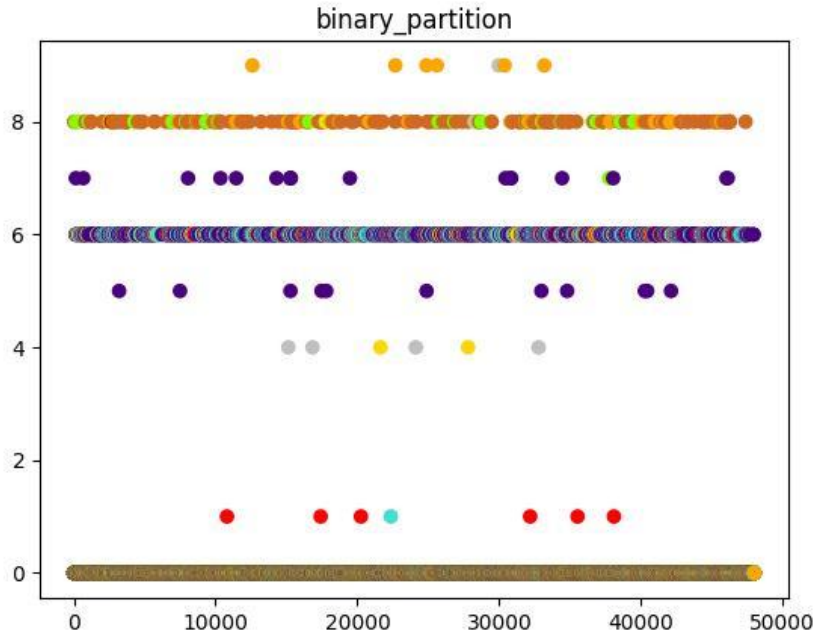


Pour la partition binaire, nous obtenons des comparaisons très similaires, mais des résultats légèrement meilleurs. Notre v-mesure bat encore une fois celle de la distance euclidienne, soit 0.49 versus une valeur près de 0.04.

Visuellement nous avons notre dissimilarité :



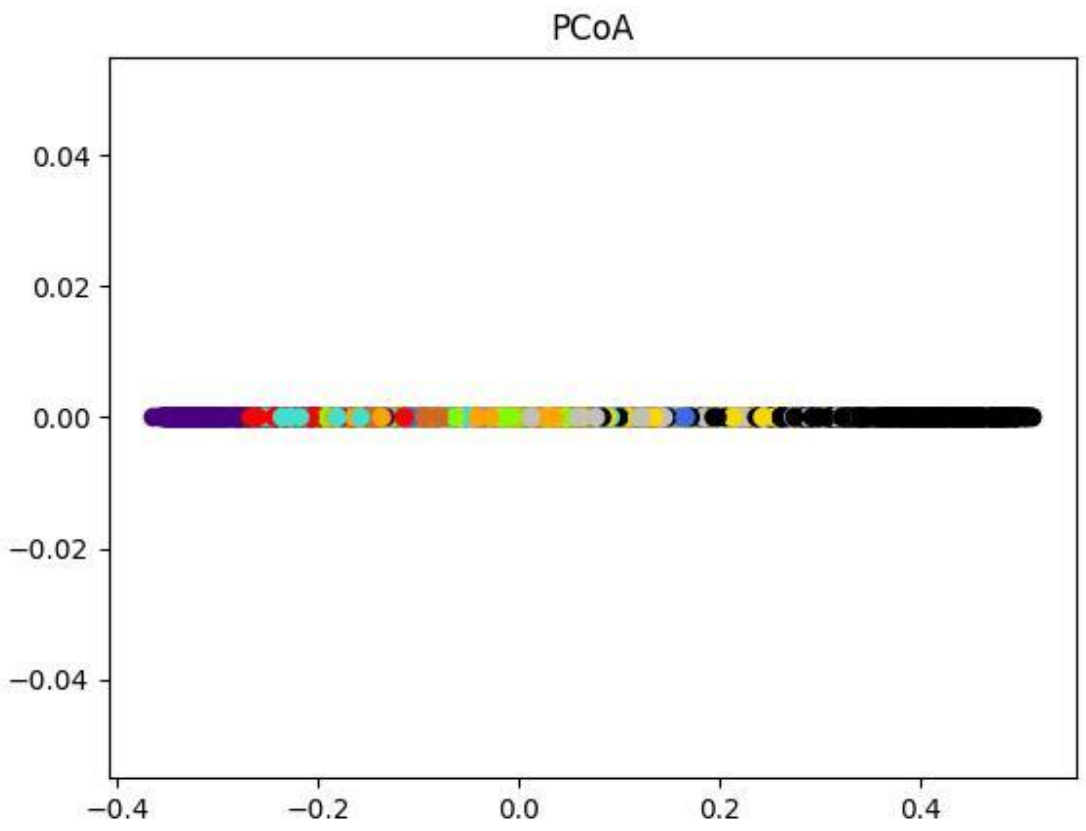
Contre la dissimilarité obtenue grâce à la distance euclidienne :



