Université Paris Cité

Rapport Projet - Reconnaissance des Formes

Réalisé par :
Rebai Mohamed Younes
Abed Nada Fatima Zohra

Date : 17 décembre 2023

Table des matières

Chapitre 1

État de l'art

1.1 Introduction

Dans ce chapitre, nous allons présenté les diverses approches employées dans la réalisation d'un système de classification, en explorant différentes méthodes de représentation des caractéristiques. Nous allons d'abords définir les concepts fondamentaux tels que k-NN (k plus proches voisins), k-means, SvM , et l'ACP (Analyse en Composantes Principales).

1.2 Fondements théoriques

1.2.1 Apprentissage supervisé

L'apprentissage supervisé est une sous-catégorie de l'apprentissage automatique et de l'intelligence artificielle. Dans ce type d'apprentissage, les données d'entraînement fonctionnent comme un superviseur qui apprend aux machines à prédire correctement la sortie.[] Cela se fait en utilisant un algorithme pour apprendre la fonction de mappage de l'entrée à la sortie. L'apprentissage supervisé se caractérise par l'utilisation de jeux de données étiquetés qui entraînent des algorithmes permettant de classer des données ou de prédire des résultats avec précision.[]

K-Nearest Neighbor (KNN)

Le l'algorithme des k plus proches voisins (KNN) est une méthode d'apprentissage supervisé utilisée pour la classification et la régression. Il fonctionne en trouvant les k voisins les plus proches d'un point de données donné en fonction d'une mesure de distance. Pour effectuer des prédictions, l'algorithme calcule la distance entre chaque nouveau point de données dans l'ensemble de test et tous les points de données dans l'ensemble d'entraînement. Le paramètre k représente le nombre de voisins à prendre en compte lors de la prise de décision.[]

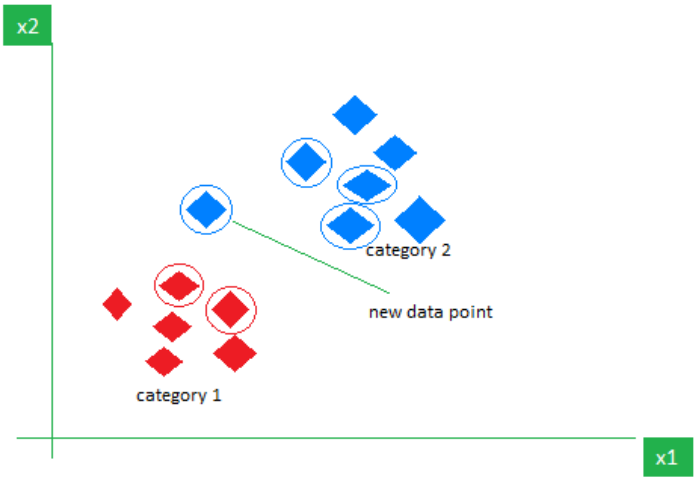

FIGURE 1.1 – K-Nearest Neighbor Knn

Pour les métriques de distance, on cite :

— **Distance euclidienne :** Il s'agit de la distance cartésienne entre deux points situés dans le plan/hyperplan. La distance euclidienne peut également être représentée comme la longueur de la ligne droite qui relie les deux points considérés. La formule de la distance euclidienne est la suivante :

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

— **Distance de Manhattan :** Cette métrique est calculée en additiont la différence absolue entre les coordonnées des points en n dimensions. Elle est calculée de la manière suivante :

$$d = |x_2 - x_1| + |y_2 - y_1|$$

Support Vector Machines (SVM)

Le Support Vector Machine (SVM) est un algorithme d'apprentissage supervisé utilisé pour la classification et la régression. Il est particulièrement efficace pour résoudre des problèmes de classification binaire, où les éléments d'un ensemble de données doivent être classés en deux groupes. L'objectif de l'algorithme SVM est de trouver la meilleure ligne ou frontière de décision qui sépare les points de données des différentes classes. Cette frontière est appelée hyperplan.

L'algorithme SVM vise à maximiser la marge, qui est la distance entre l'hyperplan et les points de données les plus proches de chaque catégorie, facilitant ainsi la distinction entre les classes de données. []

