

TRAVAIL D'ÉTUDE ET DE RECHERCHE - TER

Méthodes de réduction dimensionnelle et d'apprentissage actif

Travail présenté à Raphaël Butez
Maître de conférences en MATHÉMATIQUES APPLIQUÉES ET
APPLICATIONS DES MATHÉMATIQUES
(Université de Lille)

Réalisé par
Oscar Bidou - L3 MIAHS

Table des matières

1	Introduction	3
1.1	Classification supervisée principes fondamentaux	3
1.1.1	En pratique	3
1.2	Données	4
2	Préparation des données	5
2.1	Étape primaire	5
2.2	Analyse des corrélations	5
3	K plus proches voisins	6
3.1	Définition	6
3.2	Subtilités	6
4	Évaluation de classeur	6
4.1	Calcul du score	6
4.2	Validation et validation croisée	7
5	Réduction dimensionnelle	7
5.1	ACP - Analyse de composantes principales	8
5.1.1	ACP linéaire	8
5.1.2	Non-linéaire	9
5.2	LDA - Analyse discriminante linéaire	10
5.3	NCA - Analyse de composante de voisinage	10
5.4	NMF - Factorisation matricielle non négative	11
5.5	TSNE - intégration de voisin stochastique distribué en t	12
5.5.1	Étape 1	12
5.5.2	Étape 2	13
5.5.3	Étape 3	13
5.6	MDS - scaling multi-dimensionnel	13
5.7	Isomap - Isomap	14
5.8	LLE - encastrément localement linéaire	15
5.8.1	Étape 1	15
5.8.2	Étape 2	15
5.9	Spectral - clustering spectral	16
5.9.1	Étape 1	16
5.9.2	Étape 2	16

5.10	Résultat	17
6	Solidité des classeurs	17
6.1	Classique	17
6.2	apprentissage actif	20
7	Conclusion	21
8	Remerciements	22

Table des figures

1	Histogramme des notes	4
2	Heatmap des corrélations	5
3	Résultat PCA linéaire	9
4	Résultat PCA cosine	9
5	Résultat LDA	10
6	Résultat NCA	11
7	Résultat NMF	12
8	Résultat TSNE	13
10	Résultat ISOMAP	15
11	Résultat LLE	16
12	Résultat Clustering Spectral	17
13	Solidité du modèle sur \mathcal{X}_{test}	18
14	Solidité du modèle sur \mathcal{X}_{train}	19
15	Active learning	21

Résumé

Le but de ce TER est de nous familiariser avec quelques notions mathématiques de classifications supervisées ainsi que leur usage pratique sur un jeu de donnée test. Nous avons approfondis les différents algorithmes de réduction dimensionnelle ainsi que l'apprentissage actif.

1 Introduction

1.1 Classification supervisée principes fondamentaux

Le but de la classification est de prédire correctement la classe d'un individu en fonction de sa liste d'attributs. Soit \mathcal{D} l'ensemble des données. Une donnée $d \in \mathcal{D}$ est une liste d'attribut t.q :

$d = (a_1, a_2, \dots, a_p)$ avec $a_1, \dots, a_p \in A_1, \dots, A_p$ Nous avons à notre disposition $\mathcal{X} \subset \mathcal{D}$

La classification supervisée est un cas particulier de classification où, à chaque donnée d est associé une étiquette $y \in \mathcal{Y}$. Notre objectif est de trouver une fonction $f : \mathcal{D} \rightarrow \mathcal{Y}$ telle que $f(d) = y$ pour tous couples $(d; y) \in \mathcal{D} \times \mathcal{Y}$

1.1.1 En pratique

Autrement dit, afin d'apprendre à notre algorithme une classification correcte de nos données, nous avons à notre disposition un échantillon de données labellisées. Cet échantillon n'est qu'un sous-ensemble de la totalité des données (on ne possède pas toutes les données existante d'un problème). Afin de programmer les hyperparamètres du classifieur (paramètre définissant le classifieur), nous allons avoir besoin de quantifier le pourcentage d'erreurs. Cette évaluation ne peut se faire sur les données qui nous ont servi à établir le modèle. Le cas échéant, notre modèle va "overfit" nos données. L'"overfitting" correspond à une optimisation "sur mesure" du modèle des données initial. Pratiquement cela s'explique par un taux d'erreur de 0% sur les données initial et un taux d'erreur de 100% sur un nouvel individu. Afin d'éviter cela, nous divisons notre premier jeu de données en trois sous-ensembles. Le premier, l'ensemble d'entraînement (\mathcal{X}_{train}) va nous permettre d'établir le modèle. Deuxièmement, l'ensemble de validation (\mathcal{X}_{val}) va nous permettre de tester les hyperparamètres. Le troisième jeu, celui de test (\mathcal{X}_{test}) va nous servir pour tester la solidité du modèle après avoir fixée les hyperparamètres. Nous ne pourrions pas tester la solidité du modèle sur l'ensemble \mathcal{X}_{val} car les données auront déjà servi à sélectionner les hyperparamètres. Cependant, nous pourrions fusionner les jeux de données \mathcal{X}_{train} et \mathcal{X}_{val} pour entraîner notre modèle.

