



MODÈLES DE MÉLANGE GAUSSIEN

*Mohcine El Harras
Mickaël El Alam
Guilhem De Viry*

8 Avril 2021

Contents

1	Introduction	2
2	Modèle de mélange	2
3	Simulation d'un modèle de mélange	2
3.1	Configuration 1	2
3.2	Configuration 2	3
3.3	Configuration 3	4
4	CEM pour les modèles parcimonieux	5
4.1	Forme sphérique car $A=I$, orientation identique, proportions et volumes identiques $[\pi, \lambda I]$	7
4.2	Forme sphérique car $A=I$, orientation identique, proportions différentes et volumes identiques $[\pi_k, \lambda I]$	7
4.3	Forme sphérique car $A=I$, orientation identique, proportions identiques et volumes différents $[\pi, \lambda_k I]$	8
4.4	Forme sphérique car $A=I$, orientation identique, proportions et volumes différents $[\pi_k, \lambda_k I]$	9
5	Comparaison de l'algorithme CEM avec K-means	10
5.1	Choix du Dataset	10
5.2	Comparaison des 2 algorithmes sur des données réelles	11
5.3	Données simulées	12
5.4	Comparaison des 2 algorithmes sur des données simulées	13
5.5	Bonus	16
6	Apprentissage supervisé avec les GMM	17
6.1	Adaptation des GMM à l'apprentissage supervisé	17
6.2	Régression logistique	17
6.3	Comparaison des deux méthodes	18
6.3.1	Données simulées	18
6.3.2	Données réelles	19
7	Bonus : Apprentissage semi supervisée avec les modèles de mélange	21
8	Conclusion	22

Abstract

Dans le cadre de notre cours intitulé "Data Analysis and Machine Learning", nous avons étudié dans le chapitre 3 les modèles de mélange. ce rapport est une synthèse des algorithmes de K-means, EM et CEM appliqués aux modèles de mélange gaussien. Tous les codes python utilisés sont exécutables sur Jupyter Notebook/Lab ou directement sur Google Colaboratory, de plus, ils sont disponibles sur Github¹

1 Introduction

Dans le monde industriel, les entreprises peu importe leur domaine d'activité disposent d'un très grand nombre de données semi labelisées stockées dans de grosses bases de données souvent non exploitées. En effet, la classification des données avec les algorithmes d'apprentissage non supervisé souvent appelée "clustering" est une technique qui permet d'exploiter les données non labelisées et d'en extraire des informations utiles, elle commence effectivement à se déployer chez plusieurs industriels qui sont conscients de l'impact et le potentiel que peuvent avoir les données dans leur activité.

2 Modèle de mélange

Un modèle de mélange gaussien (usuellement abrégé par l'acronyme anglais GMM (pour Gaussian Mixture Model) est un modèle statistique exprimé selon une densité mélange. Il sert usuellement à estimer paramétriquement la distribution de variables aléatoires en les modélisant comme une somme de plusieurs gaussiennes (appelées noyaux). Il s'agit alors de déterminer la variance, la moyenne et l'amplitude de chaque gaussienne. Ces paramètres sont optimisés selon un critère de maximum de vraisemblance pour approcher le plus possible la distribution recherchée. Cette procédure se fait le plus souvent itérativement via l'algorithme espérance-maximisation (EM).

$$g(\mathbf{x}, \Phi) = \sum_{k=1}^g \pi_k f(\mathbf{x}, \boldsymbol{\theta}_k) \quad (1)$$

avec $f(\mathbf{x}, \boldsymbol{\theta}_k)$, la loi normale multidimensionnelle paramétrée par $\boldsymbol{\theta}_k$

3 Simulation d'un modèle de mélange

3.1 Configuration 1

On considère les 2 distributions gaussiennes suivantes :

$$\begin{aligned} N(\mu_1 = 0, \sigma_1 = 0.1); \pi_1 = 0.4; \\ N(\mu_2 = 0.2, \sigma_2 = 0.2); \pi_2 = 0.6; \end{aligned}$$

¹https://github.com/mohcineelharras/daml_project/blob/main/project/daml_project.ipynb

On peut également les définir sous le format suivant :

- Centre : $\mu_1 = 0$; Forme : $A_1 = 1$; Orientation : $D_1 = 1$; Volume : $\lambda_1 = 0.1$; Proportion : $\pi_1 = 0.4$;
- Centre : $\mu_2 = 0.2$; Forme : $A_2 = 1$; Orientation : $D_2 = 1$; Volume : $\lambda_2 = 0.2$; Proportion : $\pi_2 = 0.6$;

On obtient les résultats suivants :

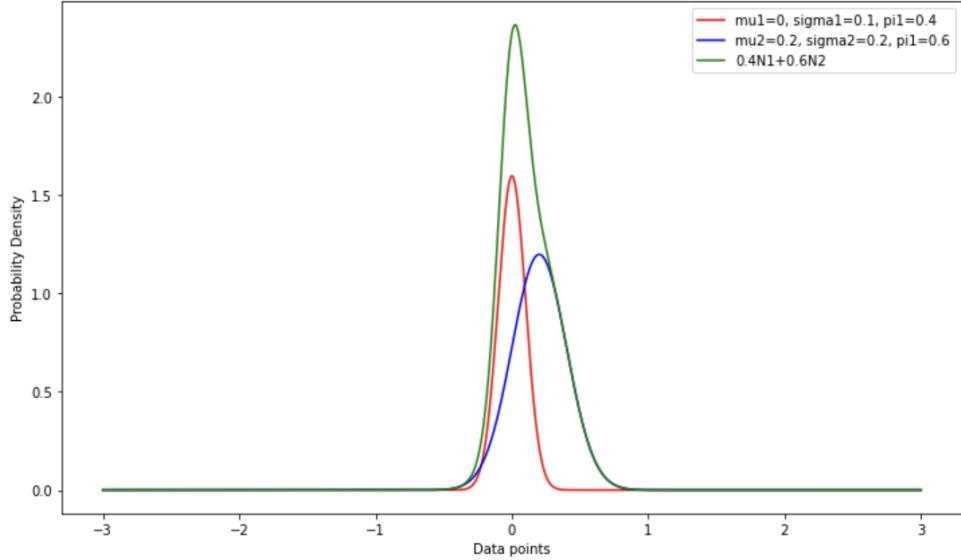


Figure 1: Simulation de la première configuration

3.2 Configuration 2

On considère les 2 distributions gaussiennes suivantes :

$$N(\mu_1 = 0.5, \sigma_1 = 0.5); \pi_1 = 0.2; \\ N(\mu_2 = 1.2, \sigma_2 = 0.4); \pi_2 = 0.8;$$

On peut également les définir sous le format suivant :

- Centre : $\mu_1 = 0.5$; Forme : $A_1 = 1$; Orientation : $D_1 = 1$; Volume : $\lambda_1 = 0.5$; Proportion : $\pi_1 = 0.4$;
- Centre : $\mu_2 = 1.2$, Forme : $A_2 = 1$; Orientation : $D_2 = 1$; Volume : $\lambda_2 = 0.4$; Proportion : $\pi_2 = 0.6$;

On obtient les résultats suivants :

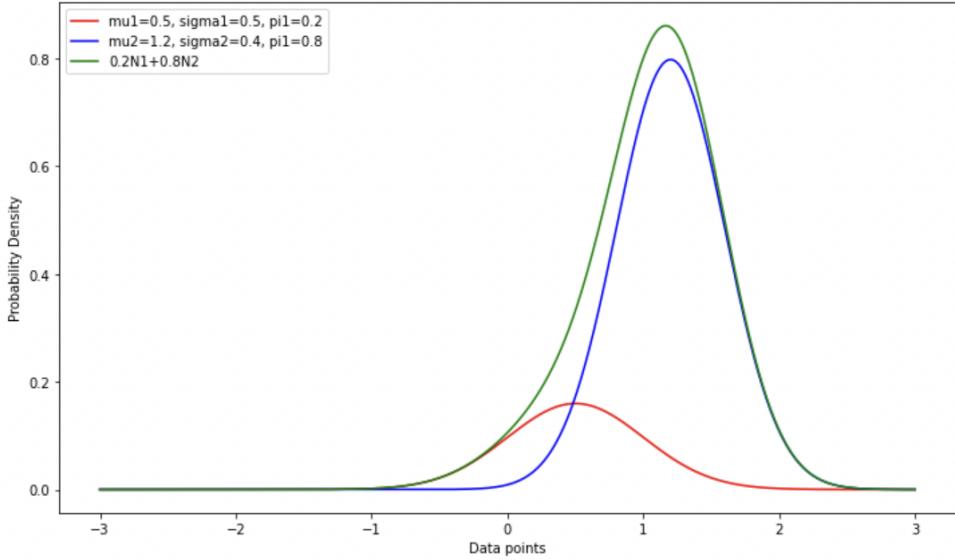


Figure 2: Simulation de la deuxième configuration

3.3 Configuration 3

On considère les 2 distributions gaussiennes suivantes :

$$N(\mu_1 = 0, \text{diag}(\sigma_1^2)) = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}; \pi_1 = 0.4;$$

$$N(\mu_2 = 0.2, \text{diag}(\sigma_2^2)) = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}; \pi_2 = 0.6;$$