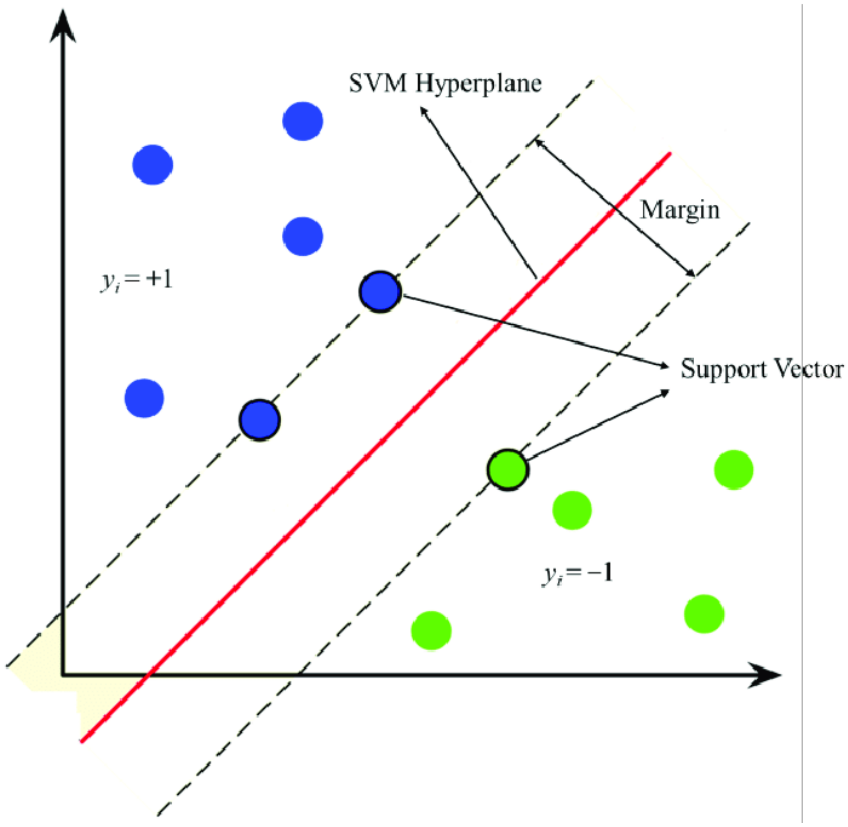

FIGURE 1.2 – Support Vector Machine

1.2.2 Apprentissage Non-supervisé

L'apprentissage non supervisé est une branche de l'apprentissage automatique caractérisée par l'analyse et le regroupement de données non étiquetées. Contrairement à l'apprentissage supervisé, où le modèle reçoit des données d'entraînement étiquetées, l'apprentissage non supervisé apprend à partir de données non étiquetées pour trouver des schémas ou des groupes dans les données, avec très peu d'intervention humaine. Cette méthode implique l'observation de plusieurs occurrences d'un vecteur X et l'apprentissage de la probabilité de distribution p(X) pour ces occurrences. Les deux principaux types de problèmes d'apprentissage non supervisé sont le clustering et l'association. Dans notre travail on s'intéresse au clustering[].

K-Means

L'algorithme K-Means est une méthode de regroupement non supervisée utilisée pour partitionner un ensemble de données en k clusters. Il fonctionne en itérativement assig0t les observations à l'un des k clusters définis par des centroïdes, où k est choisi avant le démarrage de l'algorithme. Les étapes de l'algorithme consistent à initialiser les centroïdes de manière aléatoire, puis à affecter les points de données aux clusters en fonction des centroïdes actuels, et enfin à recalculer les centroïdes en tant que moyennes des points de données dans les clusters. L'algorithme continue ces étapes itératives jusqu'à ce qu'aucune observation ne change de cluster, ou que le changement soit inférieur à un seuil spécifié[].

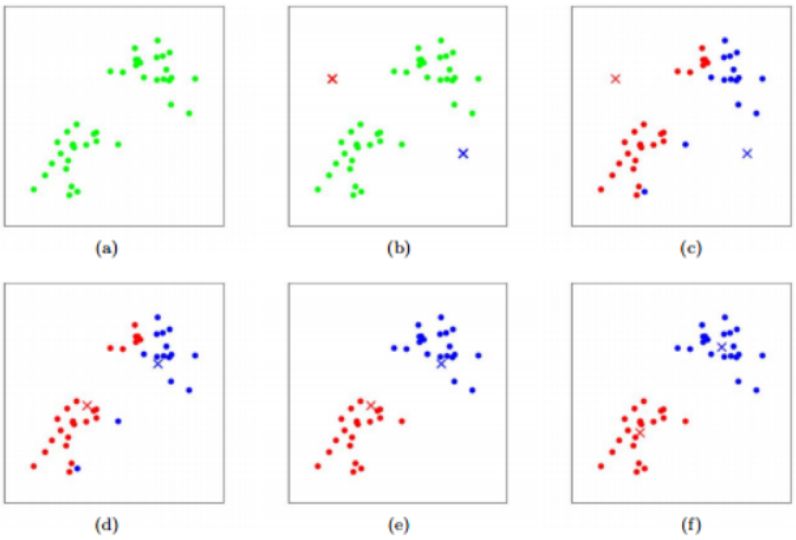

FIGURE 1.3 – Support Vector Machine

1.2.3 Réduction de dimension

La réduction de dimension est le processus de réduction du nombre de caractéristiques (ou dimensions) dans un ensemble de données tout en conservant autant d'informations que possible. Cela peut être fait pour réduire la complexité d'un modèle, améliorer les performances d'un algorithme d'apprentissage ou faciliter la visualisation des données. Il existe deux approches principales pour la réduction de dimension : la sélection de caractéristiques et l'extraction de caractéristiques. La sélection de caractéristiques implique de choisir un sous-ensemble des caractéristiques originales les plus pertinentes pour le problème à résoudre, tandis que l'extraction de caractéristiques consiste à projeter les données dans un espace de dimension inférieure en préservant les informations importantes[].

Analyse en composantes Principales

L'analyse en composantes principales (ACP) est une technique de réduction de dimension largement utilisée. Elle permet de transformer un ensemble de variables corrélées en un nouvel ensemble de variables non corrélées, appelées composantes principales, tout en préservant autant d'informations que possible. La première composante principale capture le maximum de variance dans les données, la deuxième capture le maximum de variance restante, et ainsi de suite. Ainsi, l'ACP est utile pour visualiser des données de haute dimension, détecter des schémas, réduire le bruit et comprimer les données. Elle est largement utilisée en statistique, en analyse de données et en apprentissage automatique[].
Les étapes de l'analyse en composantes principales (ACP) sont les suivantes :

1. **Standardisation des données** : Les données sont centrées et réduites pour que chaque variable ait une moyenne nulle et une variance égale à un.
2. **Calcul de la matrice de covariance** : Une matrice de covariance est calculée à partir des données standardisées pour mesurer les relations linéaires entre les variables.
3. **Décomposition de la matrice de covariance** : La matrice de covariance est ensuite décomposée en ses vecteurs et valeurs propres.

4. **Sélection des composantes principales** : Les vecteurs propres sont classés par valeurs propres associées, et les premiers vecteurs propres (composantes principales) qui capturent la plus grande variance sont sélectionnés.
5. **Projection des données** : Les données sont projetées dans l'espace des composantes principales pour obtenir un nouvel ensemble de variables décorrélées.
6. **Interprétation des composantes principales** : Les composantes principales sont interprétées pour comprendre comment les variables d'origine contribuent à chaque composante et pour identifier les schémas ou les relations dans les données.

1.3 Conclusion

Dans le chapitre suivant, nous détaillerons notre approche de mise en œuvre et fournirons une justification éclairée de nos choix.

Chapitre 2

Implémentation

2.1 Introduction

Dans ce chapitre, nous allons expliquer les détails d'implémentation de notre approche, nous commencerons d'abord par expliquer comment sont stocker nos données puis nous explorerons en détail chaque approche.

2.2 L'approche proposée

Dans notre approche nous avons utilisé l'orienté objet donc nos implémentations sont sous formes de classes (Java).