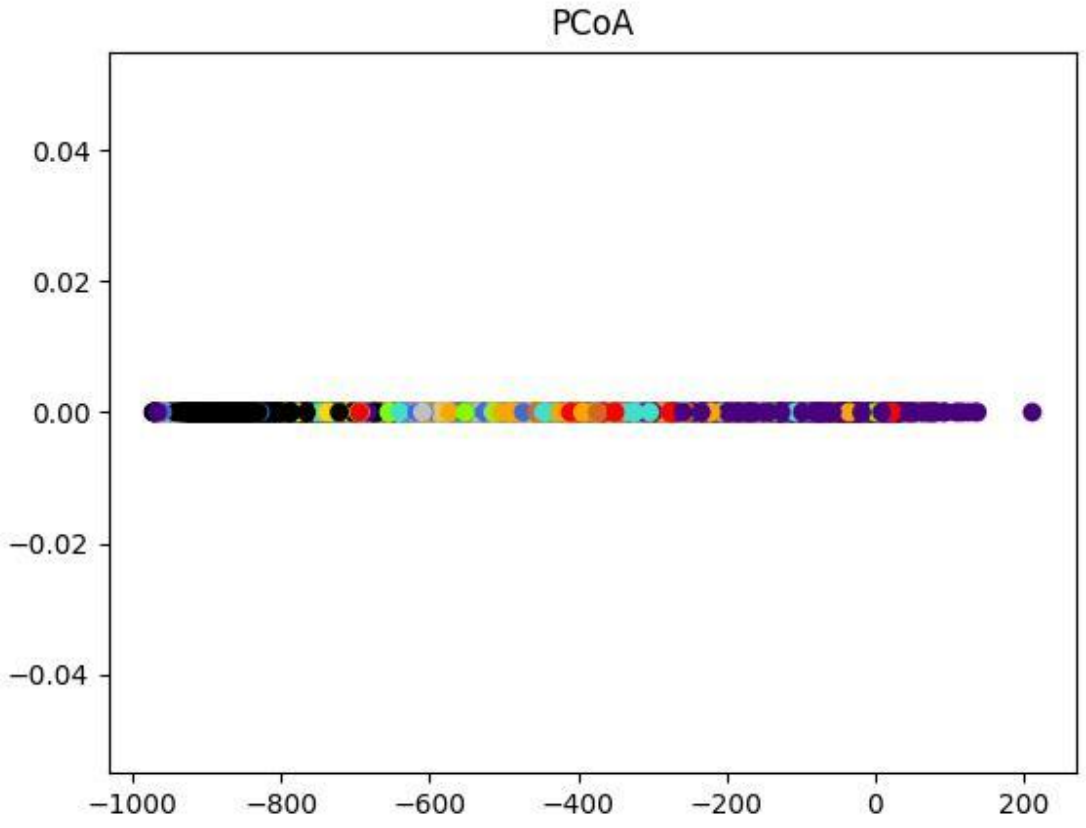
iii. Isomap et PCoA

Finalement, nous avons les algorithmes de réduction de dimensions. Plus utilisé pour visualiser les similarités et les dissimilarités, dans le cas de mnist, il est plus difficile de voir que notre dissimilarité semble obtenir de bons résultats.

Pour le positionnement multidimensionnel (PCoA), nous obtenons :