On peut également les définir sous le format suivant :

- *Centre* : $mu_1 = (0 \ 0)$; *Forme* : $A_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$; *Orientation* : $D_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$; *Volume* : $\lambda_1 = 0.5$; *Proportion* : $\pi_1 = 0.4$;
- *Centre* : $mu_2 = (1 \ 1)$, , *Forme* : $A_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$; *Orientation* : $D_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$; *Volume* : $\lambda_2 = 1$; *Proportion* $\pi_2 = 0.6$;

On obtient les résultats suivants :

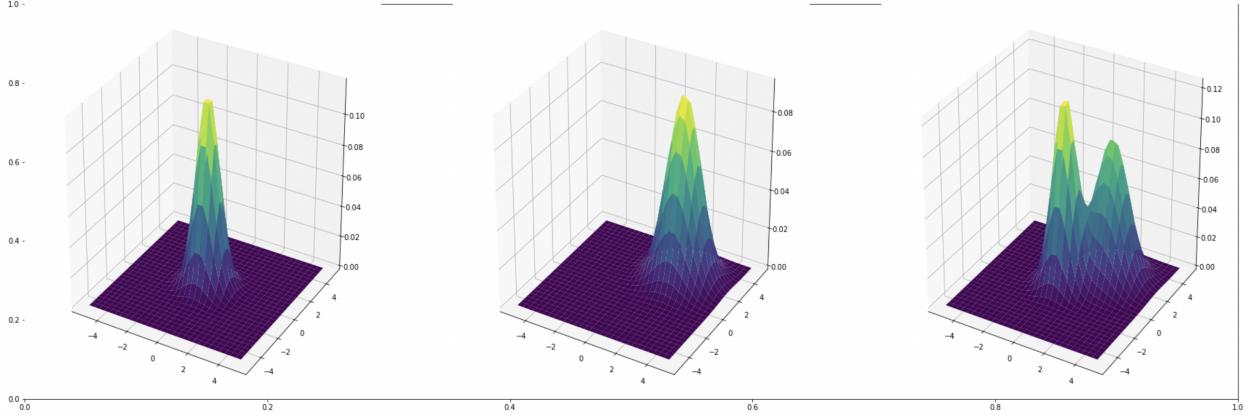


Figure 3: Simulation de la troisième configuration

4 CEM pour les modèles parcimonieux

L'algorithme "Expectation Maximisation" ou EM est un algorithme itératif qui permet de déterminer les paramètres d'un modèle de mélange en maximisant la vraisemblance.

- Initialisation : Affecter à θ_0 une valeur initiale
- E-Step : calculer la probabilité pour un individu x_i d'appartenir à une classe k :

$$t_{ik} = \frac{\pi_k^{(c)} f(x_i, \theta_k^{(c)})}{\sum_{\ell=1}^g \pi_\ell^{(c)} f(x_i, \theta_\ell^{(c)})} \quad (2)$$

- M-Step : Estimer une nouvelle valeur de θ_{i+1} pour maximiser la vraisemblance, en d'autres termes dans le cas d'un mélange gaussien :

$$\hat{\mu}^{(c+1)} = \sum_{i=1}^n \frac{t_{ik}^{(c)} x_i}{n} \quad (3)$$

$$(\hat{\sigma}^2)^{(c+1)} = \frac{1}{n} \frac{\sum_{i=1}^n t_{ik}^{(c)} (x_i - \mu_k^{(c+1)})^2}{\sum_{i=1}^n t_{ik}^{(c)}} \quad (4)$$

$$\pi_k^{(c+1)} = \frac{1}{\sum_{i=1}^n t_{ik}^{(c)}} \quad (5)$$

C'est un algorithme itératif, donc on arrête l'algorithme quand on atteint la précision ϵ souhaitée, en d'autres termes lorsque l'écart de la vraisemblance entre 2 itérations est inférieur à ϵ .

L'algorithme CEM est un algorithme EM modifié. En d'autres termes une nouvelle étape est rajoutée appelée C ou Classification où chaque point x_i sera affecté à une classe k, donc les t_{ik} vont être remplacés par $z_{ik}=1$ si $t_{ik} \geq 0.5$ et $z_{ik}=0$ sinon.

- Initialisation : Affecter à θ_0 une valeur initiale

- E-Step : calculer la probabilité pour un individu x_i d'appartenir à une classe.
- C-Step : $z(c) = (\max)_k(t_{ik}^{(c)})$ pour tout i
- M-Step : Estimer une nouvelle valeur de θ_{i+1} pour maximiser la vraisemblance, en d'autres termes dans le cas d'un mélange gaussien

L'a Ici une comparaison des 2 algorithmes (CEM à gauche et EM à droite) sur le même jeu de données

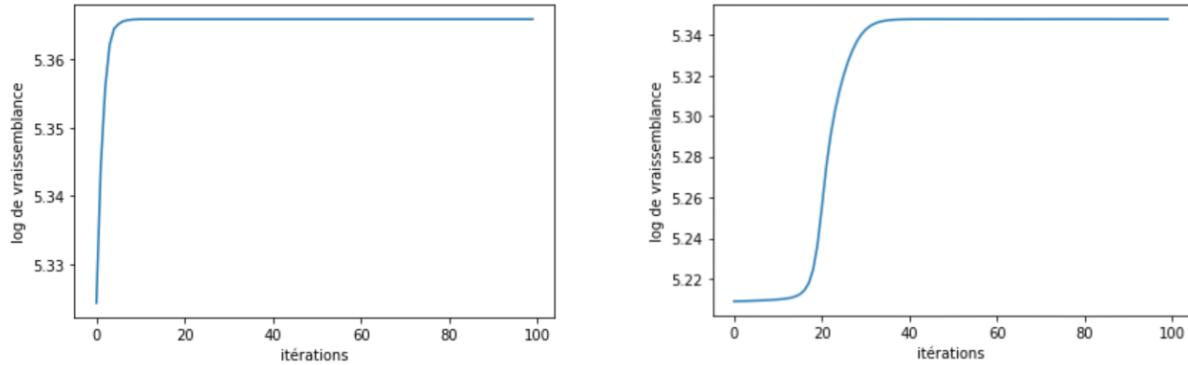


Figure 4: CEM vs EM ; Évolution du log de la vraisemblance

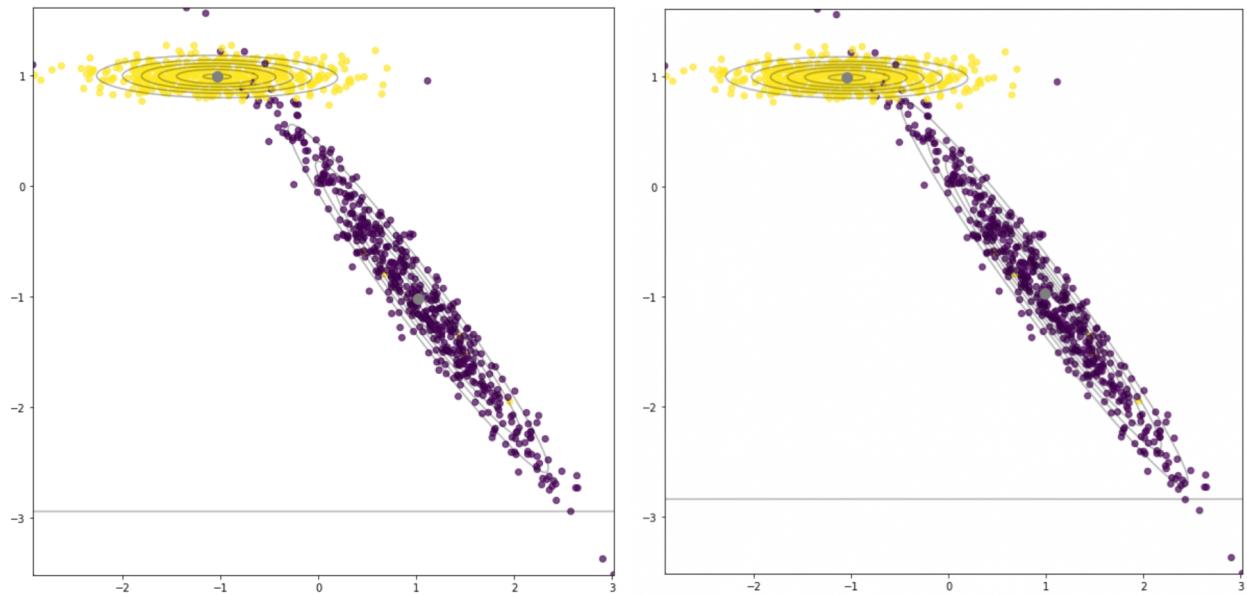


Figure 5: CEM vs EM ; Lignes de niveau pour la classification

Maintenant, on retrouve ci-dessous voir les tracés des distributions avec des lignes de niveau pour les 4 modèles parcimonieux différents de formes sphériques et orientations identiques. L'algorithme CEM après 20 itérations donne les résultats suivants :