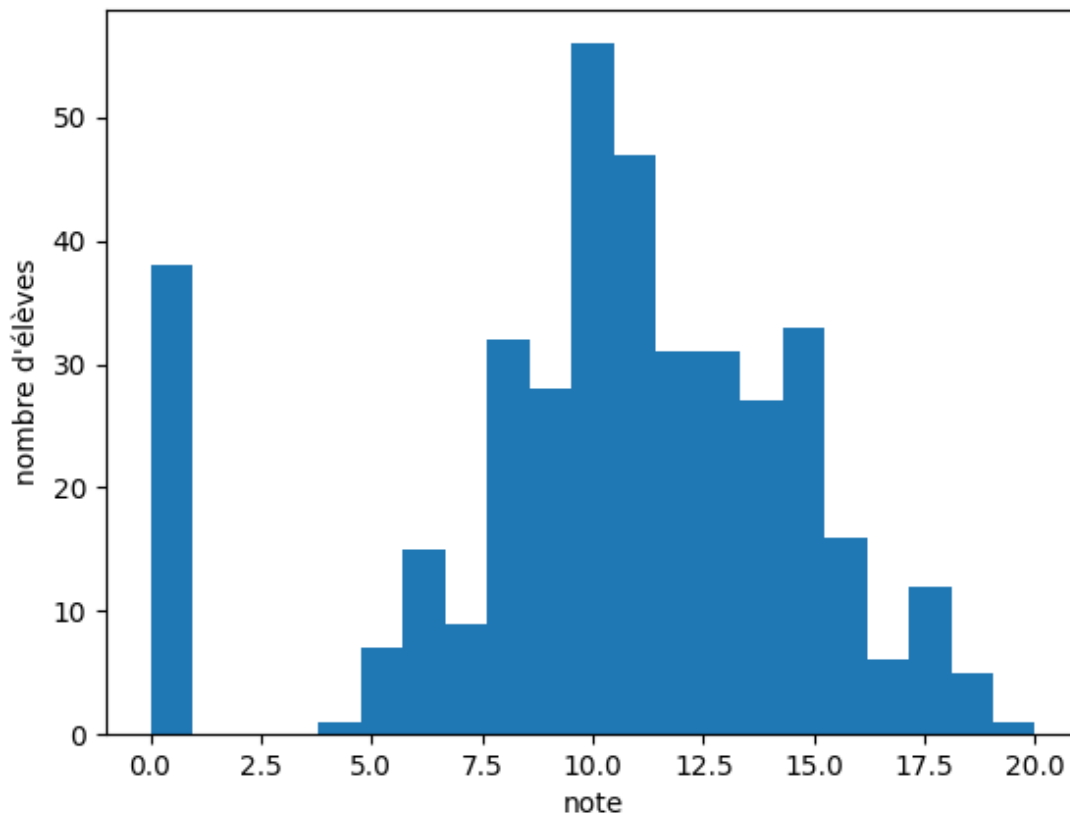


FIGURE 1 – Histogramme des notes

1.2 Données

Le jeu de données à notre disposition [Alcohol Effects On Study](#) a été récupéré sur kaggle, il met en parallèle les notes des 3 semestres de lycéens d'un lycée au Mexique et une trentaine de caractéristiques social (32). Nous retrouvons notamment, (l'âge, le sexe, la consommation d'alcool en semaine, le statut familiale,..). Le but est de classer correctement la note du troisième semestre. Si nous arrivons à avoir un score non négligeable, nous pourrions en déduire les caractéristiques sociales prédictives d'échec scolaire. Nous aurions pu supposer que les notes suivaient une loi normale, mais avoir regardé la répartition de chaque note, nous rejetons cette hypothèse avec un atome aussi important en 0.

Nous ne pouvons donc pas classer en deux catégories nos données, mais trois (0, au dessus de 0 et en dessous de 10, au dessus de 10). Le but est d'essayer de prédire la classe du troisième semestre en fonction des listes attributs. Dans la description des données, il est préconisé de ne pas prendre en considération les notes du premier et second semestre pour avoir une classification plus pertinente au prix de la performance.

2 Préparation des données

2.1 Étape primaire

La première étape est de nettoyer les données pour pouvoir effectuer un algorithme des K plus proches voisins. Après avoir enlevé les notes des deux premiers semestres, comme indiqué dans la description du jeu de données, la première chose à faire est de transformer les attributs catégoriels en vecteur numérique. Par exemple, si nous avons un attribut 'couleur' prenant les valeurs 'bleu', 'rouge' ou 'vert' nous allons créer 3 nouveaux attributs. L'attribut 'bleu', 'rouge' ainsi que l'attribut 'vert' tous les trois à valeurs dans $\{0,1\}$. Nous appellerons cette méthode 'OneHotEncodeur'. Lors d'un algorithme des K plus proches voisins, nous allons calculer la distance qui sépare un vecteur test à chaque vecteur de notre jeu de données. Nous allons donc normaliser chacun des attributs du jeu de données afin d'éviter les problèmes d'échelles et d'unités.

2.2 Analyse des corrélations

Pour qu'un algorithme de classification marche, il faut que les attributs soit indépendants entre eux. Pour cette raison, à chaque fois que nous avons appliqué un algorithme OneHotEncoder, nous avons enlevé un attribut. Pour reprendre l'exemple précédent, nous abandonnons la colonne 'vert'. Ainsi nous conservons toute l'information tout en perdant la dépendance des données.

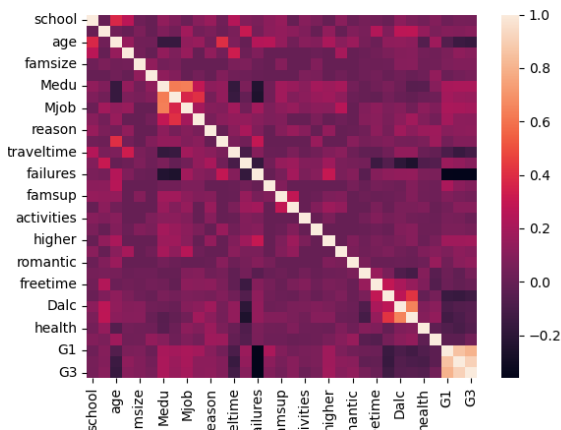


FIGURE 2 – Heatmap des corrélations

Pour vérifier l'indépendance de nos données, nous regardons la matrice de corrélation. Théoriquement, il y a indépendance seulement si la corrélation vaut zéro. Pratiquement, celle ci ne vaut jamais zéro. Nous considérons une corrélation entre deux attributs supérieur à 0.4 ou inférieure à -0.4 comme étant un signe d'une dépendance trop forte, dans ce cas là, nous retirons l'attribut ayant la corrélation la plus faible avec la liste Y . Ainsi, il ne reste plus que les données indépendants entre elles et expliquant une part de variabilité importante de la liste \mathcal{Y} . Dans

nos données, nous avons abandonné 5 attributs en plus des moyennes du premier et second semestre. Notamment :

- *Fedu* - niveau d'éducation du père
- *Mjob* - travail de la mère
- *guardian* - le gardien de l'enfant (père, mère, autre)
- *Fjob* - travail du père
- *Walc* - consommation d'alcool le week-end

3 K plus proches voisins

3.1 Définition

L'algorithme des K plus proches voisins (K_{nn} en anglais pour K -nearest-neighbors) est une méthode de classification supervisée. Elle labellise une nouvelle donnée par le label le plus présent parmi les K points les plus proches de celle ci. La manière la plus simple de trouver les K points les plus proches est de mesurer la distance Euclidienne entre tous les point de l'ensemble d'entraînement et le nouvel individu.

3.2 Subtilités

Un algorithme K_{nn} peut sembler simpliste, mais il cache de nombreux hyperparamètres. Le premier étant le nombre de voisins optimal pour un problème donnée. Nous pourrions aussi relever comme hyperparamètre le calcul de la distance entre le vecteur à classer et les individus de \mathcal{X}_{train} . Nous utiliserons la distance Euclidienne. Mais nous pouvons aussi mesurer la distance de la façons que l'on veut. Notamment en modifiant la pondération des différents attributs. Ainsi, certains éléments auraient plus de poids dans la prise de décision que d'autre. Nous pouvons penser à une pondération qui découle de la corrélation avec la liste \mathcal{Y} .

4 Évaluation de classeur

4.1 Calcul du score

Il existe plusieurs manières d'évaluer le score d'un classeur. Le plus courant étant de diviser le nombre de bonnes labellisations par le nombre total de labellisations. Cependant, nos données ne possède le même nombre d'individus dans les trois groupes. Alors 10% des individus ont 0, 23% d'entre eux ont entre 0 et 10 et 67% des étudiants valident le semestre avec plus de 10, nos données sont complètement déséquilibrées. Si nous appliquons la méthode d'évaluation ci-dessus, alors un classeur qui prédit systématiquement la classe 2 (plus de 10) aura un

score de 67% ! Ce qui n'est pas mauvais, mais nous nous attendons à un score de 30% pour un classer prédisant au hasard. Une solution serait de rééquilibrer le jeu de données en abandonnant des données de la classe 1 (entre 0 et 10) et de la classe 2 (plus de 10). Mais, ne voulant pas perdre en information, nous avons construit une nouvelle façon d'évaluer le classer. Nous moyennons le pourcentage de bonnes réponses pour chaque classe. Ainsi, un classer prédisant systématiquement la classe 2, aura un score de 30% ($\frac{0+0+100}{3}$). Initialement, le score de chaque classe a un poids égale. Nous nous laissons la possibilité de la modifier à loisir, celle ci n'étant pas un hyperparamètre.