2.2.1 Organisation de données

Nous avons organisé nos données de telle manière que chaque échantillon soit associé à une structure notée s1 comprent la classe, le numéro d'échantillon et le vecteur caractéristique. Cette structure est ensuite utilisée pour simplifier la gestion de nos dossiers (F0, GFD, etc.). En d'autres termes, pour chaque type de représentation, on crée un vecteur de s1 afin de stocker tous les echantillions.
De plus, pour partitionner nos données en ensembles d'entraînement et de test, nous réalisons cette opération à chaque exécution. Cette approche est justifiée par la taille de nos données qui est relativement petite. Pour notre méthode, nous utilisons un ratio pour déterminer le nombre d'échantillons réservés à l'entraînement. Nous conservons la partie entière du résultat, et pour chaque classe, nous conservons ce nombre d'échantillons, attribuant le reste à l'ensemble de test.

2.2.2 K-Nearest Neighbor

Pour chaque élément de test, nous parcourons l'ensemble de données d'entraînement, calculons la distance, puis classons les résultats. Nous sélectionnons ensuite les k éléments avec la plus petite distance et attribuons cet élément à la classe la plus fréquente. Si les fréquences entre les classes sont égales, la classe à la téte de la liste des distances triées est conservée.

2.2.3 Kmeans

Nous initialisons k centroïdes de manière aléatoire, dans cette implémentation la valeur de k est fixée à 9 en raison de notre connaissance préalable du nombre de classes dans notre ensemble de données. Par la suite, nous évaluons les distances entre l'ensemble de nos points, représentés par nos vecteurs caractéristiques, et ces centroïdes. Une fois les clusters obtenus, nous recalculons les nouveaux centroïdes en utilisant la moyenne, et répétons cette étape de manière itérative. Les critères d'arrêt incluent la stabilité des clusters ou un nombre d'itérations spécifié par l'utilisateur.

2.2.4 SVM

On commence d'abord par initialiser les structures de données nécessaires, telles que les poids et les biais pour chaque classe, ainsi que le taux d'apprentissage qui contrôle la vitesse à laquelle l'algorithme apprend. Lors de l'entraînement, le SVM utilise les données d'entraînement pour ajuster les poids et les biais de manière à maximiser la marge entre les classes. Cela se fait en itérant sur les données d'entraînement, en calculant les scores pour chaque classe en utilisant le produit scalaire des poids et des caractéristiques d'entrée, puis en ajustant les poids en fonction des erreurs. La mise à jour des poids est guidée par la fonction de perte de marge, qui pénalise les classifications incorrectes et encourage une marge plus large entre les classes. Pour la prédiction on utilise les poids et les biais appris pour évaluer les scores de nouvelles données et prédire la classe correspondante. Le score le plus élevé indique la classe la plus probable pour l'exemple donné.

2.3 Conclusion

Dans ce chapitre, nous décrivons la logique derrière notre approche choisie. Dans le chapitre suivant, nous examinerons en détail les différents résultats que nous avons obtenus et discuterons des tests proposés.

Chapitre 3

Tests et Résultats

3.1 Introduction

Dans ce chapitre nous allons présenter les différents tests effectué ainsi que les divers résultats obtenus.

3.2 Tests et Résultats

Dans cette section nous allons varier les hyperparametres de chaque algorithme pour chaque type de représentation.

3.2.1 K-Nearest Neighbor

- E34

Les résultats obtenus pour la représentation E34 avec différentes configurations de l'algorithme KNN (k plus proches voisins) et la distance euclidienne ou Manhattan sont présentés dans plusieurs tableaux. Le modèle a été évalué en utilisant différentes métriques telles que la précision, le rappel, et le score F1 pour chaque classe, ainsi que des métriques macro-pondérées pour l'ensemble du modèle. Les résultats sont divisés en trois parties principales en fonction de la distance utilisée (Euclidienne ou Manhattan), du nombre de voisins (k) choisi et le ratio entre l'ensemble d'entrainement et de test.

Class	Micro Precision	Micro Recall	Micro F1-Score
1	1.0	1.0	1.0
2	0.4	0.4	0.4
3	0.8	0.8	0.8
4	1.0	1.0	1.0
5	0.5	0.5	0.5
6	1.0	1.0	1.0
7	0.6667	0.6667	0.6667
8	0.5	0.5	0.5
9	0	0	0

TABLE 3.1 – E34 - train : 60% - k = 1 (KNN) - Euclidean distance

1	2	0	0	0	0	1	0	0
0	4	0	0	0	0	0	0	0
0	0	4	0	0	0	0	0	0
0	0	0	4	0	0	0	0	0
0	0	0	0	4	0	0	0	0
0	1	0	0	0	3	0	0	0
0	2	0	0	0	0	2	0	0
0	0	0	0	3	0	0	1	0
0	1	1	0	1	0	0	1	0

TABLE 3.3 – Confusion Matrix - E34 - train : 60% - k = 1 (KNN) - Euclidean distance

Metric	**Value**
Macro-Averaged Precision	0.7333
Macro-Averaged Recall	0.7333
Macro-Averaged F1 Score	0.7333
Execution time	77 milliseconds

TABLE 3.2 – E34 - train : 60% - k = 1 (KNN) - Euclidean distance

Class	**Precision**	**Recall**	**F1-Score**
1	0.3333	0.3333	0.3333
2	0.4	0.4	0.4
3	0.8	0.8	0.8
4	1.0	1.0	1.0
5	0.5	0.5	0.5
6	0.75	0.75	0.75
7	1.0	1.0	1.0
8	0	0	0
9	0	0	0

TABLE 3.4 – E34 - train : 60% - k = 2 (KNN) - Class Metrics

Metric	**Value**
Macro-Averaged Precision	0.5979
Macro-Averaged Recall	0.5979
Macro-Averaged F1 Score	0.5979
Execution time	8 milliseconds

TABLE 3.5 – E34 - train : 60% - k = 2 (KNN) - Macro Metrics

1	3	0	0	0	0	0	0	0
0	4	0	0	0	0	0	0	0
0	0	4	0	0	0	0	0	0
0	0	0	4	0	0	0	0	0
0	0	0	0	4	0	0	0	0
0	1	0	0	0	3	0	0	0
1	1	0	0	0	0	2	0	0
1	0	0	0	3	0	0	0	0
0	1	1	0	1	1	0	0	0