4.1 Forme sphérique car $A=I$, orientation identique, proportions et volumes identiques $[\pi, \lambda I]$

$$\begin{aligned}\mu_k &= \bar{x} = \sum_{i=1}^n \frac{x_i}{n} \\ \pi_k &= \frac{1}{p} \\ \lambda_k &= \frac{S_W}{p}\end{aligned}$$

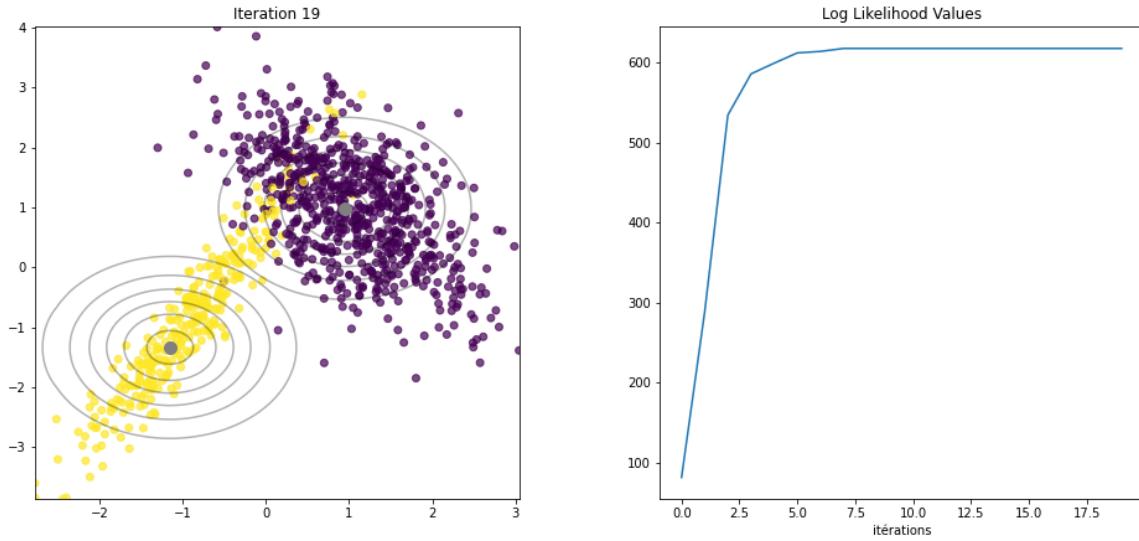


Figure 6: Proportions et volumes identiques $[\pi, \lambda I]$

$\text{Trace}(SW) \rightarrow$ classes sphériques, même proportion et même volume.
On tombe exactement dans le cas K-means, CEM = k-means

4.2 Forme sphérique car $A=I$, orientation identique, proportions différentes et volumes identiques $[\pi_k, \lambda I]$

$$\begin{aligned}\mu_k &= \bar{x} = \sum_{i=1}^n \frac{x_i}{n} \\ \pi_k &= \frac{n_k}{n} \\ \lambda_k &= \frac{S_W}{p}\end{aligned}$$

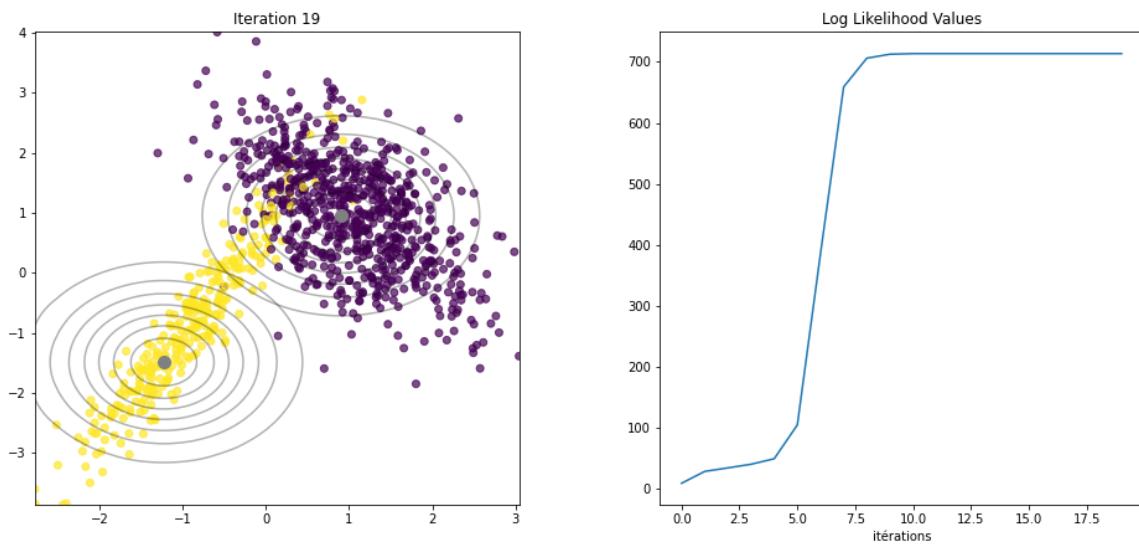


Figure 7: Proportions différentes et volumes identiques $[\pi_k, \lambda I]$

4.3 Forme sphérique car $A=I$, orientation identique, proportions identiques et volumes différents $[\pi, \lambda_k I]$

$$\begin{aligned}\mu_k &= \bar{x} = \sum_{i=1}^n \frac{x_i}{n} \\ \pi_k &= \frac{1}{p} \\ \lambda_k &= \frac{S_k}{p}\end{aligned}$$

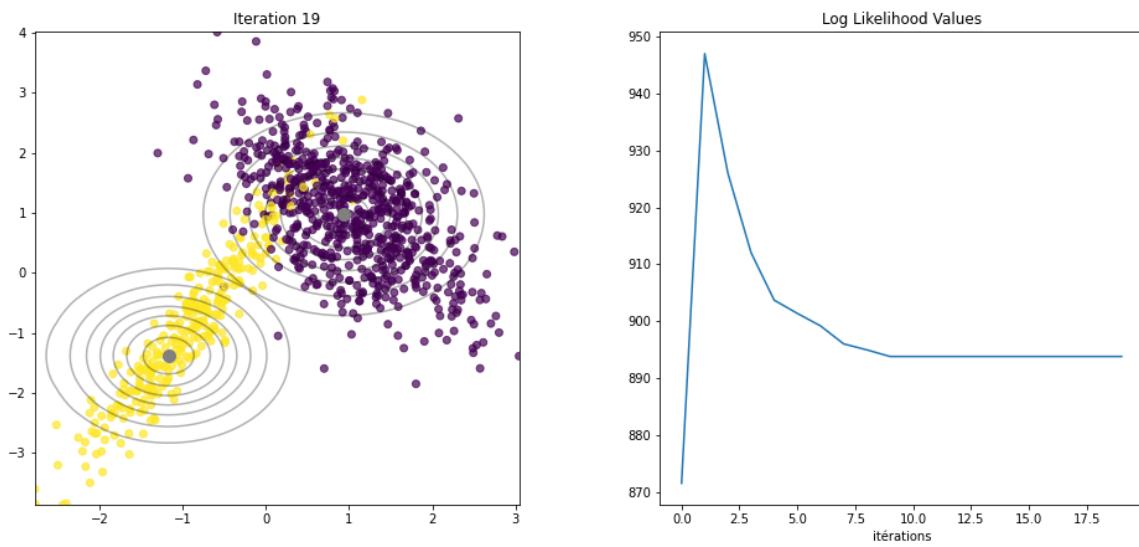


Figure 8: Proportions identiques et volumes différents $[\pi, \lambda_k I]$

4.4 Forme sphérique car $A=I$, orientation identique, proportions et volumes différents $[\pi_k, \lambda_k I]$

$$\begin{aligned}\mu_k &= \bar{x} = \sum_{i=1}^n \frac{x_i}{n} \\ \pi_k &= \frac{n_k}{n} \\ \lambda_k &= \frac{S_k}{p}\end{aligned}$$