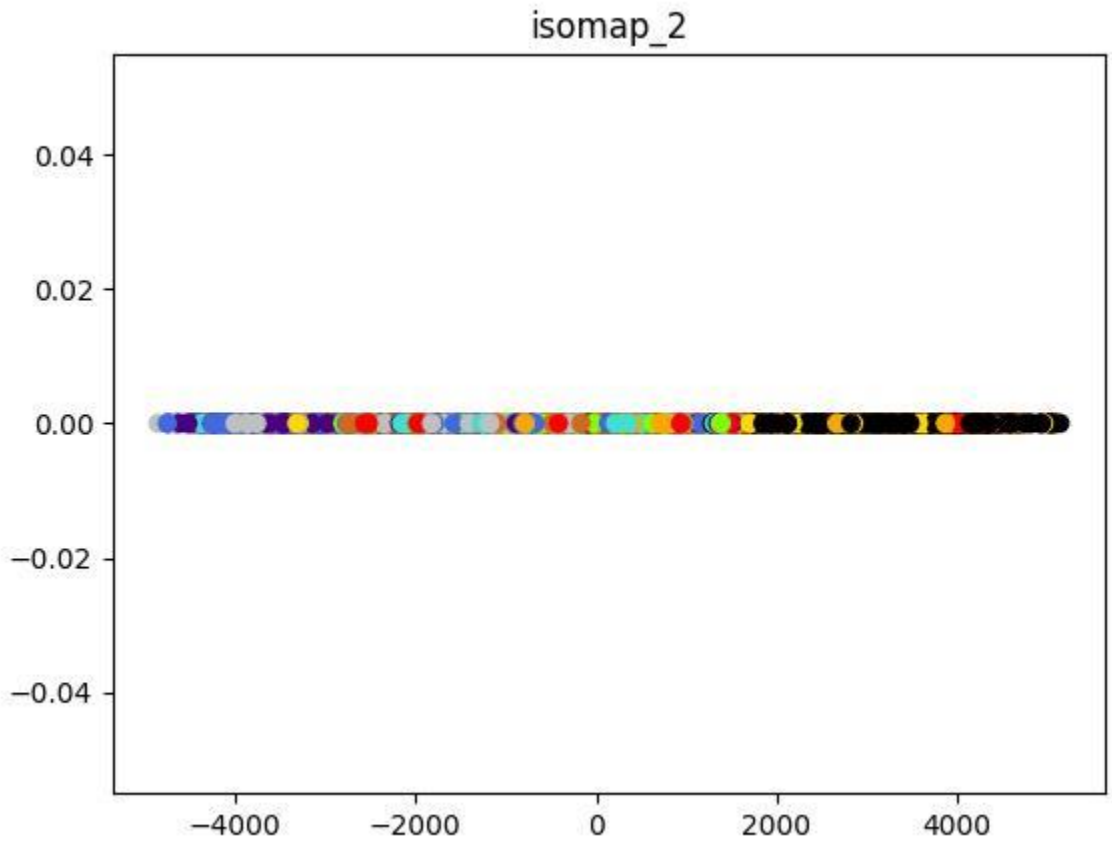


Contre la distance euclidienne :