Il pourra être pertinent de mesurer le score sur l'ensemble d'entraînement en plus de celui sur l'ensemble de test. Mais, nous pondérons le K plus proches voisins par la distance des données. L'ensemble d'entraînement aura donc systématiquement un score proche de 1. Pour cette raison, nous ne le représenterons pas.

4.2 Validation et validation croisée

Comme expliquer dans l'introduction, la validation simple consiste à diviser nos données en trois sous-ensemble. Si il existe beaucoup de manières différentes de diviser nos données, toutes gardent une composante aléatoire. Pour éviter des divisions trop aberrantes de nos données (tous les membres d'une catégorie sans dans un ensemble), nous effectuons une validation croisée. Pour ce faire, nous allons divisé nos données aléatoirement, puis nous stockons le score obtenus et réeffectuons l'opération avec une nouvelle division aléatoire. Après suffisamment d'itérations pour que nous puissions parler de convergence (ici une cinquantaine), nous moyennons les résultats. Cette opération peut devenir très vite très coûteuse en temps de calcul, mais elle assure des résultats consistants. Pour cette raison, tous les résultats présentés dans cet article ont été trouvé avec une validation croisée.

5 Réduction dimensionnelle

L'algorithme des K plus proches voisins nécessite un nombre exponentiel d'individus pour chaque attributs. Ici, nous avons 350 individus pour 27 attributs, ce n'est pas suffisant. Nous allons donc procéder à une réduction dimensionnelle afin de garder le plus de variance possible tout en diminuant le nombre d'attributs. Idéalement, nous conservons 50% de la variance avec deux attributs (ce qui permet une représentation graphique en deux dimensions). Ici, nous nous intéresserons à différents types de réduction dimensionnelle que nous avons pu trouver dans la littérature. Il est important de noter que chacune de ces méthodes possède de nombreux hyperparamètres, seul les plus intéressantes combinaisons seront traitées. Au

total, nous allons observer 10 différents algorithmes de réduction dimensionnelle.

- *PCA linéaire* - analyse de composantes principales avec un noyau linéaire
- *PCA cosine* - analyse de composantes principales avec un noyau cosinus
- *LDA* - analyse discriminante linéaire
- *NCA* - analyse de composante de voisinage
- *NMF* - factorisation matricielle non négative
- *TSNE* - intégration de voisin stochastique distribué en t
- *MDS* - scaling multi-dimensionnelle
- *Isomap* - Isomap
- *LLE* - encastrément localement linéaire
- *Spectral* - clustering spectral

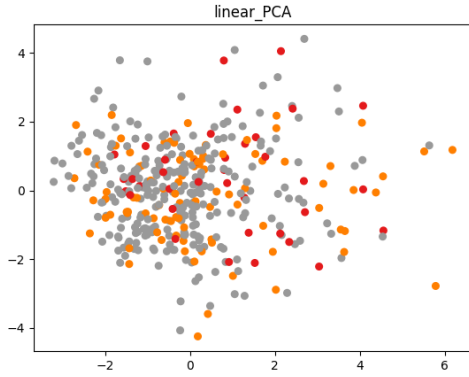
Seul, PCA linéaire ainsi que LDA, NCA, NMF et MDS sont des méthodes de projection linéaires. Effectuer une réduction de la dimensionnalité équivaut à effectuer la première partie d'une classification non supervisée. Nous divisons nos données en groupe avant de comparer les groupes aux labels.

5.1 ACP - Analyse de composantes principales

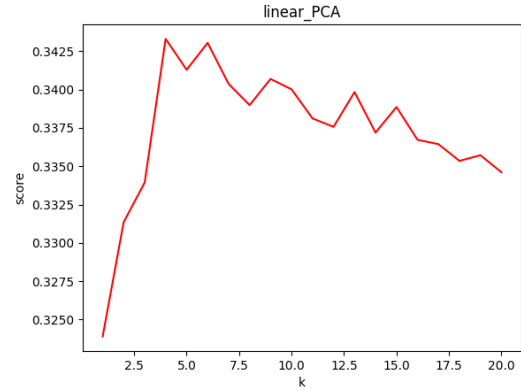
L'analyse de composantes principales est la méthode de réduction dimensionnelle la plus courante et la plus simple. Cette méthode permet une projection linéaire de nos données dans un nouvel espace. Cependant, nous pouvons la pimenter en rajoutant un noyau. En effet, cet ajout permet de projeter des données non-linéaire dans un espace d'une dimensions supérieur.. De cette façon, nos données deviennent linéairement séparable. Pour revenir à un espace en deux dimensions, nous sacrifions la dimension qui explique le moins de variance des deux obtenus avec une ACP, et nous la remplaçons avec un nouveau vecteur obtenus en appliquant une fonction prédéterminé à nos données.

5.1.1 ACP linéaire

Utiliser un noyau linéaire revient au même que de ne pas utiliser de noyau (à une différence de complexité près). Pour effectuer une ACP, il suffit de calculer les deux vecteurs propres associés aux deux valeurs propres les plus importantes. Nous multiplions ensuite les données par la matrice composée des deux vecteurs propres. La proportion de la variance expliquée (donc la qualité du modèle) est donnée par la valeur des deux meilleurs valeurs propres. Leurs sommes correspond à la proportion de variance expliquée par le modèle. Dans notre cas, les deux premières dimensions explique 17% de la variance, ce qui est très mauvais. Nous n'observons pas de division très claire entre les catégories des données. Ce qui se retranscrit



(a) Données projetées avec PCA



(b) score en fonction de k

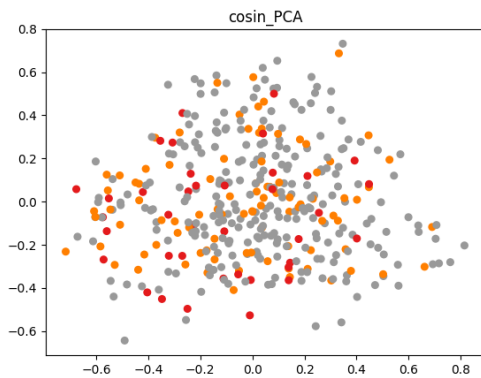
FIGURE 3 – Résultat PCA linéaire

dans un score très mauvais 30%(presque le hasard).

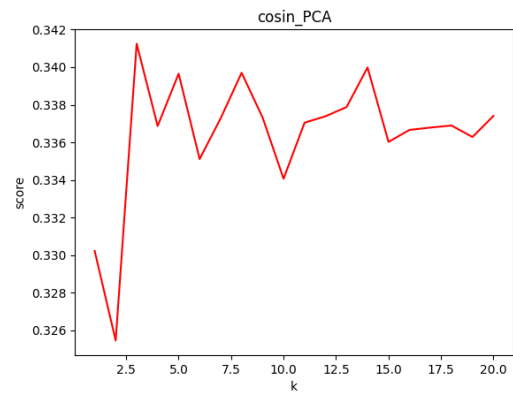
5.1.2 Non-linéaire

La nouvelle dimension a été générée en calculant la similarité cosinus. Soit k le noyau, x_1, x_2 nos données projetées dans un nouvel espace avec une PCA.

$$k(x_1, x_2) = \frac{x_1 x_2^T}{\|x_1\| \|x_2\|} \text{ la troisième dimension de notre sous-espace}$$



(a) Projection avec cosinus PCA



(b) Score en fonction de k

FIGURE 4 – Résultat PCA cosinus

Nous n'observons pas non plus de division très claire entre les catégories de données. Ce qui se traduit une nouvelle fois par un score de 30%.