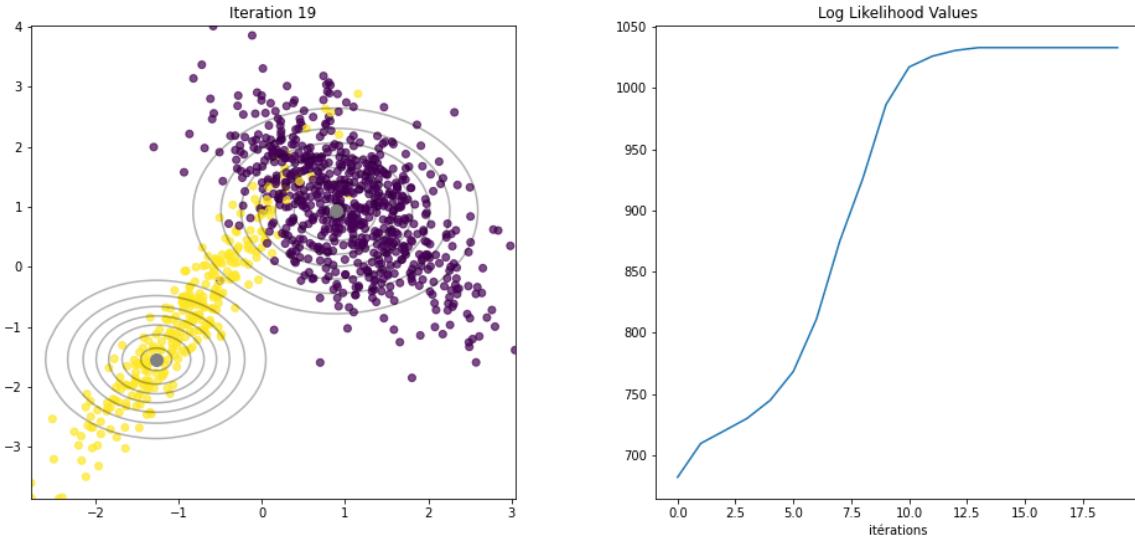


Figure 9: Proportions et volumes différents $[\pi_k, \lambda_k I]$

5 Comparaison de l'algorithme CEM avec K-means

5.1 Choix du Dataset

Le Dataset Audit a été choisi parce qu'il contient un vecteur de label y binaire (0 ou 1), des features numériques et ainsi, il est intéressant d'essayer des techniques de classification (clustering) pour identifier les 2 classes Risk=0 (représente 61% du jeu de données) et Risk=1 (39%), en d'autres termes, pour identifier les entreprises frauduleuses sur la base des facteurs de risque actuels et historiques. Les features de ce jeu de données sont :

[‘Sector score’, ‘LOCATION ID’, ‘PARA A’, ‘Score A’, ‘Risk A’, ‘PARA B’, ‘Score B’, ‘Risk B’, ‘TOTAL’, ‘numbers’, ‘Score B.1’, ‘Risk C’, ‘Money Value’, ‘Score MV’, ‘Risk D’, ‘District Loss’, ‘PROB’, ‘RiSk E’, ‘History’, ‘Prob’, ‘Risk F’, ‘Score’, ‘Inherent Risk’, ‘CONTROL RISK’, ‘Detection Risk’, ‘Audit Risk’, ‘Risk’]

Tout d'abord, nous allons visualiser les données réelles sur un espace 2D en faisant de la réduction de dimension avec 2 méthodes différentes :

- LDA : Analyse discriminante linéaire (tient en compte les labels y pour trouver l'espace de projection)
- PCA : Analyse des composantes principales (méthode brut)

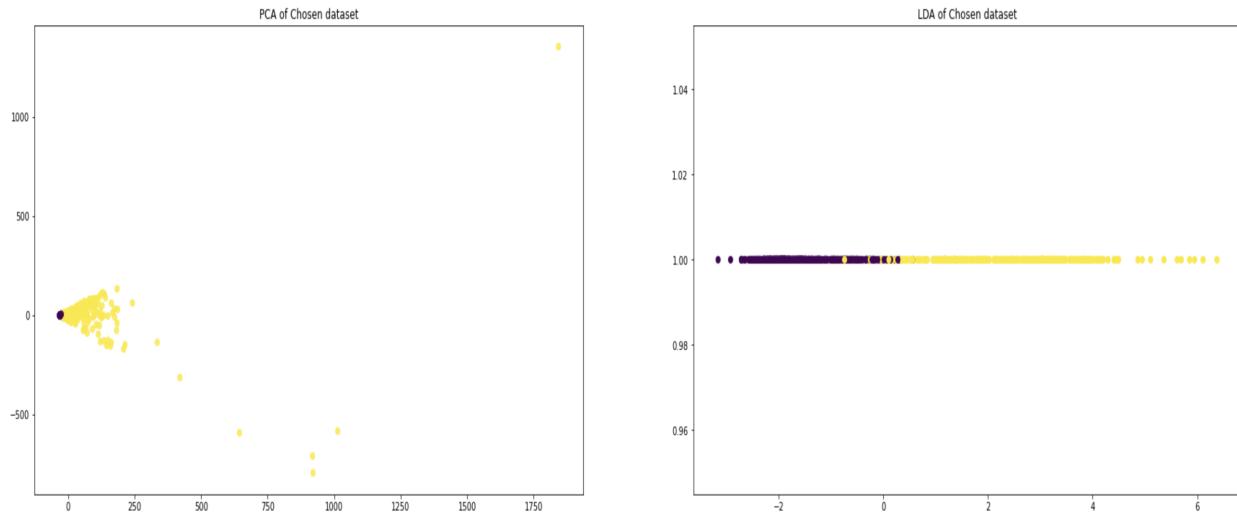


Figure 10: PCA et LDA ; True labels

5.2 Comparaison des 2 algorithmes sur des données réelles

Les 2 algorithmes ont des comportements différents sur les données réelles, les plots 1 et 2 ci-dessous représentent respectivement, le résultats de classification avec la méthode des k-means et la méthode de CEM.

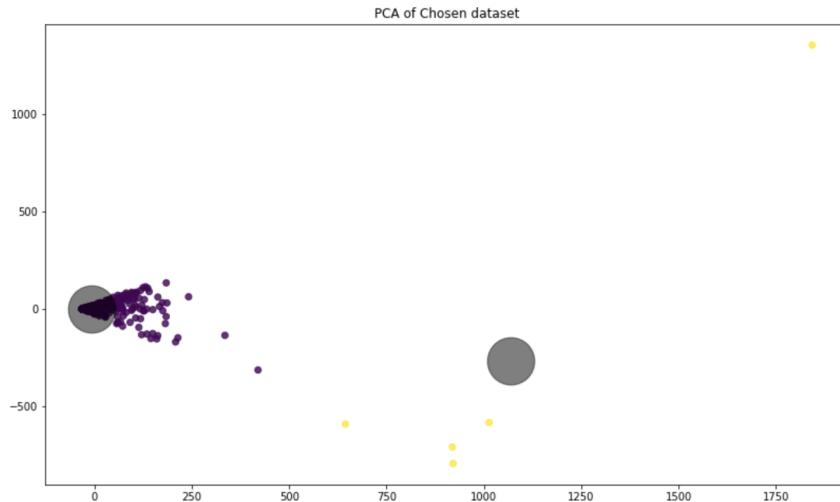


Figure 11: K-means données réelles

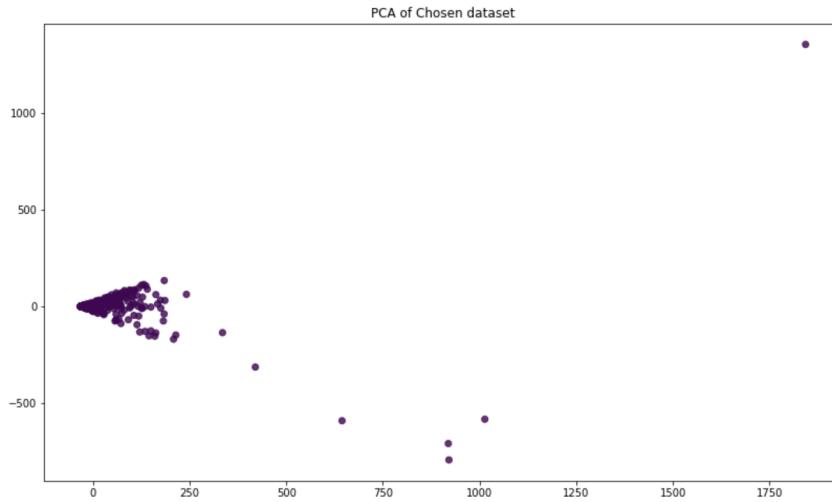


Figure 12: CEM données réelles

L'algorithme CEM converge en mettant toutes les sorties à 0, il trouve un maximum local de vraisemblance et il n'arrive pas à atteindre le maximum global.