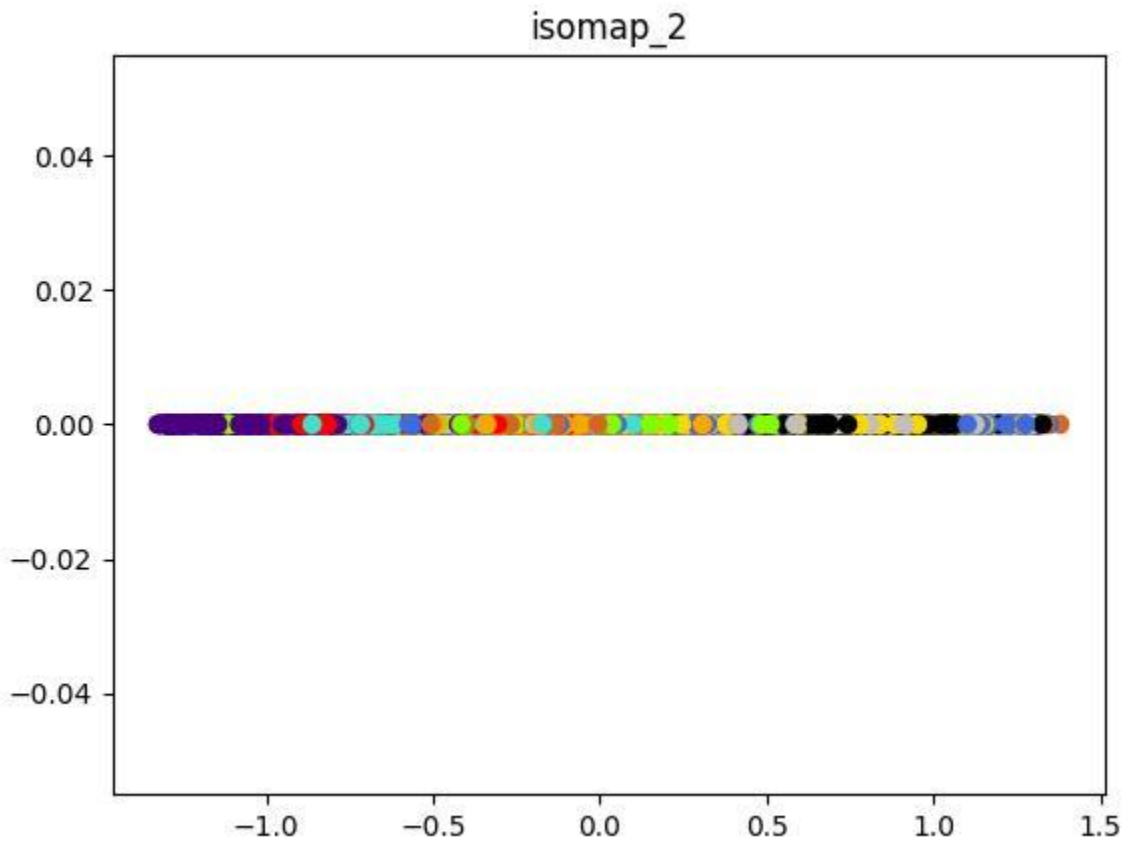


Isomap ne semble pas nous aider à distinguer les résultats non plus :

Nos résultats :



Contre la distance euclidienne



Pour comparer ces résultats, comme mentionnés plus tôt, nous les avons testés avec des k voisin. À ce moment, il semble que notre algorithme les ait mieux séparés. Si nous comparons les meilleurs v-mesure, nous obtenons 0,14 versus 0.10 pour PCoA et 0.11 versus 0.01 pour les isomap.

On peut donc conclure que la réduction de dimension n'est pas très utile dans notre analyse de comparaisons entre nos dissimilarités. La façon que nous avons présenté et calculé a été prise du cours. Probablement que d'autre façon aurait pu être utilisée pour mieux réduire la dimension et ainsi mieux comparer les données.

4.4. Discussion de l'approche

Notre approche semble avoir plutôt bien fonctionné. Lorsque nous regardons les résultats de l'annexe 6.2, nous avons remarqué que l'erreur est plutôt faible (environ 12% pour les k voisin).

Nous avons été étonnés que la distance euclidienne donne des résultats aussi faibles. Notre approche était bien meilleure. Pour avoir une meilleure image de la qualité du modèle, nous sommes allés vérifier d'autres sources. Nous avons fait un test de plus, nous avons essayé d'utiliser la librairie Sklearn sans imposer des similarités pré- calculer. Dans ce cas-ci notre modèle est inférieur. Sklearn obtient des résultats supérieurs à 0.9 pour les k-voisin par exemple.

Nous savons aussi que la base de données est assez populaire et après quelques recherches, il est facile de trouver des résultats mieux que les nôtres[2].

Ainsi bien que notre modèle ait d'assez bons résultats et qu'il soit mieux que la distance euclidienne, il n'est rien de révolutionnaire.

5. Conclusion

Dans cette analyse, nous avons présenté deux regroupements de dissimilarité et similarité, chacune associé à une base de données différente. Premièrement, nous avons présenter une dissimilarité qui utilise des égalités et des distances absolues avec les informations trouvées dans la base de données d'adultes. La deuxième dissimilarité étudiait les données de mnist et se faisait en deux étapes. Première étape était de calculer une distance euclidienne avec des moyennes. La deuxième était simplement de refaire une distance euclidienne en utilisant les premiers résultats obtenus.

[2] https://paperswithcode.com/sota/image-classification-on-mnist

De cette analyse nous pouvons conclure que la première idée fonctionnait moins bien. La v-mesure de toutes l'algorithme testé était plutôt basse. La similarité aurait besoin d'être retravailler.

Pour la deuxième similarité proposée, nous avons eu des problèmes avec certains algorithmes. Nous avons seulement testé certain cas avec les algorithmes de réduction de dimensionnalité et le résultat était plutôt faible. Il serait intéressant d'en plus amélioré l'algorithme utilisé avec les données d'adultes, de refaire une analyse plus poussée avec les algorithme de réduction de dimensionnalité pour mnist. Une idée possible serait d'essayer de réduire vers plus de dimension, afin de mieux séparer visuellement les données.