5.2 LDA - Analyse discriminante linéaire

L'algorithme d'analyse discriminante linéaire nécessite les labels de nos données, ce qui en fait une méthode de classification supervisée. Là où la PCA va essayer de capturer le maximum de variance expliqué, l'ADL maximise la séparabilité de nos catégories connues. L'ADL calcule le centre de nos données avant de les projeter dans un espace en deux dimensions qui maximise la distance moyenne de chaque catégorie au centre calculé, tout en minimisant la variance inter-groupe. Cet algorithme nécessite quatre suppositions, indépendance des résultats, homoscedasticité, multicollinéarité et, enfin, normalité multivariée.

Nous pouvons aussi rajouter un noyau à l'ADL afin de projeter des données non-linéaires, ce cas n'a pas été traité.

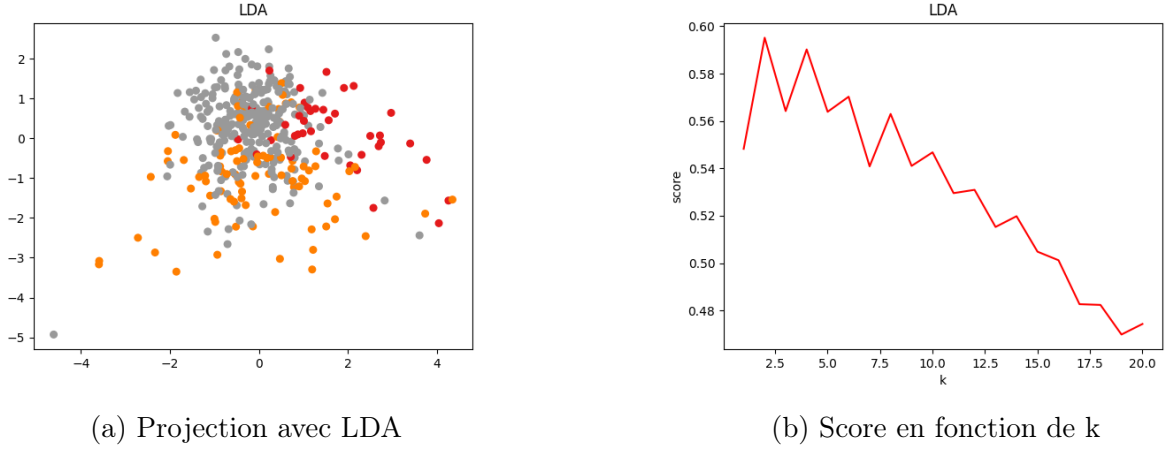


FIGURE 5 – Résultat LDA

Non seulement la séparation des groupes est visible, mais le score est jusqu'à deux fois supérieur que le hasard sur l'ensemble \mathcal{X}_{val} .

5.3 NCA - Analyse de composante de voisinage

Tout comme l'ADL, l'ACV est un algorithme supervisé. Cet algorithme va chercher un espace qui permet une transformation linéaire de nos données tel qu'un algorithme Leave One Out (LOO) du K_{nn} ($K = 1$) ait un score maximal. Nous allons essayer de maximiser la probabilité de labellisation correcte p_i .

$$\arg \max_L \sum_{i=0}^{N-1} p_i$$

Où N est le nombre d'individus, L la projection de nos données dans notre nouvel espace et

$$p_i = \sum_{j=0}^{N-1} p_{ij} \mathbb{1}_{y_i=y_j}$$

Avec y_i le label de x_i . Enfin, p_{ij} est défini ci dessous :

$$p_{ij} = \frac{e^{-\|Lx_i - Lx_j\|^2}}{\sum_{k \neq i} e^{-\|Lx_i - Lx_k\|^2}}$$

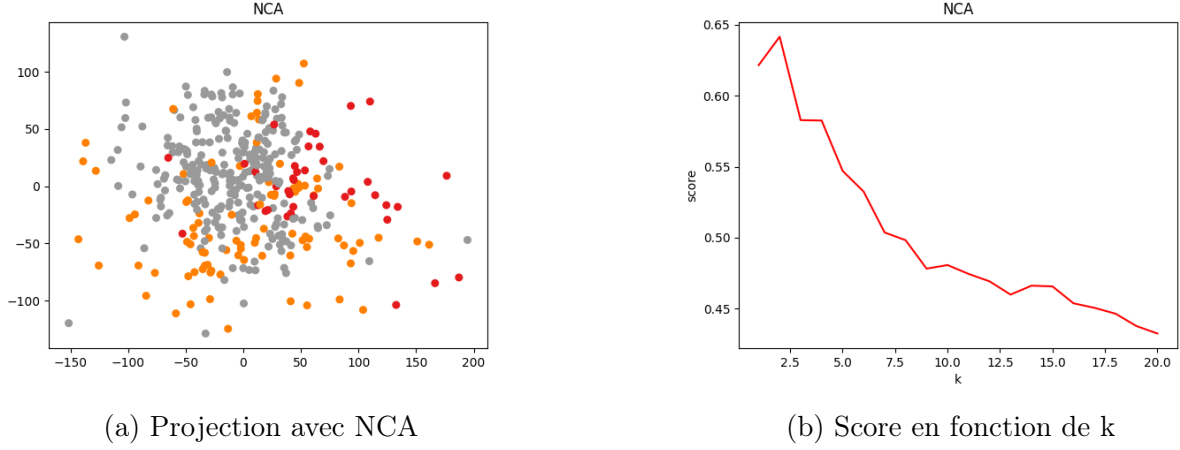
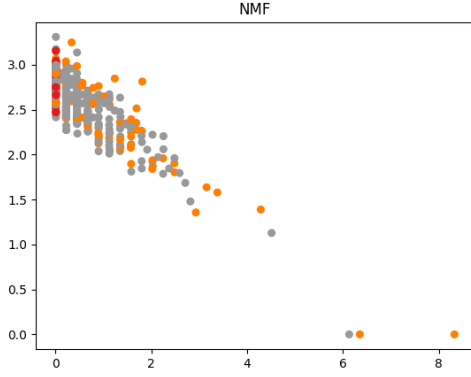


FIGURE 6 – Résultat NCA

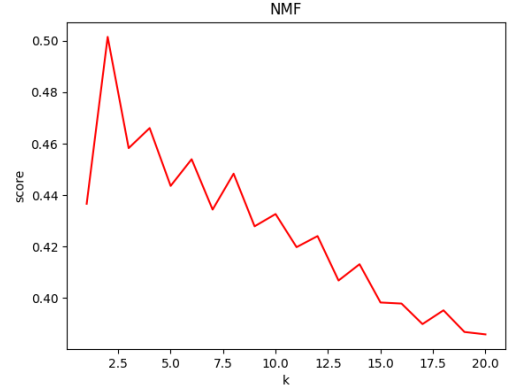
La séparation des classes semble bien meilleur qu'une PCA. Résultat qui se retranscrit dans le score, 65% ce qui est le meilleur jusqu'à présent.