TABLE 3.6 – Confusion Matrix - E34 - train : 60% - k = 2 (KNN)

Class	**Precision**	**Recall**	**F1-Score**
1	0.0	0.0	0
2	0.4	0.4	0.4
3	0.8	0.8	0.8
4	1.0	1.0	1.0
5	0.6	0.6	0.6
6	0.75	0.75	0.75
7	1.0	1.0	1.0
8	0.0	0.0	0
9	0	0	0

TABLE 3.7 – E34 - train : 60% - k = 3 (KNN) - Class Metrics

Metric	**Value**
Macro-Averaged Precision	0.65
Macro-Averaged Recall	0.65
Macro-Averaged F1 Score	0.65
Execution time	6 milliseconds

TABLE 3.8 – E34 - train : 60% - k = 3 (KNN) - Macro Metrics

0	4	0	0	0	0	0	0	0
0	4	0	0	0	0	0	0	0
0	0	4	0	0	0	0	0	0
0	0	0	4	0	0	0	0	0
0	0	0	0	3	0	0	1	0
0	1	0	0	0	3	0	0	0
1	1	0	0	0	0	2	0	0
3	0	0	0	1	0	0	0	0
1	0	1	0	1	1	0	0	0

TABLE 3.9 – Confusion Matrix - E34 - train : 60% - k = 3 (KNN)

Class	Precision	Recall	F1-Score
1	0.0	0.0	0.0
2	0.4444	0.4444	0.4444
3	0.8	0.8	0.8
4	1.0	1.0	1.0
5	0.8	0.8	0.8
6	1.0	1.0	1.0
7	1.0	1.0	1.0
8	1.0	1.0	1.0
9	0.0	0.0	0.0

TABLE 3.10 – E34 - train : 60% - k = 4 (KNN) - Class Metrics

Metric	Value
Macro-Averaged Precision	0.7556
Macro-Averaged Recall	0.7556
Macro-Averaged F1 Score	0.7556
Execution time	6 milliseconds

TABLE 3.11 – E34 - train : 60% - k = 4 (KNN) - Macro Metrics

0	3	0	0	0	0	0	0	1
0	4	0	0	0	0	0	0	0
0	0	4	0	0	0	0	0	0
0	0	0	4	0	0	0	0	0
0	0	0	0	4	0	0	0	0
0	1	0	0	0	3	0	0	0
1	1	0	0	0	0	2	0	0
3	0	0	0	0	0	0	1	0
2	0	1	0	1	0	0	0	0

TABLE 3.12 – Confusion Matrix - E34 - train : 60% - k = 4 (KNN)

Class	Precision	Recall	F1-Score
1	0.0	0.0	0.0
2	0.4286	0.4286	0.4286
3	0.75	0.75	0.75
4	1.0	1.0	1.0
5	1.0	1.0	1.0
6	1.0	1.0	1.0
7	1.0	1.0	1.0
8	0.6667	0.6667	0.6667
9	0.0	0.0	0.0

TABLE 3.13 – E34 - train : 70% - k = 4 (KNN) - Class Metrics

Metric	Value
Macro-Averaged Precision	0.7307
Macro-Averaged Recall	0.7307
Macro-Averaged F1 Score	0.7307
Execution time	6 milliseconds

TABLE 3.14 – E34 - train : 70% - k = 4 (KNN) - Macro Metrics

0	3	0	0	0	0	0	0	0
0	3	0	0	0	0	0	0	0
0	0	3	0	0	0	0	0	0
0	0	0	3	0	0	0	0	0
0	0	0	0	3	0	0	0	0
0	1	0	0	0	2	0	0	0
1	0	0	0	0	0	2	0	0
1	0	0	0	0	0	0	2	0
1	0	1	0	0	0	0	1	0

TABLE 3.15 – Confusion Matrix - E34 - train : 70% - k = 4 (KNN)

Class	Precision	Recall	F1-Score
1	0.0	0.0	0.0
2	0.5	0.5	0.5
3	1.0	1.0	1.0
4	1.0	1.0	1.0
5	0.6667	0.6667	0.6667
6	1.0	1.0	1.0
7	1.0	1.0	1.0
8	1.0	1.0	1.0
9	0.0	0.0	0.0

TABLE 3.16 – E34 - train : 80% - k = 4 (KNN) - Class Metrics

Metric	Value
Macro-Averaged Precision	0.7708
Macro-Averaged Recall	0.7708
Macro-Averaged F1 Score	0.7708
Execution time	4 milliseconds

TABLE 3.17 – E34 - train : 80% - k = 4 (KNN) - Macro Metrics

0	2	0	0	0	0	0	0	0
0	2	0	0	0	0	0	0	0
0	0	2	0	0	0	0	0	0
0	0	0	2	0	0	0	0	0
0	0	0	0	2	0	0	0	0
0	0	0	0	0	2	0	0	0
1	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	1	0
1	0	0	0	1	0	0	0	0

TABLE 3.18 – Confusion Matrix - E34 - train : 80% - k = 4 (KNN)

Nous avons gardé les memes hyperparametres mais nous avons changé la métrique de distance.

Class	**Precision**	**Recall**	**F1-Score**
1	0.0	0.0	0.0
2	0.4	0.4	0.4
3	1.0	1.0	1.0
4	1.0	1.0	1.0
5	1.0	1.0	1.0
6	1.0	1.0	1.0
7	1.0	1.0	1.0
8	0.5	0.5	0.5
9	0.0	0.0	0.0

TABLE 3.19 – E34 - train : 80% - k = 4 (KNN) - Class Metrics 'Manhattan distance'

Metric	**Value**
Macro-Averaged Precision	0.7375
Macro-Averaged Recall	0.7375
Macro-Averaged F1 Score	0.7375
Execution time	4 milliseconds

TABLE 3.20 – E34 - train : 80% - k = 4 (KNN) - Macro Metrics 'Manhattan distance'

0	2	0	0	0	0	0	0	0
0	2	0	0	0	0	0	0	0
0	0	2	0	0	0	0	0	0
0	0	0	2	0	0	0	0	0
0	0	0	0	2	0	0	0	0
0	1	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	1	0