Comme les points des 2 classes sont très rapprochés, Le k-means n'arrive pas à détecter les différences entre les 2 classes, ainsi il converge vers une configuration où il sépare l'espace en 2, il définit une classe "normale" (en violet) et une autre "outliers" ou points aberrants (en jaune) qui s'éloignent beaucoup de la majorité des données.

Lorsqu'on représente le jeu de données réel comme un nuage de points sur ses deux composantes principales à l'aide de l'analyse des composantes principales, on remarque que les classes ne sont pas bien séparées. Comme vu dans le cours, on peut conclure que les méthodes CEM et K-means ne sont pas adaptées à des jeux de données pas bien séparés.

5.3 Données simulées

Nous allons utiliser la fonction `make_classification` de la bibliothèque Scikit-learn. Cette fonction permet de générer aléatoirement un problème à n classes de classification.

Ci-dessous, une visualisation de ces données :

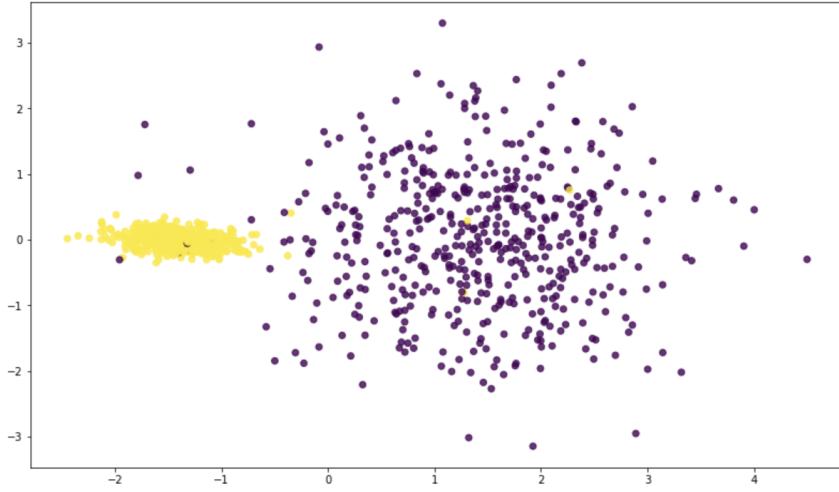


Figure 13: Simulation de données de 2 classes bien séparées

5.4 Comparaison des 2 algorithmes sur des données simulées

On rappelle un certain nombre de métriques permettant d'évaluer les modèles de classement. La matrice de confusion est une matrice qui contient les indicateurs suivants : TP : True Positive ; FN : False Negative ; TN : True Negative ; FP : False Positive

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}; \quad (6)$$

$$Precision = \frac{TP}{TP + FP}; \quad (7)$$

$$Rappel = \frac{TP}{TP + FN}; \quad (8)$$

$$F1 - score = \frac{2 \times Precision \times recall}{Precision + recall}; \quad (9)$$

Ainsi, pareillement pour le jeu de données réel, on compare les labels réels des données avec ceux prédits avec K-means et CEM.

On compare les 2 algorithmes lorsque les données sont :

- err = 5% ; $\pi_1 = 50$; $\pi_2 = 50$;