5.4 NMF - Factorisation matricielle non négative

L'algorithme de factorisation matricielle non-négative permet une projection linéaire de nos données. Il transforme notre matrice de données \mathcal{V} en un produit de deux matrices, notre nouvel espace \mathcal{W} par une matrice de pondération \mathcal{H} . Nous trouvons ces matrices en minimisant la fonction d'erreur $\|\mathcal{V} - \mathcal{W}\mathcal{H}\|_F$. Il est nécessaire que les données soient strictement positives, nous ne normalisons donc pas nos données avec d'appliquer cet algorithme. Si la répartition des données est particulière, les résultats ne sont pas particulièrement bons, un score de 50% reste très médiocre.



(a) Projection avec NMF



(b) Score en fonction de k

FIGURE 7 – Résultat NMF

5.5 TSNE - intégration de voisin stochastique distribué en t

Cet algorithme est une méthode non linéaire qui consiste à créer une distribution de probabilités qui représente les similarités entre voisins dans un espace en grande dimension et dans un espace de deux dimensions. Il se découpe en trois étapes.

5.5.1 Étape 1

Dans un premier temps, nous centrons une gaussienne à chaque point et nous calculons la probabilité conditionnelle $p_{i|j}$ que x_i choisirait x_j comme voisin. Cette probabilité s'exprime de la forme :

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma_i^2)}$$

Nous mesurons la probabilité (aire sous la courbe) entre le point x_i et x_j , normalisée par la somme des probabilités. Ainsi, $\sum_{ij} p_{i|j} = 1$. De plus, $p_{i|i} = 0$.

Nous avons donc :

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2N}$$

Où N est le nombre de dimensions.

Un des hyperparamètres de cet algorithme est la perplexité $Perp$. σ_i est positivement corrélée à la perplexité :

$$Perp(P_i) = 2^{-\sum p_{i|j} \log_2 p_{j|i}}$$

5.5.2 Étape 2

Secondement, nous allons projeter nos données dans un sous-espace bi-dimensionnelle, avec une ACP. Similairement à la première étape, nous mesurons une nouvelle distribution de probabilité $q_{i|j}$. Mais, cette fois ci, nous utiliserons une distribution de Student à un degré de liberté.

$$q_{i|j} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}}$$

5.5.3 Étape 3

Pour améliorer la position de nos données dans notre espace de dimension, nous allons essayer de faire coïncider nos deux mesure de similarité. Le but est donc de minimiser la divergence de Kullback-Leibler avec une descente de gradient.

$$\text{KL}(P \parallel Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

Nous obtenons donc les résultat ci-dessous.

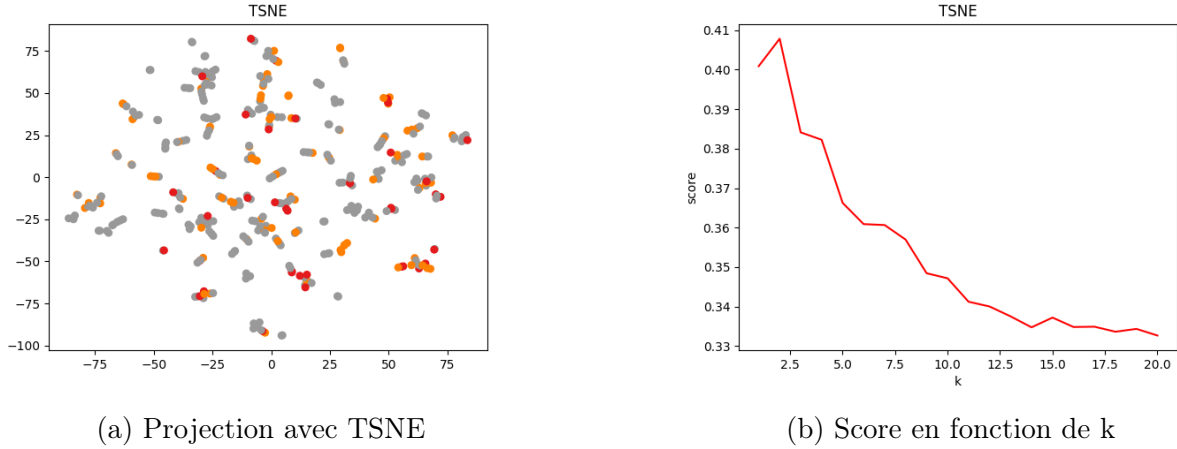


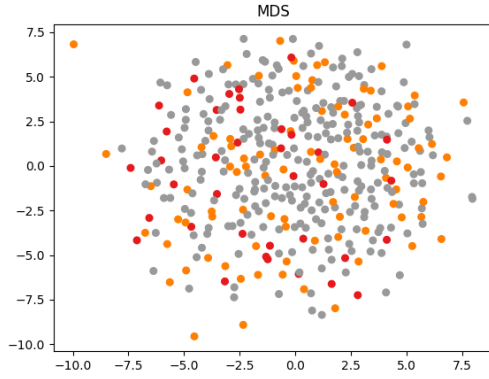
FIGURE 8 – Résultat TSNE

La représentation en deux dimensions est très particulière, mais caractéristique de cette approche. Cependant, les résultats ne sont pas au rendez vous avec un score de 40%.

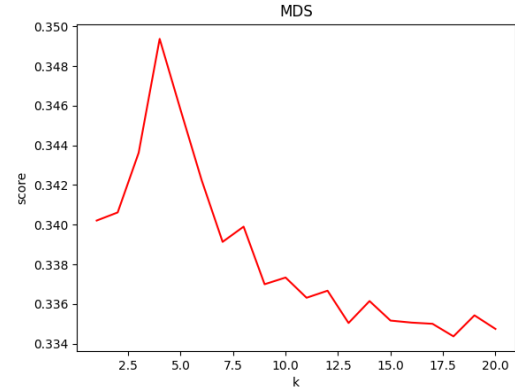
5.6 MDS - scaling multi-dimensionnel

Le positionnement multi-dimensionnelles consiste à trouver les N vecteurs $\mathbf{y}_1, \dots, \mathbf{y}_N$ de taille m (ici $m = 2$) qui minimisent le stress/fonction de coût $S(\mathbf{y}_1, \dots, \mathbf{y}_N)$.

Nous avons décidé d'utiliser une fonction de stress métrique.



(a) Projection avec MDS



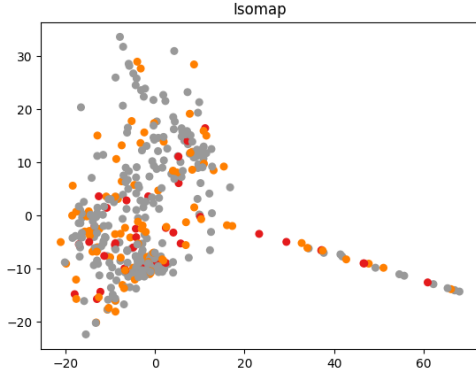
(b) Score en fonction de k

$$S(\mathbf{y}_1, \dots, \mathbf{y}_N) = \sum_{i \neq j} (d_{ij} - \|\mathbf{y}_i - \mathbf{y}_j\|)^2$$

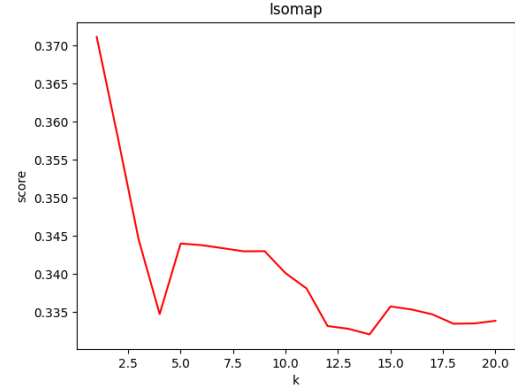
Le partitionnement en deux dimensions est particulièrement mauvais, le score ne dépasse pas le hasard.