TABLE 3.21 – Confusion Matrix - E34 - train : 80% - k = 4 (KNN) 'Manhattan distance'

Suite à la modification de la distance utilisée,On remarque que la distance de Manhattan a considérablement amélioré les performances de notre modèle. Ainsi, après un processus de fine-tuning, nous avons abouti à la configuration optimale suivante :

Class	**Precision**	**Recall**	**F1-Score**
1	0.5	0.5	0.5
2	0.6	0.6	0.6
3	0.75	0.75	0.75
4	1.0	1.0	1.0
5	0.6	0.6	0.6
6	1.0	1.0	1.0
7	1.0	1.0	1.0
8	1.0	1.0	1.0
9	0.0	0.0	0.0

TABLE 3.22 – E34 - train : 70% - k = 2 (KNN) - Class Metrics - 'Manhattan distance'

Metric	**Value**
Macro-Averaged Precision	0.80625
Macro-Averaged Recall	0.80625
Macro-Averaged F1 Score	0.80625
Execution time	6 milliseconds

TABLE 3.23 – E34 - train : 70% - k = 2 (KNN) - Macro Metrics- 'Manhattan distance'

2	1	0	0	0	0	0	0	0
0	3	0	0	0	0	0	0	0
0	0	3	0	0	0	0	0	0
0	0	0	3	0	0	0	0	0
0	0	0	0	3	0	0	0	0
0	0	0	0	0	2	0	0	1
1	0	0	0	0	0	2	0	0
1	0	0	0	1	0	0	1	0
0	1	1	0	1	0	0	0	0

TABLE 3.24 – Confusion Matrix - E34 - train : 70% - k = 2 (KNN)- 'Manhattan distance'

Parameter	**Value**
Train percentage	70%
Test percentage	30%
k	2
Distance	Manhattan distance

TABLE 3.25 – Meilleures paramètre du modele de la représentation E34

On remarque également que, pour cette représentation, toutes les configurations du modèle ont produit d'excellents résultats pour la plupart des classes, à l'exception des classes 01 et 09. La meilleure configuration a réussi à identifier plus ou moins les échantillons appartenant à la classe 01. Cependant, aucune amélioration significative n'a été observée pour la classe 09, pour laquelle les métriques demeurent nulles.

- GFD

Comme le montrent les résultats présentés dans les tableaux, le modèle affiche des performances parfaites pour toutes les configurations.

Class	**Precision**	**Recall**	**F1-Score**
1	1.0	1.0	1.0
2	1.0	1.0	1.0
3	1.0	1.0	1.0
4	1.0	1.0	1.0
5	1.0	1.0	1.0
6	1.0	1.0	1.0
7	1.0	1.0	1.0
8	1.0	1.0	1.0
9	1.0	1.0	1.0

TABLE 3.26 – GFD - train : 60% - k = 1 (KNN) - Class Metrics

Metric	Value
Macro-Averaged Precision	1.0
Macro-Averaged Recall	1.0
Macro-Averaged F1 Score	1.0
Execution time	80 milliseconds

TABLE 3.27 – GFD - train : 60% - k = 1 (KNN) - Macro Metrics

4	0	0	0	0	0	0	0	0
0	4	0	0	0	0	0	0	0
0	0	4	0	0	0	0	0	0
0	0	0	4	0	0	0	0	0
0	0	0	0	4	0	0	0	0
0	0	0	0	0	4	0	0	0
0	0	0	0	0	0	4	0	0
0	0	0	0	0	0	0	4	0
0	0	0	0	0	0	0	0	4

TABLE 3.28 – Confusion Matrix - GFD - train : 60% - k = 1 (KNN)

Class	Precision	Recall	F1-Score
1	1.0	1.0	1.0
2	1.0	1.0	1.0
3	1.0	1.0	1.0
4	1.0	1.0	1.0
5	1.0	1.0	1.0
6	1.0	1.0	1.0
7	1.0	1.0	1.0
8	1.0	1.0	1.0
9	1.0	1.0	1.0

TABLE 3.29 – GFD - train : 60% - k = 2 (KNN) - Class Metrics

Metric	Value
Macro-Averaged Precision	1.0
Macro-Averaged Recall	1.0
Macro-Averaged F1 Score	1.0
Execution time	18 milliseconds

TABLE 3.30 – GFD - train : 60% - k = 2 (KNN) - Macro Metrics

4	0	0	0	0	0	0	0	0
0	4	0	0	0	0	0	0	0
0	0	4	0	0	0	0	0	0
0	0	0	4	0	0	0	0	0
0	0	0	0	4	0	0	0	0
0	0	0	0	0	4	0	0	0
0	0	0	0	0	0	4	0	0
0	0	0	0	0	0	0	4	0
0	0	0	0	0	0	0	0	4

TABLE 3.31 – Confusion Matrix - GFD - train : 60% - k = 2 (KNN)

Class	**Precision**	**Recall**	**F1-Score**
1	1.0	1.0	1.0
2	1.0	1.0	1.0
3	1.0	1.0	1.0
4	1.0	1.0	1.0
5	1.0	1.0	1.0
6	1.0	1.0	1.0
7	1.0	1.0	1.0
8	1.0	1.0	1.0
9	1.0	1.0	1.0

TABLE 3.32 – GFD - train : 60% - k = 3 (KNN) - Class Metrics

Metric	**Value**
Macro-Averaged Precision	1.0
Macro-Averaged Recall	1.0
Macro-Averaged F1 Score	1.0
Execution time	17 milliseconds

TABLE 3.33 – GFD - train : 60% - k = 3 (KNN) - Macro Metrics

4	0	0	0	0	0	0	0	0
0	4	0	0	0	0	0	0	0
0	0	4	0	0	0	0	0	0
0	0	0	4	0	0	0	0	0
0	0	0	0	4	0	0	0	0
0	0	0	0	0	4	0	0	0
0	0	0	0	0	0	4	0	0
0	0	0	0	0	0	0	4	0
0	0	0	0	0	0	0	0	4

TABLE 3.34 – Confusion Matrix - GFD - train : 60% - k = 3 (KNN)