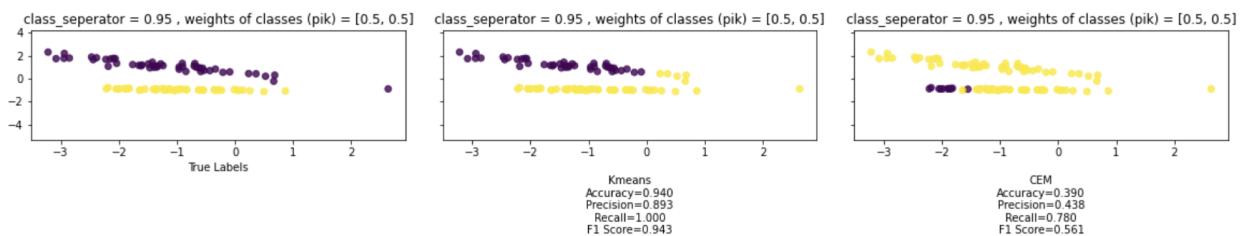


Figure 14: err = 5% ; $\pi_1 = 50$; $\pi_2 = 50$

- $\text{err} = 5\% ; \pi_1 = 40; \pi_2 = 60;$

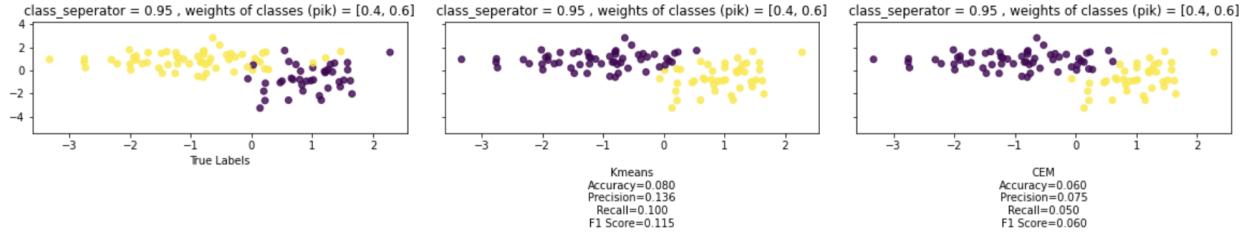


Figure 15: $\text{err} = 5\% ; \pi_1 = 40; \pi_2 = 60$

- $\text{err} = 5\% ; \pi_1 = 15; \pi_2 = 85;$

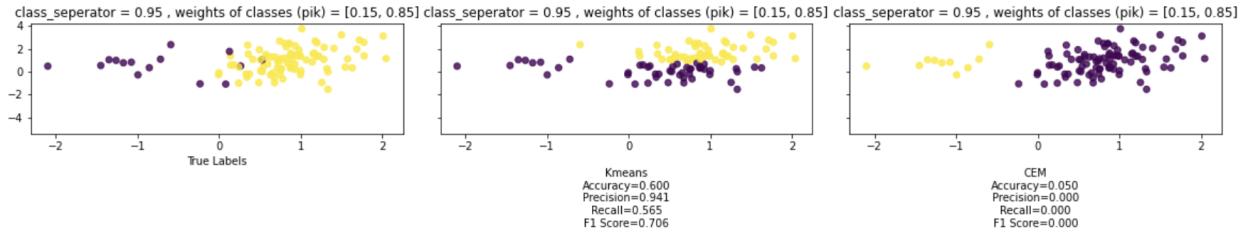


Figure 16: $\text{err} = 5\% ; \pi_1 = 15; \pi_2 = 85;$

- $\text{err} = 15\% ; \pi_1 = 50; \pi_2 = 50;$

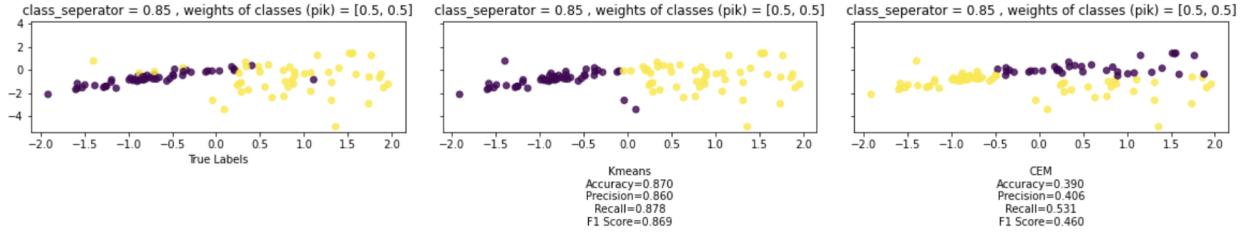


Figure 17: $\text{err} = 15\% ; \pi_1 = 50; \pi_2 = 50;$

- $\text{err} = 15\% ; \pi_1 = 40; \pi_2 = 60;$

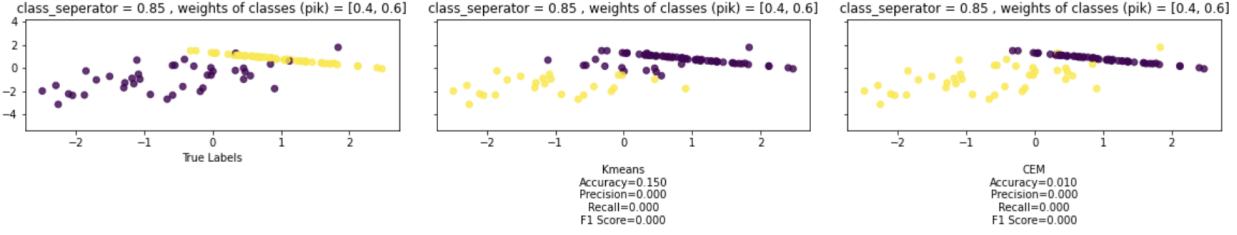


Figure 18: $\text{err} = 15\% ; \pi_1 = 40; \pi_2 = 60;$

- $\text{err} = 15\% ; \pi_1 = 15; \pi_2 = 85;$

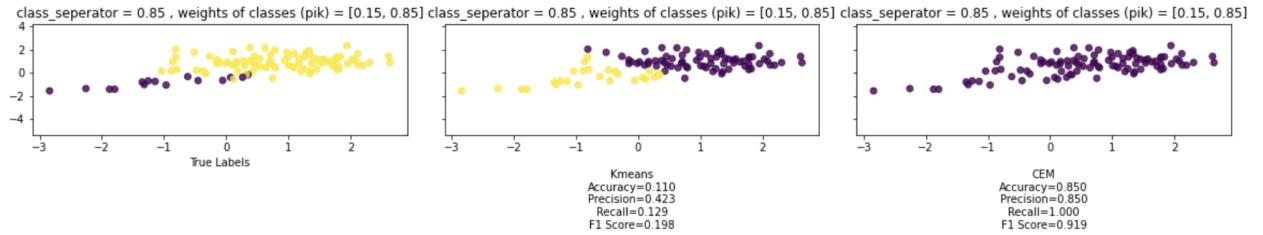


Figure 19: $\text{err} = 15\% ; \pi_1 = 15; \pi_2 = 85;$

- $\text{err} = 30\% ; \pi_1 = 50; \pi_2 = 50;$

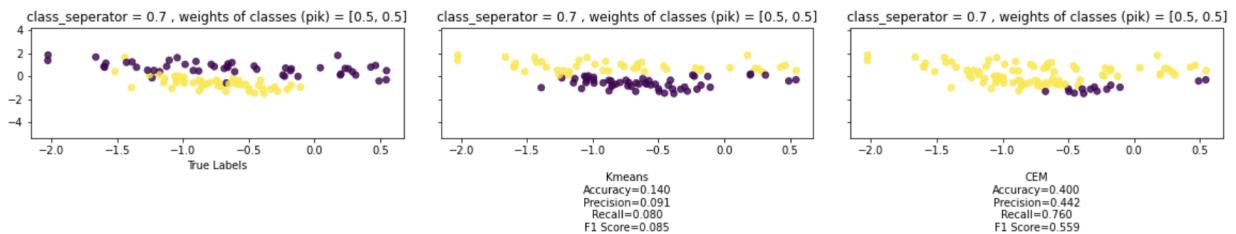


Figure 20: $\text{err} = 30\% ; \pi_1 = 50; \pi_2 = 50;$

- $\text{err} = 30\% ; \pi_1 = 40; \pi_2 = 60;$

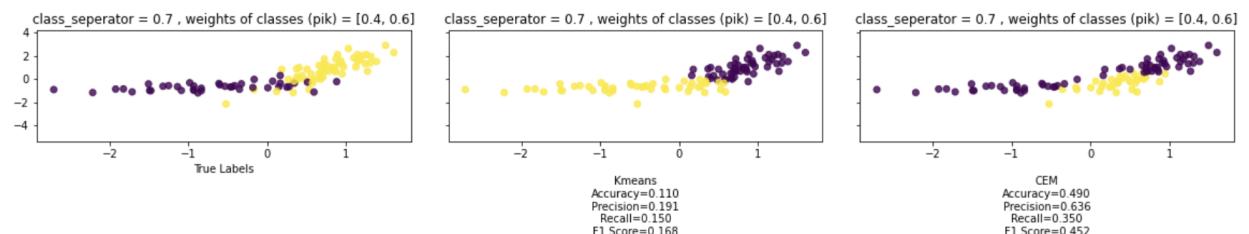


Figure 21: $\text{err} = 30\% ; \pi_1 = 40; \pi_2 = 60;$

- $\text{err} = 30\% ; \pi_1 = 15; \pi_2 = 85;$

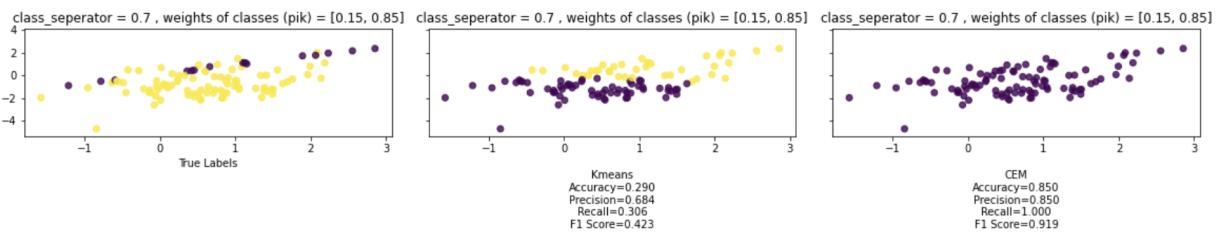


Figure 22: $\text{err} = 30\% ; \pi_1 = 15; \pi_2 = 85;$

Comme pour le jeu de données réel, on remarque que plus il y a un chevauchement de classes, moins bien sont les résultats obtenus avec les méthodes de CEM et K-means. On peut conclure que les méthodes CEM et K-means ne sont pas adaptées à des jeu de données avec un chevauchement des classes.

5.5 Bonus

Dans la figure ci-dessus, on compare l'algorithme EM et CEM sur des données avec une erreur de chevauchement de classe de 90%.