5.7 Isomap - Isomap

L'algorithme Isomap est une projection non linéaire du positionnement multidimensionnelles (MDS). Pour une projection en deux dimensions, les deux nouveaux vecteurs sont définis comme étant les meilleurs vecteurs de la matrice de distance géodésique. Cette matrice est la distance entre chaque individus calculée par un algorithme de Dijkstra (somme des arêtes multipliées par leur poids). Il existe plusieurs façons de construire la matrice de voisinage sur laquelle nous allons appliquer l'algorithme de Dijkstra. La plus courante est de trouver les K plus proches voisins de chaque points. Ils sont donc reliés uniquement à leurs voisins avec un poids égaux à leur distance. La projection des données est caractéristique des projection non-linéaire. Cependant, une nouvelle fois, le score est très mauvais 40%.



(a) Projection avec Isomap



(b) Score en fonction de k

FIGURE 10 – Résultat ISOMAP

5.8 LLE - encastrément localement linéaire

L'ELL est méthode de réduction non linéaire. Très similaire d'une Isomap, il existe beaucoup de variantes, kernel LLE, inverse LLE ou même des fusions de LLE (ISOLLE, ...). Nous traiterons seulement du cas le plus simple, LLE classique.

5.8.1 Étape 1

Nous calculons la matrice de voisinage avec un algorithme des K plus proche voisins. Ensuite, nous calculons la matrice poids W de chaque arêtes en minimisant la fonction E .

$$E(W) = \sum_i \left| \mathbf{x}_i - \sum_j \mathbf{W}_{ij} \mathbf{x}_j \right|^2$$

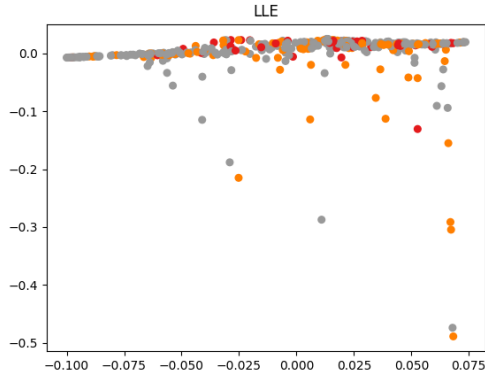
Avec pour contrainte $\sum_j \mathbf{W}_{ij} = 1$

5.8.2 Étape 2

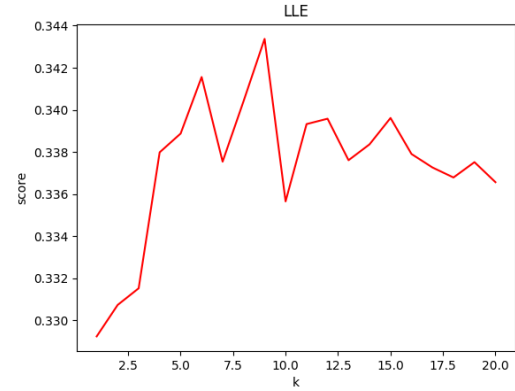
Les poids étant trouvés, nous allons pouvoir optimiser la positions de nos données dans notre nouvel espace en 2 dimensions crée par les deux meilleurs vecteurs propres. La projection de celles ci dans notre nouvel espace sont appelées $\mathbf{y}_1, \dots, \mathbf{y}_N$. Nous allons minimiser la fonction de coût C .

$$C(Y) = \sum_i \left| \mathbf{y}_i - \sum_j \mathbf{W}_{ij} \mathbf{y}_j \right|^2$$

Nous trouvons donc la répartition ci-dessous La répartition est très mauvaise avec un score à peine supérieure a 30%.



(a) Projection avec LLE



(b) Score en fonction de k

FIGURE 11 – Résultat LLE

5.9 Spectral - clustering spectral

Le clustering Spectral est un grand classique de la classification non supervisée, cet algorithme est lui aussi une projection non linéaire des données.

5.9.1 Étape 1

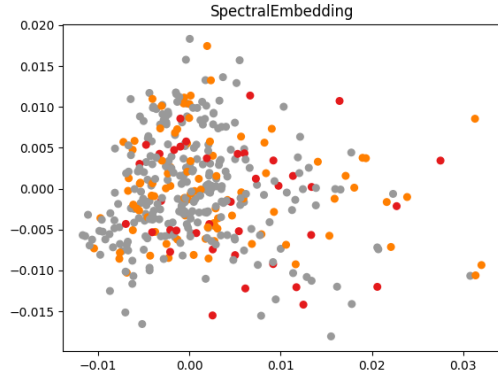
Nous calculons le Laplacien du graphe.

$$L = D - A$$

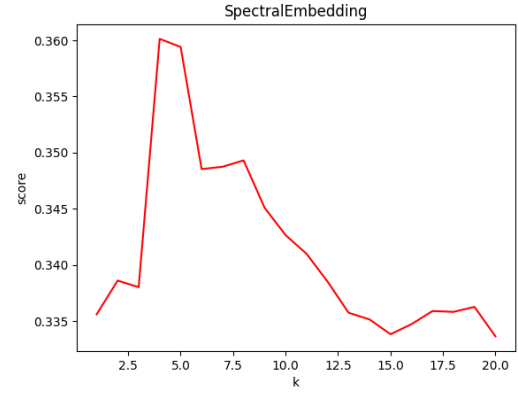
avec A la matrice de voisinage et D la matrice des degrés, c.a.d $D_{ii} = \sum_j A_{ij}$

5.9.2 Étape 2

La suite est trivial, les deux meilleurs vecteurs propres du Laplacien sont les nouvelles coordonnées de nos données dans un nouvel espace en deux dimensions.



(a) Projection avec Spectral Embedding



(b) Score en fonction de k

FIGURE 12 – Résultat Clustering Spectral

5.10 Résultat

Comme nous avons pu le voir, pour ce jeu de données, toutes les méthodes de réductions dimensionnelles ne se valent pas. En effet, les scores varient entre 64% et 34%. Clairement, deux classeurs sont plus performants que les autres, LDA ainsi que NCA.