Class	Precision	Recall	F1-Score
1	1.0	1.0	1.0
2	1.0	1.0	1.0
3	1.0	1.0	1.0
4	1.0	1.0	1.0
5	1.0	1.0	1.0
6	1.0	1.0	1.0
7	1.0	1.0	1.0
8	1.0	1.0	1.0
9	1.0	1.0	1.0

TABLE 3.35 – GFD - train : 60% - k = 4 (KNN) - Class Metrics

Metric	Value
Macro-Averaged Precision	1.0
Macro-Averaged Recall	1.0
Macro-Averaged F1 Score	1.0
Execution time	17 milliseconds

TABLE 3.36 – GFD - train : 60% - k = 4 (KNN) - Macro Metrics

4	0	0	0	0	0	0	0	0
0	4	0	0	0	0	0	0	0
0	0	4	0	0	0	0	0	0
0	0	0	4	0	0	0	0	0
0	0	0	0	4	0	0	0	0
0	0	0	0	0	4	0	0	0
0	0	0	0	0	0	4	0	0
0	0	0	0	0	0	0	4	0
0	0	0	0	0	0	0	0	4

TABLE 3.37 – Confusion Matrix - GFD - train : 60% - k = 4 (KNN)

Lorsque nous fixons k à 4, les résultats restent constants pour les autres tests, même en faisant varier le pourcentage d'entraînement. La seule variation observée concerne le temps d'exécution de quelques microsecondes. En revanche, avec k égal à 1, le modèle a parfois tendance à commettre des erreurs, comme le montrent les tableaux suivants. Cependant, pour le reste des scénarios, les prédictions demeurent sans fautes.

Class	Precision	Recall	F1-Score
1	1.0	1.0	1.0
2	1.0	1.0	1.0
3	1.0	1.0	1.0
4	1.0	1.0	1.0
5	1.0	1.0	1.0
6	1.0	1.0	1.0
7	1.0	1.0	1.0
8	0.75	0.75	0.75
9	1.0	1.0	1.0

TABLE 3.38 – GFD - train : 70% - k = 1 (KNN) - Class Metrics

Metric	Value
Macro-Averaged Precision	0.9722
Macro-Averaged Recall	0.9722
Macro-Averaged F1 Score	0.9722
Execution time	27 milliseconds

TABLE 3.39 – GFD - train : 70% - k = 1 (KNN) - Macro Metrics

3	0	0	0	0	0	0	0	0
0	3	0	0	0	0	0	0	0
0	0	2	0	0	0	0	1	0
0	0	0	3	0	0	0	0	0
0	0	0	0	3	0	0	0	0
0	0	0	0	0	3	0	0	0
0	0	0	0	0	0	3	0	0
0	0	0	0	0	0	0	3	0
0	0	0	0	0	0	0	0	3

TABLE 3.40 – Confusion Matrix - GFD - train : 70% - k = 1 (KNN)

En ce qui concerne la distance de Manhattan, les résultats sont plus ou moins similaires, mais il est à noter que le modèle a tendance à commettre davantage d'erreurs lorsque k est égal à 1 par rapport à la distance euclidienne, comme illustré dans ce qui suit :

Class	Precision	Recall	F1-Score
1	1.0	1.0	1.0
2	1.0	1.0	1.0
3	1.0	1.0	1.0
4	1.0	1.0	1.0
5	1.0	1.0	1.0
6	1.0	1.0	1.0
7	1.0	1.0	1.0
8	0.8	0.8	0.8
9	1.0	1.0	1.0

TABLE 3.41 – GFD - train : 60% - k = 1 (KNN) - Class Metrics

Metric	Value
Macro-Averaged Precision	0.9778
Macro-Averaged Recall	0.9778
Macro-Averaged F1 Score	0.9778
Execution time	80 milliseconds

TABLE 3.42 – GFD - train : 60% - k = 1 (KNN) - Macro Metrics

4	0	0	0	0	0	0	0	0
0	4	0	0	0	0	0	0	0
0	0	3	0	0	0	0	1	0
0	0	0	4	0	0	0	0	0
0	0	0	0	4	0	0	0	0
0	0	0	0	0	4	0	0	0
0	0	0	0	0	0	4	0	0
0	0	0	0	0	0	0	4	0
0	0	0	0	0	0	0	0	4

TABLE 3.43 – Confusion Matrix - GFD - train : 60% - k = 1 (KNN)

- F0

Pour les autres représentations, nous avons conservé uniquement la meilleure configuration après avoir ajusté les hyperparamètres, comme dans les cas de GFD et E34.

Class	Precision	Recall	F1-Score
1	0.666	0.666	0.666
2	1.0	1.0	1.0
3	0.5714	0.5714	0.5714
4	1.0	1.0	1.0
5	0.333	0.333	0.333
6	0.5	0.5	0.5
7	0.5	0.5	0.5
8	0.75	0.75	0.75
9	1.0	1.0	1.0

TABLE 3.44 – F0 - train : 60% - k = 4 (KNN) - Class Metrics

Metric	Value
Macro-Averaged Precision	0.7023809523809523
Macro-Averaged Recall	0.7023809523809523
Macro-Averaged F1 Score	0.7023809523809523
Execution time	32 milliseconds

TABLE 3.45 – F0 - train : 60% - k = 4 (KNN) - Macro Metrics

2	0	0	0	0	1	0	1	0
0	4	0	0	0	0	0	0	0
0	0	4	0	0	0	0	0	0
1	0	0	3	0	0	0	0	0
0	0	3	0	1	0	0	0	0
0	0	0	0	1	3	0	0	0
0	0	0	0	1	2	1	0	0
0	0	0	0	0	0	1	3	0
0	0	0	0	0	0	0	0	4

TABLE 3.46 – Confusion Matrix - F0 - train : 60% - k = 4 (KNN)

Parameter	Value
Train percentage	60%
Test percentage	40%
k	4
Distance	Euclidean distance

TABLE 3.47 – Meilleures paramètre du modele de la représentation F0

On constate que le modèle présente des résultats satisfaisants, il parvient à distinguer certaines classes tout en éprouvant des difficultés avec d'autres (05 et 07). Cependant, il est important de souligner que le meilleur modèle obtenu demeure inférieur à celui de GFD et E34.