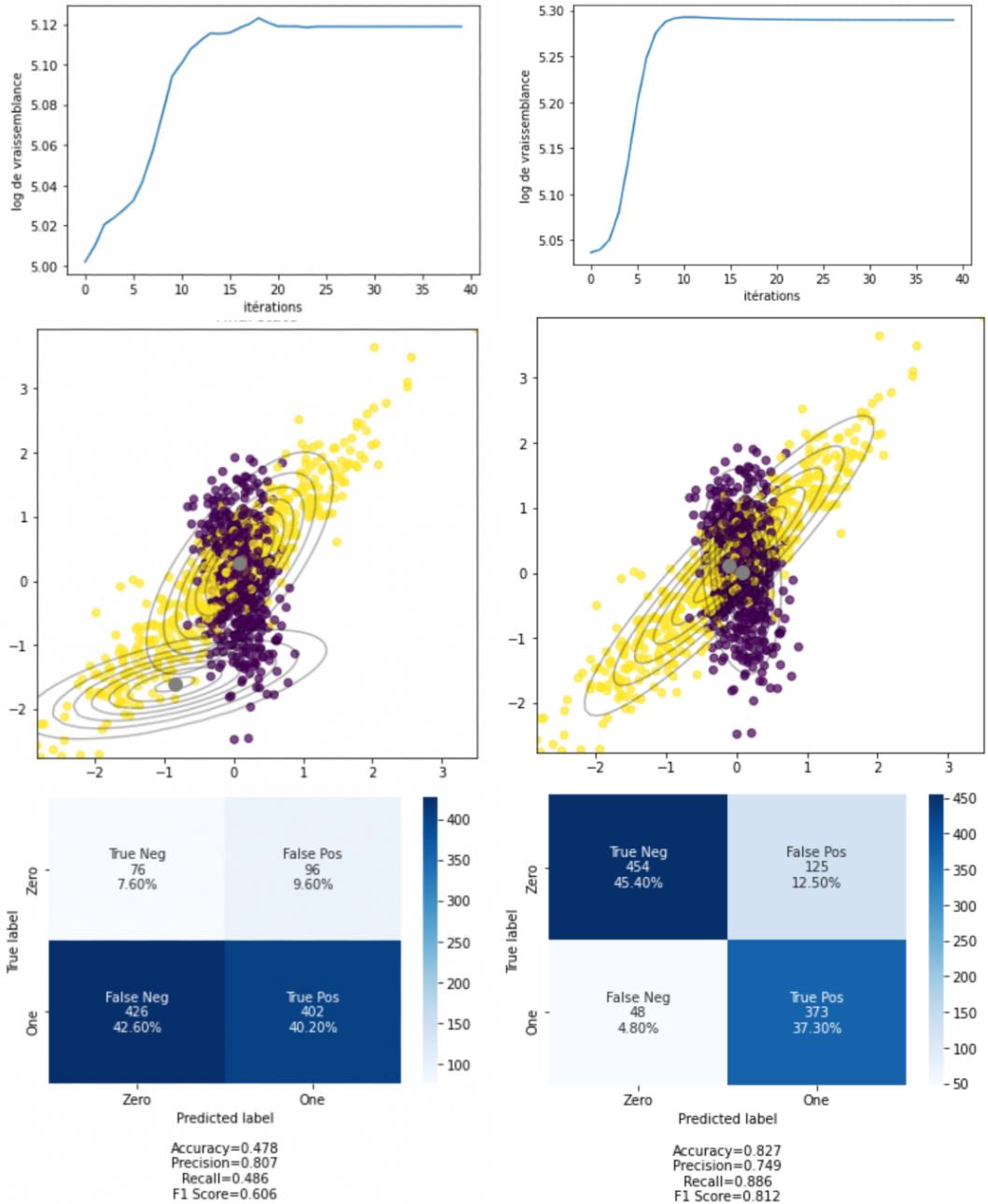


Figure 23: CEM et EM

6 Apprentissage supervisé avec les GMM

6.1 Adaptation des GMM à l'apprentissage supervisé

Quand les i labels des i sont connus, on sait qu'on a 2 estimateurs sans biais pour μ_k et σ_k qui maximisent la vraisemblance et ainsi on peut avoir la loi de probabilité. Par conséquent, on peut calculer la probabilité d'appartenance à une classe k.

$$\hat{\mu} = \bar{x} = \sum_{i=1}^n \frac{x_i}{n} \quad (10)$$

$$\hat{\sigma}^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2. \quad (11)$$

$$\pi_k = \frac{n_k}{n} \quad (12)$$

Ensuite, il suffit de calculer t_{ik} les probabilités d'appartenance à chaque classe k ce qui va permettre de déterminer \hat{y}_i :

$$t_{ik} = \frac{\pi_k^{(c)} f(x_i, \theta_k^{(c)})}{\sum_{\ell=1}^g \pi_\ell^{(c)} f(x_i, \theta_\ell^{(c)})} \quad (13)$$

$$\begin{aligned} Si, t_{i0} > 0.5 &\Rightarrow \hat{y}_i = 0 \\ Si, t_{i1} > 0.5 &\Rightarrow \hat{y}_i = 1 \end{aligned}$$

6.2 Régression logistique

Algorithme d'apprentissage supervisé qui permet de faire de la classification :

$$Z = \beta_0 + \beta_1 X \quad (14)$$

1. Hypothèse fondamentale (Sigmoid Function)

$$h(x) = \frac{1}{(1 + e^{-(\beta_0 + \beta_1 X)})} \quad (15)$$

$$P(class = 1) = (x) = \sigma(Z) \quad (16)$$

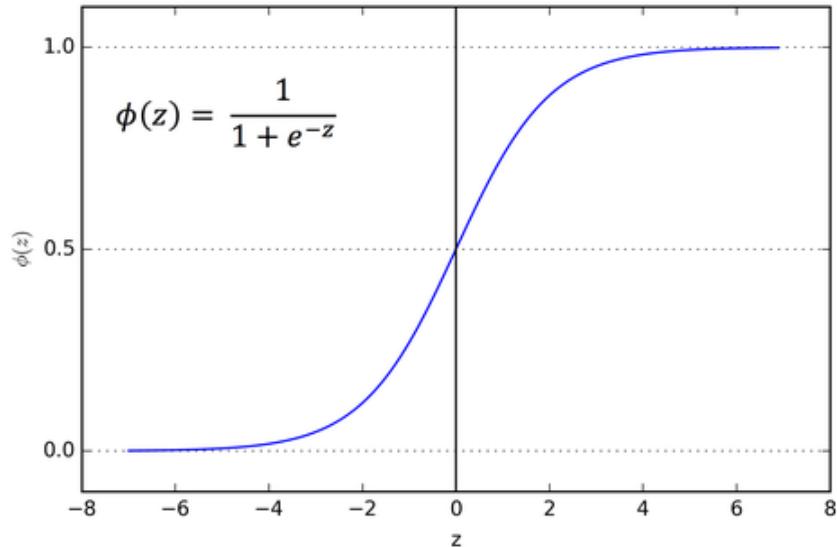


Figure 24: Sigmoid function

2. Frontière de décision

$$p \geq 0.5 \Rightarrow \text{class} = 1 \quad (17)$$

$$p < 0.5 \Rightarrow \text{class} = 0 \quad (18)$$

3. Fonction de coût: Pour des raisons de convexité, on définit la fonction de coût suivante :

$$\text{Cost}((x), y) = y \log((x))(1y) \log(1(x)) \quad (19)$$

4. Gradient Descent

$$\theta_{j+1} = \theta_j - \nabla J(\theta) \quad (20)$$

5. Modèle retenu pour la régression logistique

```
LogisticRegression(C=100000.0, class_weight=None, dual=False,
                    fit_intercept=True, intercept_scaling=1, l1_ratio=None,
                    max_iter=100, multi_class='auto', n_jobs=None, penalty='l2',
                    random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                    warm_start=False)
```

Figure 25: Hyperparamètres du modèle utilisé pour la comparaison

6.3 Comparaison des deux méthodes

6.3.1 Données simulées

Les figures ci-dessous, correspondent aux matrice de confusion pour les 2 méthodes (GMM supervisé à gauche et Régression logistique à droite)