	Division Linéaire				
	lin_PCA	LDA	NCA	NMF	MDS
best K	4	2	2	2	4
score	0.34	0.6	0.64	0.5	0.35

	Division non-Linéaire				
	cos_PCA	TSNE	Isomap	LLE	Spectral
best K	3	1	1	9	4
score	0.34	0.41	0.37	0.34	0.36

6 Solidité des classeurs

6.1 Classique

Maintenant que les hyperparamètres ont été fixés, nous devons désormais tester la qualité des classeurs. Pour cela, nous regardons leur solidité. La solidité est la dépendance du score à la taille du nouvel échantillon d'entraînement \mathcal{X}_{train}^* t.q

$$\mathcal{X}_{train}^* = \mathcal{X}_{train} \cup \mathcal{X}_{val}$$

Et, évidemment

$$\mathcal{Y}_{train}^* = \mathcal{Y}_{train} \cup \mathcal{Y}_{val}$$

Fonctionnellement, nous entraînons le classeur sur les 10 premières données de l'espace d'entraînement \mathcal{X}_{train}^* avant de calculer le score sur la totalité de l'espace de test \mathcal{X}_{test} . Nous recommençons mais cette fois ci avec les 20 premiers individus. Ainsi de suite jusqu'à ce que nous la totalité de l'ensemble d'entraînement soit utilisé. Dans un premier temps, la sélection des nouveaux individus se fais au hasard. Si nous regardons les score pour chacune des classes,

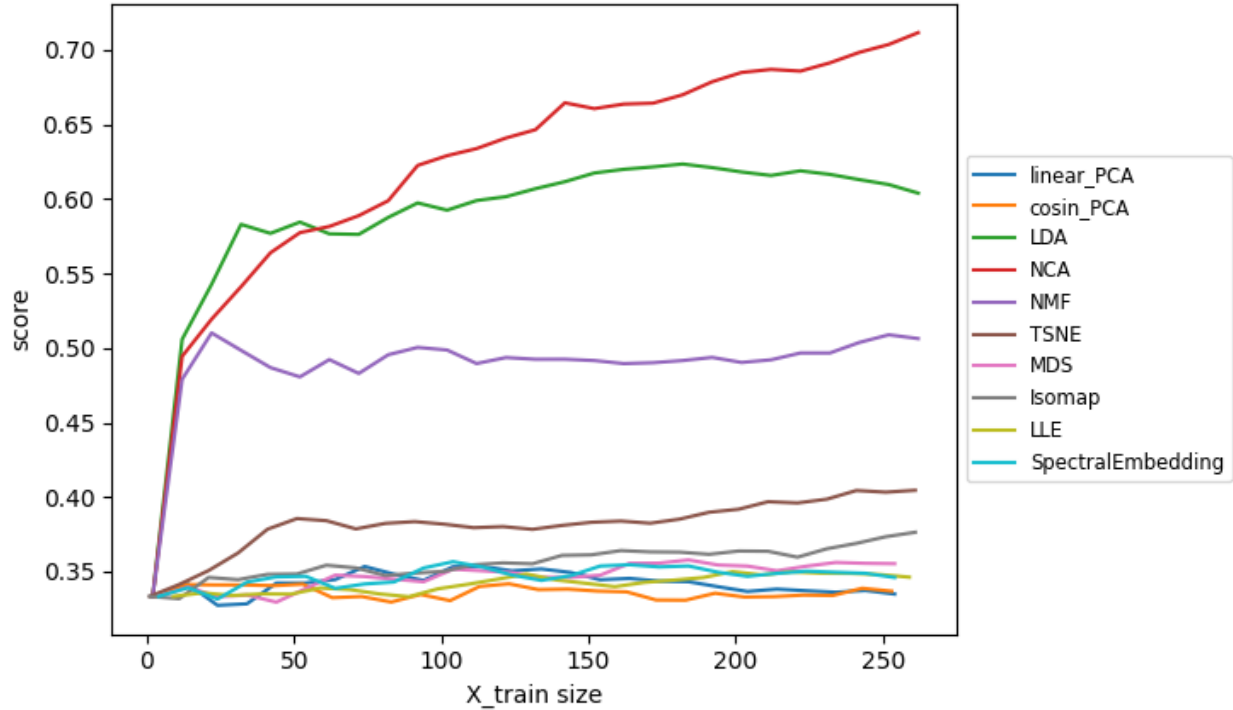


FIGURE 13 – Solidité du modèle sur \mathcal{X}_{test}

nous remarquons que tous les classeurs ont beaucoup de mal à labelliser correctement la première classe (note du troisième semestre égale à 0). À l'exception de deux classeurs, LDA et NCA. LDA, comme NCA, a un score équivalent dans toutes les classes ce qui fait preuve d'une réduction dimensionnelle adaptée à nos données. Il n'est cependant pas très surprenant que l'ACV soit l'algorithme le plus performant, en effet la réduction de la dimensionnalité est effectuée afin de maximiser le score d'un algorithme des K plus proches voisins. L'algorithme MNF lui aussi classe avec un score égale tous les labels. Cependant, il reste plutôt bas, 50%, en comparaison à NCA ou LDA. Nous relevons aussi que les transpositions linéaire sont en moyenne beaucoup plus performant que les transposition non-linéaire qui peine à dépasser un score de 30%.