- SA En ce qui concerne cette représentation, le meilleur modèle obtenu surpasse celui de F0 et est légèrement inférieur à celui d'E34. Toutefois, il est important de noter que le modèle GFD demeure le plus performant.

Class	**Precision**	**Recall**	**F1-Score**
1	0.25	0.25	0.25
2	1.0	1.0	1.0
3	0.8	0.8	0.8
4	1.0	1.0	1.0
5	0.364	0.364	0.364
6	0.8	0.8	0.8
7	1.0	1.0	1.0
8	1.0	1.0	1.0
9	1.0	1.0	1.0

TABLE 3.48 – SA - train : 60% - k = 1 (KNN) - Class Metrics

Metric	**Value**
Macro-Averaged Precision	0.8015151515151515
Macro-Averaged Recall	0.8015151515151515
Macro-Averaged F1 Score	0.8015151515151515
Execution time	85 milliseconds

TABLE 3.49 – SA - train : 60% - k = 1 (KNN) - Macro Metrics

1	0	0	0	2	1	0	0	0
1	2	0	0	1	0	0	0	0
0	0	4	0	0	0	0	0	0
0	0	0	4	0	0	0	0	0
0	0	0	0	4	0	0	0	0
0	0	0	0	0	4	0	0	0
2	0	0	0	1	0	1	0	0
0	0	0	0	3	0	0	1	0
0	0	1	0	0	0	0	0	3

TABLE 3.50 – Confusion Matrix - SA - train : 60% - k = 1 (KNN)

Parameter	**Value**
Train percentage	60%
Test percentage	40%
k	1
Distance	Manhattan distance

TABLE 3.51 – Meilleures paramètre du modele de la représentation F0

3.2.2 SVM

Pour SVM, nous avons varié le bias, le pourcentage du jeu de données(test et entrainement) ainsi que le nombre d'epoques. **- E34**
Les résultats présentés dans les tableaux révèlent un mauvais ajustement du modèle pour les classes 01, 03 et 04, caractérisé par des métriques nulles. En revanche, la classe 02 affiche des performances parfaites. Dans l'ensemble, les résultats globaux ne sont pas très satisfaisants.

Class	**Precision**	**Recall**	**F1-Score**
1	0.0	0.0	0.0
2	1.0	1.0	1.0
3	0.0	0.0	0.0
4	0.0	0.0	0.0
5	0.5	1.0	0.667
6	1.0	1.0	1.0
7	0.5	0.25	0.333
8	1.0	0.333	0.5
9	1.0	0.667	0.8

TABLE 3.52 – E34 - train : 80% - Bias = 0.01 - Nb epoch 10000 (SVM) - Class Metrics

Metric	**Value**
Macro-Averaged Precision	0.556
Macro-Averaged Recall	0.472
Macro-Averaged F1 Score	0.478
Execution time	1019 milliseconds

TABLE 3.53 – E34 - train : 80% - Bias = 0.01 - Nb epoch 10000 (SVM) - Macro Metrics

0	0	0	0	0	0	2	0	0
0	2	0	0	0	0	0	0	0
0	0	0	0	0	0	0	2	0
0	0	0	0	0	0	1	1	0
0	0	0	0	1	0	0	1	0
0	0	0	0	0	2	0	0	0
0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	2	0
0	0	0	0	0	0	0	0	2

TABLE 3.54 – Confusion Matrix - E34 - train : 80% - Bias = 0.01 - Nb epoch 10000 (SVM)

Parameter	Value
Train percentage	80%
Test percentage	20%
Bias	0.01
Number of epochs	10000

TABLE 3.55 – Meilleures paramètre du modele de la représentation E34

- GFD

Pour ce type de représentation, le modèle montre une excellente performance pour toutes les classes, avec des valeurs parfaites pour les métriques.Egalement la performance globale est élevée avec un temps d'exécution relativement court.

Class	Precision	Recall	F1-Score
1	0.667	1.0	0.8
2	1.0	1.0	1.0
3	1.0	1.0	1.0
4	1.0	0.75	0.857
5	1.0	1.0	1.0
6	1.0	1.0	1.0
7	1.0	1.0	1.0
8	1.0	1.0	1.0
9	1.0	1.0	1.0

TABLE 3.56 – GFD - train : 70% - Bias = 0.01 - Nb epoch 1000 (SVM) - Class Metrics

Metric	Value
Macro-Averaged Precision	0.963
Macro-Averaged Recall	0.972
Macro-Averaged F1 Score	0.962
Execution time	216 milliseconds

TABLE 3.57 – GFD - train : 70% - Bias = 0.01 - Nb epoch 1000 (SVM) - Macro Metrics

2	0	0	1	0	0	0	0	0
0	3	0	0	0	0	0	0	0
0	0	3	0	0	0	0	0	0
0	0	0	3	0	0	0	0	0
0	0	0	0	3	0	0	0	0
0	0	0	0	0	3	0	0	0
0	0	0	0	0	0	3	0	0
0	0	0	0	0	0	0	3	0
0	0	0	0	0	0	0	0	3

TABLE 3.58 – Confusion Matrix - GFD - train : 70% - Bias = 0.01 - Nb epoch 1000 (SVM)

Parameter	Value
Train percentage	70%
Test percentage	30%
Bias	0.01
Number of epochs	1000

TABLE 3.59 – Meilleures paramètre du modele de la représentation GFD

- F0

Les performances du modèle dans cette représentation spécifique sont médiocres, avec des métriques nulles ou proches de zéro pour chaque classe et de manière globale.