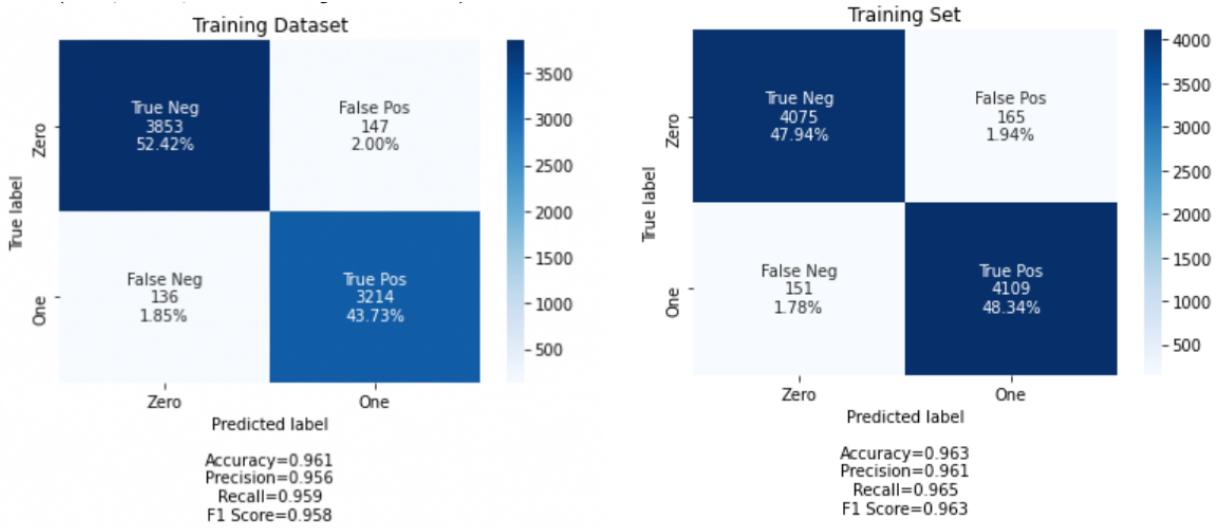


Figure 26: GMM vs Régression logistique sur le training dataset simulé

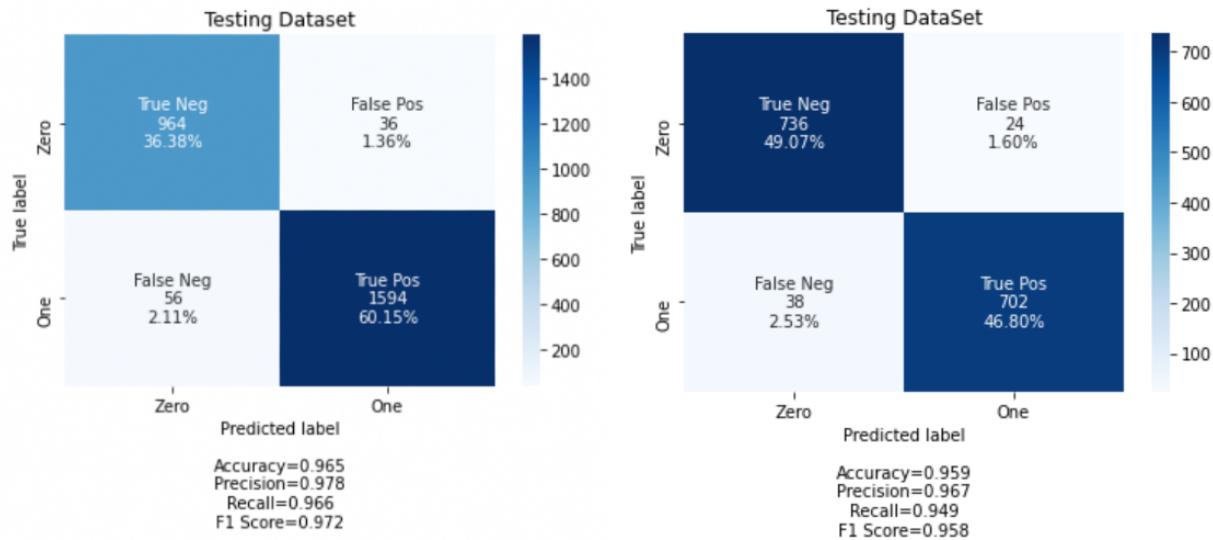


Figure 27: GMM vs Régression logistique sur le testing dataset simulé

On remarque sur les données simulées qui ont des classes bien séparées les unes des autres que les 2 algorithmes s'en sortent bien sur le jeu de données de test (pas vu par le modèle pendant l'entraînement). En effet, l'approche supervisée pour les GMM se généralise mieux que la régression logistique et permet d'avoir un F1-score = 0.972 alors que la régression logistique n'atteint que 0.958.

6.3.2 Données réelles

Toujours avec les données d'Audit dans le paragraphe précédent, on teste cette fois 2 algorithmes d'apprentissage supervisé, GMM et la régression linéaire.



Figure 28: GMM vs Régression logistique sur le training dataset réel



Figure 29: GMM vs Régression logistique sur le testing dataset réel

On remarque sur les données réelles qui sont mal séparées les 2 algorithmes s'en sortent bien sur le jeu de données de test (pas vu par le modèle pendant l'entraînement). L'approche supervisée pour les GMM se généralise moins bien que la régression logistique et permet d'avoir un F1-score = 0.942 alors que la régression logistique atteint 0.983.

7 Bonus : Apprentissage semi supervisée avec les modèles de mélange

Apprentissage supervisé pour estimer le θ_0 qui va initialiser derrière un Expectation maximum algorithme

Tout d'abord, on peut diviser le jeu de données en 2 partitions. Une première contenant les données labelisées donc des couples (x_i, y_i) et une deuxième contenant que des $(x_i, ??)$.

On commence avec le jeu de données labelisé. On peut estimer les paramètres optimaux suivants :

$$\widehat{\mu_{k,opt}} = \bar{x} = \sum_{i=1}^n \frac{x_i}{n}$$

$$\widehat{\sigma_{k,opt}}^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2$$

$$\pi_{k,opt} = \frac{n_k}{n}$$

Ensuite, on passe au jeu de données non labelisé, on utilise l'algorithme EM ou CEM pour déterminer les paramètres optimaux des distributions. L'algorithme utilisé doit être initialisé avec les $\widehat{\mu_{k,opt}}$, $\widehat{\sigma_{k,opt}}^2$, $\pi_{k,opt}$ trouvés à partir des données labelisés et on itère jusqu'à maximiser la vraisemblance comme expliqué dans la partie 4.

Enfin, on peut comparer les 2 approches supervisée et semi - supervisée, on sait d'avance que l'approche supervisée est censée donner des résultats meilleurs puisque le modèle dispose de tous les labels donc il peut estimer avec précision les paramètres des 2 distributions gaussiennes.

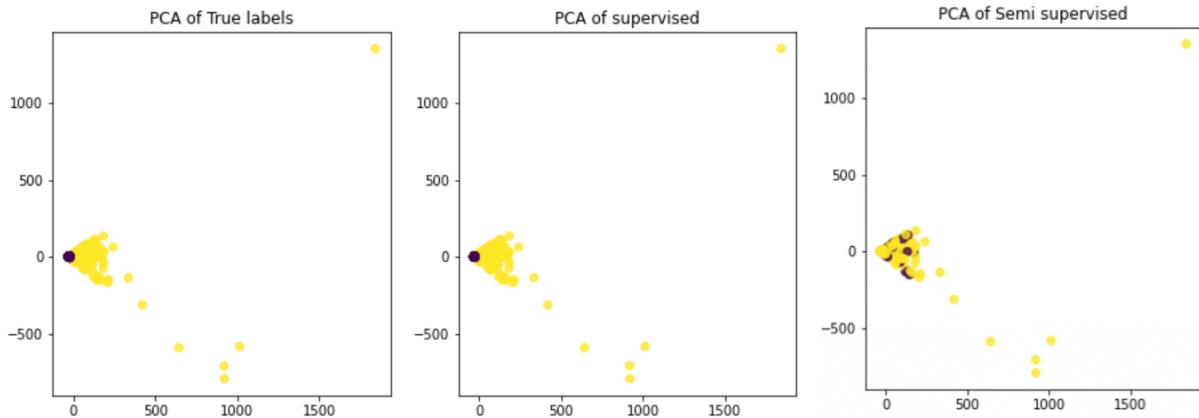


Figure 30: plot des prédictions : Apprentissage supervisé GMM vs Apprentissage semi-supervisé GMM sur le DataSet réel

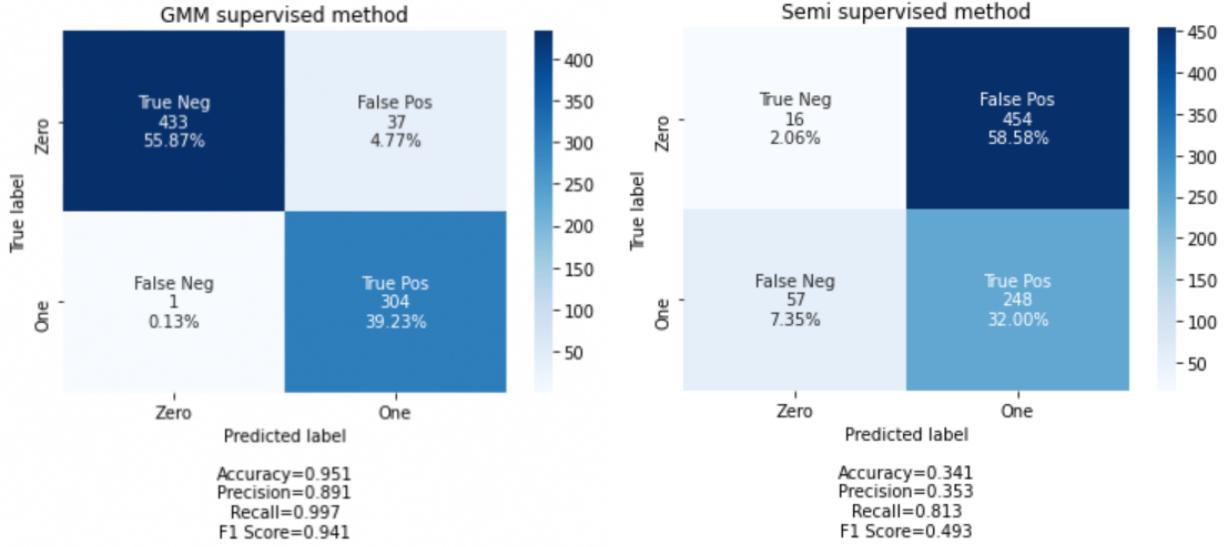


Figure 31: Matrice de confusion : Apprentissage supervisée GMM vs Apprentissage semi-supervisée GMM sur le DataSet réel

8 Conclusion

Pour conclure, les techniques de modèle de mélange gaussiens sont très intéressantes, elles permettent de classifier des données avec des classes chevauchées. Cependant, les GMM ont un inconvénient, ils utilisent l'algorithme EM ou CEM pour maximiser la vraisemblance, c'est une approche itérative donc elle peut converger rapidement vers un minimum local qui n'est pas optimal pour l'ensemble de données. Pour éviter ce problème, les GMM sont généralement initialisés avec la méthode de K-Means. Cela fonctionne généralement assez bien et améliore les clusters générés avec les K-Means. De plus, les GMM avec l'initialiseur K-Means semblent fonctionner le mieux et les clusters sont presque les mêmes que pour les données d'origine. Si on recherche la robustesse, GMM avec l'initialiseur K-Means semble être la meilleure option.

References

- [1] Paul David McNicholas, Thomas Brendan Murphy, *Parsimonious Gaussian mixture models*, (Springer Science+Business Media, LLC 2008)
- [2] Trevor Hastie, Robert Tibshirani and Jerome Friedman *The Elements of Statistical Learning Data Mining, Inference, and Prediction* (springer)
- [3] *Gaussian mixture models*, available at https://en.wikipedia.org/wiki/Mixture_model.