Finalement, nous avons aussi pu remarquer que dans les deux cas, les K voisins nous permettent d'obtenir de bons résultats. Malgré la simplicité de l'algorithme, une fois l'hyperparamètre trouver, il est possible de faire de bonnes prédictions par rapport aux autres algorithmes utilisés. Il serait intéressant de poursuivre l'analyse en utilisant d'autre algorithme de regroupement.

6. Annexe

6.1. Sortie pour adultes

neighbour_2 analyse :

v-mesure:0.0987973747543009

Total errors: 866 on 4000

>50K: 183 on 489

<=50K: 683 on 3511

neighbour_3 analyse :

v-mesure:0.13839457860947285

Total errors: 845 on 4000

>50K: 359 on 862

<=50K: 486 on 3138

neighbour_4 analyse :

v-mesure:0.1303952922641252

Total errors: 814 on 4000

>50K: 191 on 557

<=50K: 623 on 3443

neighbour_5 analyse :

v-mesure:0.16277328520633846

Total errors: 787 on 4000

>50K: 307 on 816

<=50K: 480 on 3184

neighbour_6 analyse :

v-mesure:0.1455506087067676

Total errors: 789 on 4000

>50K: 188 on 576

<=50K: 601 on 3424

neighbour_7 analyse :

v-mesure:0.16978712431962814

Total errors: 768 on 4000

>50K: 279 on 779

<=50K: 489 on 3221

k_medoids analyse :

v-mesure:0.12946153649550987

Total errors: 2828 on 4000

>50K: 2068 on 2297

<=50K: 760 on 1703

binary_partition analyse :

v-mesure:0.00014491851532472935

Total errors: 2881 on 4000

>50K: 2826 on 3760

<=50K: 55 on 240

PCoA_neighbour_2 analyse :

v-mesure:0.026849965017929283

Total errors: 803 on 3200

>50K: 178 on 323

<=50K: 625 on 2877

PCoA_neighbour_3 analyse :

v-mesure:0.026849965017929283

Total errors: 803 on 3200

>50K: 178 on 323

<=50K: 625 on 2877

PCoA_neighbour_4 analyse :

v-mesure:0.026849965017929283

Total errors: 803 on 3200

>50K: 178 on 323

<=50K: 625 on 2877

PCoA_neighbour_5 analyse :

v-mesure:0.026849965017929283

Total errors: 803 on 3200

>50K: 178 on 323

<=50K: 625 on 2877

isomap_neighbour_2 analyse :

v-mesure:0.022119295941346956

Total errors: 821 on 3200

>50K: 195 on 339

<=50K: 626 on 2861

isomap_neighbour_3 analyse :

v-mesure:0.022119295941346956

Total errors: 821 on 3200

>50K: 195 on 339

<=50K: 626 on 2861

isomap_neighbour_4 analyse :

v-mesure:0.022119295941346956

Total errors: 821 on 3200

>50K: 195 on 339

<=50K: 626 on 2861

isomap_neighbour_5 analyse :

v-mesure:0.022119295941346956

Total errors: 821 on 3200

>50K: 195 on 339

<=50K: 626 on 2861

6.2. Sortie pour mnist avec la similarité personnelle

neighbour_2 analyse :

v-mesure:0.7193145784220174

Total errors: 6989 on 48000

0: 779 on 5313

1: 461 on 5722

2: 892 on 5109

3: 1303 on 5424

4: 1098 on 5269

5: 841 on 4243

6: 267 on 4450

7: 523 on 5034

8: 437 on 3827

9: 388 on 3609

neighbour_3 analyse :

v-mesure:0.7401981177130792

Total errors: 6115 on 48000

0: 678 on 5174

1: 400 on 5643

2: 675 on 4826

3: 788 on 4749

4: 702 on 4734

5: 788 on 4385

6: 306 on 4637

7: 358 on 4904

8: 602 on 4333

9: 818 on 4615

neighbour_4 analyse :

v-mesure:0.7488960144362159

Total errors: 5888 on 48000

0: 583 on 5082

1: 370 on 5609

2: 608 on 4772

3: 786 on 4871

4: 747 on 4907

5: 794 on 4435

6: 317 on 4652

7: 436 on 5036

8: 589 on 4316

9: 658 on 4320

neighbour_5 analyse :

v-mesure:0.7524280351473589

Total errors: 5765 on 48000

0: 523 on 5014

1: 353 on 5590

2: 570 on 4714

3: 705 on 4725

4: 649 on 4739

5: 841 on 4518

6: 346 on 4714

7: 399 on 4990

8: 606 on 4385

9: 773 on 4611

neighbour_6 analyse :