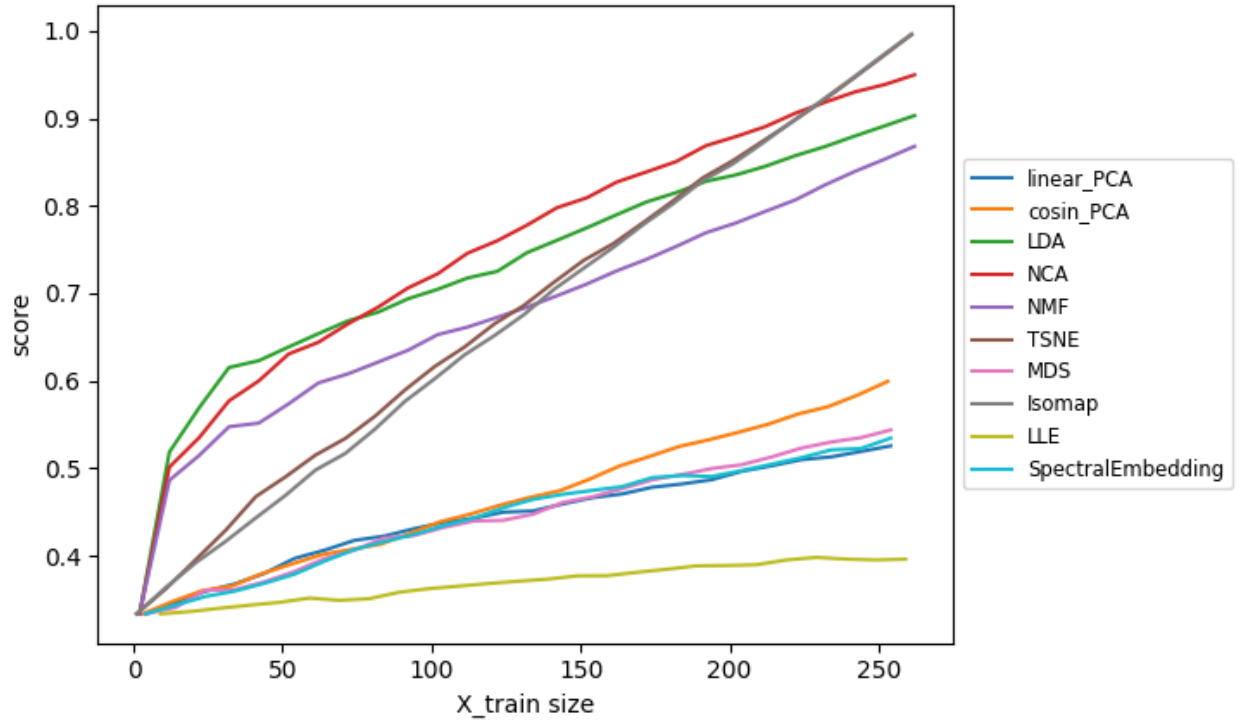


FIGURE 14 – Solidité du modèle sur \mathcal{X}_{train}

ensemble		classeur	x=0	0<x<10	10<=x<=20	mean
Test	linear	linear_PCA	0.01	0.26	0.73	0.33
		LDA	0.61	0.58	0.62	0.60
		NCA	0.68	0.68	0.77	0.71
		NMF	0.52	0.52	0.47	0.50
		MDS	0.10	0.23	0.73	0.35
	non-linear	cosine_PCA	0.01	0.11	0.86	0.32
		TSNE	0.30	0.43	0.51	0.41
		Isomap	0.15	0.25	0.73	0.37
		LLE	0.03	0.11	0.91	0.35
		Spectral	0.03	0.30	0.71	0.34
Train	linear	linear_PCA	0.19	0.58	0.83	0.53
		LDA	1.00	0.95	0.77	0.90
		NCA	1.00	0.94	0.91	0.95
		NMF	1.00	0.92	0.69	0.87
		MDS	0.25	0.57	0.83	0.55
	non-linear	cosine_PCA	0.04	0.26	0.92	0.40
		TSNE	1.00	0.88	0.71	0.86
		Isomap	1.00	1.00	1.00	1.00
		LLE	0.05	0.19	0.94	0.39
		Spectral	0.24	0.57	0.82	0.54

6.2 apprentissage actif

L'apprentissage actif est une méthode de classification dite semi-supervisée. L'usage de l'apprentissage actif est plus souvent utilisée afin de diminuer le coût de labellisation des données, ici nous allons essayer de maximiser le score en fonction du nombre d'individus à notre disposition.

Autrement dis, durant la validation du classeur, nous ne sélectionnons pas nos nouvelles données au hasard. À chaque itération, nous allons chercher 10 nouveaux individus de notre ensemble \mathcal{X}_{train}^* du label ayant le score le plus faible à l'itération précédente. Si tous les individus du label ont déjà été utilisés, nous sélectionnons 10 individus de la seconde classe ayant eu le plus faible score. Ainsi, nous devrions observer une augmentation du score encore plus forte que celle déjà observé avant un rétablissement pour les classeur LDA, NCA et NMF.

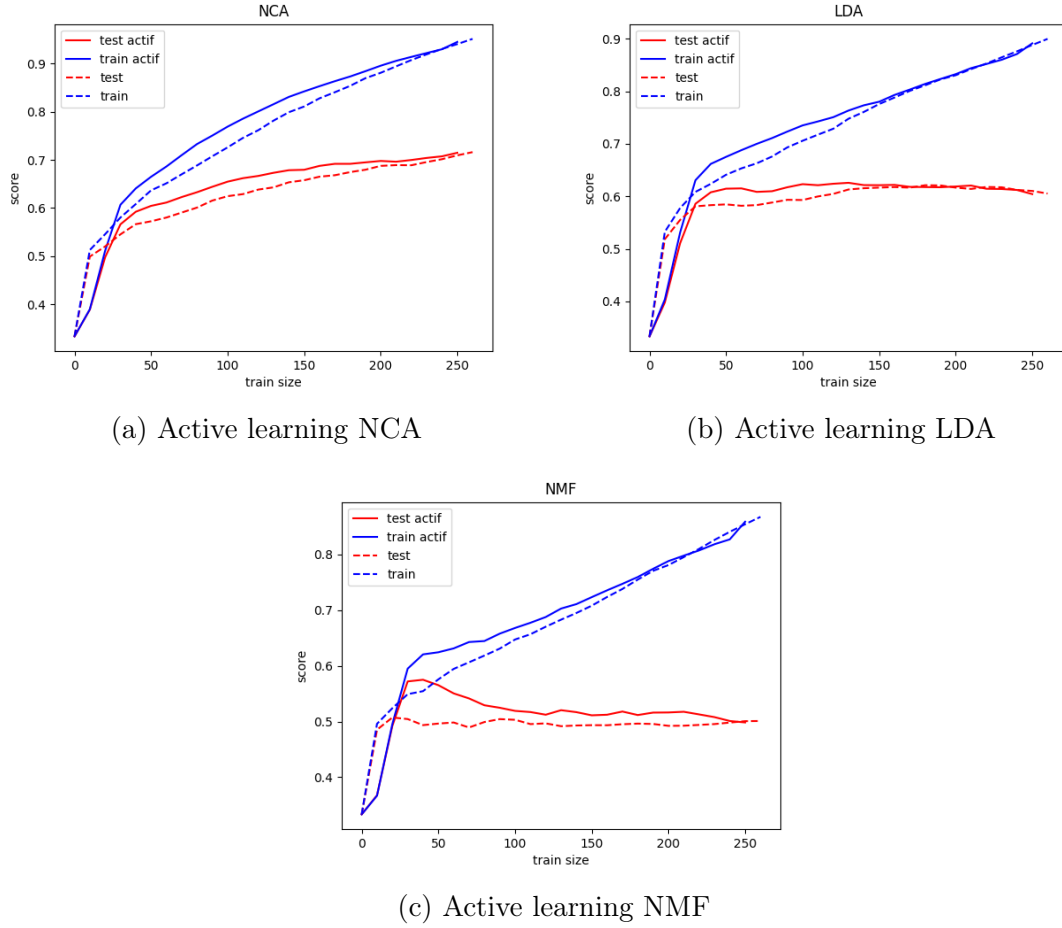


FIGURE 15 – Active learning

Cette méthode d'optimisation ne semble pas avoir un effet sur ces trois modèles.

7 Conclusion

De toutes les différentes méthodes de réduction dimensionnelle, nous avons remarqué que les divisions linéaires sont de loin les plus performantes sur nos données. Mais aussi que les algorithmes de réduction dimensionnelle supervisée ont des résultats bien meilleurs que les projections non-supervisée. Ce qui n'est pas particulièrement choquant, connaître en amont le label de nos données permet une séparations des groupes bien plus évidente que si nous ne connaissons pas le label de celles-ci. Finalement, c'est l'analyse de composantes de voisinages qui performe le mieux avec un score de 71% sur nos données test et 95% sur nos données d'entraînement. De nombreuses dimensions du classeur n'ont pas été explorées. Notamment la pondération des attributs. Ce qui devrait jouer un rôle majeur dans les capacités de réduction de la dimensionnalité de l'algorithme. Il est donc très envisageable d'augmenter grandement

le score du classer. L'apprentissage actif que nous avons essayé n'as pas été concluant. Nous n'avons pas réussi à gagner en complexité. Mais cela souligne la bonne performance des classeurs dans un premier temps.

Il serait intéressant de faire une analyse des performances des différents algorithmes de réduction de dimensionnalité de données supervisées. En effet, comme constaté, ces algorithmes ont des bien meilleurs performances que ceux pour des données non-supervisées et, il en existe beaucoup plus que les trois testé. De plus, il pourrait être particulièrement enrichissant d'approfondir le sujet de l'apprentissage actif. Malgré que nos expérimentations n'ont pas donné de résultats, un TER entier pourrait y être dédié.

C'est donc avec frustration causée par la sensation d'inachèvement que je termine mon TER. Nous avons pas eu le temps de faire le tour de tous les différents types de réduction dimensionnelle.

8 Remerciements

Je tiens à remercier Monsieur Butez pour son encadrement tout au long de l'année ainsi que pour m'avoir accordé la possibilité d'effectuer ce TER. Je remercie aussi Monsieur Chambefort pour ses indications qui m'on grandement aidé.