v-mesure:0.7545067316808114

Total errors: 5696 on 48000

0: 512 on 5005

1: 344 on 5576

2: 580 on 4717

3: 741 on 4821

4: 713 on 4878

5: 771 on 4471

6: 342 on 4712

7: 413 on 5010

8: 612 on 4379

9: 668 on 4431

neighbour_7 analyse :

v-mesure:0.7543086821749205

Total errors: 5701 on 48000

0: 505 on 4985

1: 337 on 5560

2: 542 on 4665

3: 680 on 4702

4: 621 on 4723

5: 837 on 4544

6: 363 on 4750

7: 382 on 4968

8: 653 on 4457

9: 781 on 4646

k_medoids analyse :

v-mesure:0.44903936159840513

Total errors: 39416 on 48000

0: 2838 on 2838

1: 4463 on 4463

2: 4410 on 5313

3: 2832 on 3015

4: 3992 on 5312

5: 4728 on 5805

6: 1730 on 4970

7: 7531 on 7533

8: 3097 on 3099

9: 3795 on 5652

binary_partition analyse :

v-mesure:0.4941774787608751

Total errors: 44127 on 48000

0: 4476 on 4496

1: 3921 on 3923

2: 7008 on 7071

3: 2798 on 6025

4: 4267 on 4272

5: 4114 on 4167

6: 5196 on 5233

7: 1535 on 1558

8: 5509 on 5690

9: 5303 on 5565

PCoA_neighbour_2 analyse :

v-mesure:0.13542796533947254

Total errors: 28580 on 38400

0: 2874 on 5519

1: 2837 on 6217

2: 5115 on 5940

3: 4976 on 5724

4: 4142 on 4866

5: 2750 on 3179

6: 1883 on 2175

7: 2221 on 2687

8: 1279 on 1469

9: 503 on 624

PCoA_neighbour_3 analyse :

v-mesure:0.13542796533947254

Total errors: 28580 on 38400

0: 2874 on 5519

1: 2837 on 6217

2: 5115 on 5940

3: 4976 on 5724

4: 4142 on 4866

5: 2750 on 3179

6: 1883 on 2175

7: 2221 on 2687

8: 1279 on 1469

9: 503 on 624

PCoA_neighbour_4 analyse :

v-mesure:0.13542796533947254

Total errors: 28580 on 38400

0: 2874 on 5519

1: 2837 on 6217

2: 5115 on 5940

3: 4976 on 5724

4: 4142 on 4866

5: 2750 on 3179

6: 1883 on 2175

7: 2221 on 2687

8: 1279 on 1469

9: 503 on 624

PCoA_neighbour_5 analyse :

v-mesure:0.13542796533947254

Total errors: 28580 on 38400

0: 2874 on 5519

1: 2837 on 6217

2: 5115 on 5940

3: 4976 on 5724

4: 4142 on 4866

5: 2750 on 3179

6: 1883 on 2175

7: 2221 on 2687

8: 1279 on 1469

9: 503 on 624

isomap_neighbour_2 analyse :

v-mesure:0.11567517872946702

Total errors: 29235 on 38400

0: 3543 on 5829

1: 4552 on 7356

2: 4728 on 5490

3: 4766 on 5698

4: 3693 on 4297

5: 2729 on 3231

6: 1541 on 1907

7: 2133 on 2720

8: 1023 on 1221

9: 527 on 651

isomap_neighbour_3 analyse :

v-mesure:0.11567517872946702

Total errors: 29235 on 38400

0: 3543 on 5829

1: 4552 on 7356

2: 4728 on 5490

3: 4766 on 5698

4: 3693 on 4297

5: 2729 on 3231

6: 1541 on 1907

7: 2133 on 2720

8: 1023 on 1221

9: 527 on 651

isomap_neighbour_4 analyse :

v-mesure:0.11567517872946702

Total errors: 29235 on 38400

0: 3543 on 5829

1: 4552 on 7356

2: 4728 on 5490

3: 4766 on 5698

4: 3693 on 4297

5: 2729 on 3231

6: 1541 on 1907

7: 2133 on 2720

8: 1023 on 1221

9: 527 on 651

isomap_neighbour_5 analyse :

v-mesure:0.11567517872946702

Total errors: 29235 on 38400

0: 3543 on 5829

1: 4552 on 7356

2: 4728 on 5490

3: 4766 on 5698

4: 3693 on 4297

5: 2729 on 3231

6: 1541 on 1907

7: 2133 on 2720

8: 1023 on 1221

9: 527 on 651