Class	Precision	Recall	F1-Score
1	0.000	0.000	0.000
2	0.000	0.000	0.000
3	1.000	0.500	0.667
4	1.000	0.571	0.727
5	0.000	0.000	0.000
6	0.000	0.000	0.000
7	0.000	0.000	0.000
8	1.000	0.211	0.348
9	0.250	1.000	0.400

TABLE 3.60 – F0 - train : 60% - bias = 0.01 - Nb epoch 10000 (SVM) - Class Metrics

Metric	Value
Macro-Averaged Precision	0.361
Macro-Averaged Recall	0.254
Macro-Averaged F1 Score	0.238
Execution time	1330 milliseconds

TABLE 3.61 – F0 - train : 60% - bias = 0.01 - Nb epoch 10000 (SVM) - Macro Metrics

0	0	2	1	0	0	0	1	0
0	0	0	2	0	0	0	2	0
0	0	4	0	0	0	0	0	0
0	0	0	4	0	0	0	0	0
0	0	1	0	0	0	0	3	0
0	0	0	0	0	0	0	4	0
0	0	0	0	0	0	0	4	0
0	0	0	0	0	0	0	4	0
1	0	1	0	0	0	0	1	1

TABLE 3.62 – Confusion Matrix - F0 - train : 60% - bias = 0.01 - Nb epoch 10000 (SVM)

Parameter	Value
Train percentage	60%
Test percentage	40%
Bias	0.01
Number of epochs	10000

TABLE 3.63 – Meilleures paramètre du modele de la représentation F0

- SA

Pour la représentation SA, les performances varient d'une classe à l'autre, avec certaines classes bien classifiées et d'autres mal classifiées. Dans l'ensemble la performance du modèle est moyenne, avec un temps d'exécution plus élevé par rapport aux autres modèles.

Class	Precision	Recall	F1-Score
1	0.000	0.000	0.000
2	0.750	0.750	0.750
3	1.000	0.667	0.800
4	1.000	1.000	1.000
5	0.000	0.000	0.000
6	1.000	1.000	1.000
7	0.000	0.000	0.000
8	1.000	0.308	0.471
9	0.000	0.000	0.000

TABLE 3.64 – SA - train : 60% - bias = 0.1 - Nb epoch 10000 (SVM) - Class Metrics

Metric	Value
Macro-Averaged Precision	0.528
Macro-Averaged Recall	0.414
Macro-Averaged F1 Score	0.447
Execution time	2419 milliseconds

TABLE 3.65 – SA - train : 60% - bias = 0.1 - Nb epoch 10000 (SVM) - Macro Metrics

0	1	1	0	0	0	1	0	1
0	3	0	0	0	0	0	0	1
0	0	4	0	0	0	0	0	0
0	0	0	4	0	0	0	0	0
0	0	0	0	0	0	0	4	0
0	0	0	0	0	4	0	0	0
0	0	0	0	1	0	0	3	0
0	0	0	0	0	0	0	4	0
1	0	1	0	0	0	0	2	0

TABLE 3.66 – Confusion Matrix - SA - train : 60% - bias = 0.1 - Nb epoch 10000 (SVM)

Parameter	Value
Train percentage	60%
Test percentage	40%
Bias	0.1
Number of epochs	10000

TABLE 3.67 – Meilleures paramètre du modele de la représentation SA

3.2.3 K-means

Pour l'évaluation de Kmeans nos critères, sont en fonction du nombre d'itérations, du type de distance, des composantes principales (ACP). Les résultats sont présentés dans les tableaux ci-dessus.
K-means n'est pas un algorithme de classification. Dans le but de l'adapter à notre problème et pour utiliser les memes métriques d'évaluation que les autres modèles, nous avons expérimenté une approche qui implique d'évaluer la contribution de chaque classe aux clusters et d'assigner à chaque cluster la classe correspondante, de sorte que tous les éléments de ce cluster appartiennent à cette classe. Cependant, dans cette démarche nous avons rencontré des difficultés, comme lorsque deux clusters partagent la même classe ayant la contribution la plus élevée, ce qui pourrait entraîner des ambiguïtés. En conséquence, nous avons décidé de maintenir l'approche classique de K-means tout en l'évaluant à l'aide du Silhouette Score.

— Le score de silhouette est une mesure utilisée pour évaluer la qualité d'une technique de regroupement. Il mesure à quel point un objet est similaire à son propre groupe par rapport à d'autres groupes. La valeur du score de silhouette varie de -1 à 1. Une valeur proche de 1 indique que l'objet est bien regroupé, une valeur proche de 0 indique des groupes qui se chevauchent, et une valeur proche de -1 indique que l'objet a peut-être été attribué au mauvais groupe.

La formule pour calculer le score de silhouette pour un point de données donné est :

$$s = \frac{b - a}{\max(a, b)}$$

où : a est la distance moyenne intra-cluster, c'est-à-dire la distance moyenne entre le point de données et tous les autres points de données dans le même groupe, b est la distance moyenne inter-cluster, c'est-à-dire la distance moyenne entre le point de données et tous les autres points de données dans le groupe le plus proche.

- SA

Les taux de données dans les 2 et 3 premières composantes principales sont de 52.26% et 66.11%, respectivement. Globalement, les métriques de Silhouette Score augmentent avec le nombre d'itérations, indiquant une amélioration de la séparation des clusters. comme le montre le tableau suivant la meilleure configuration est obtenues avec une distance de Manhattan et 2 composantes principales, présentant un Silhouette Score de 0.390 et un temps d'exécution de 8 ms.

Nb Iterations	Distance Type	Components (PCA)	Kmeans Results
100	Euclidean	None	Silhouette Score : 0.080 Execution Time : 630 ms
1000	Euclidean	None	Silhouette Score : 0.102 Execution Time : 4193 ms
10000	Euclidean	None	Silhouette Score : 0.110 Execution Time : 23895 ms
100	Manhattan	None	Silhouette Score : 0.167 Execution Time : 275 ms
1000	Manhattan	None	Silhouette Score : 0.108 Execution Time : 2175 ms
10000	Manhattan	None	Silhouette Score : 0.197 Execution Time : 24200 ms
100	Euclidean	3	Silhouette Score : 0.338 Execution Time : 66 ms
1000	Euclidean	3	Silhouette Score : 0.322 Execution Time : 346 ms
10000	Euclidean	3	Silhouette Score : 0.298 Execution Time : 11038 ms
100	Manhattan	3	Silhouette Score : 0.280 Execution Time : 18 ms
1000	Manhattan	3	Silhouette Score : 0.298 Execution Time : 123 ms
10000	Manhattan	3	Silhouette Score : 0.340 Execution Time : 4142 ms
100	Euclidean	2	Silhouette Score : 0.346 Execution Time : 56 ms
1000	Euclidean	2	Silhouette Score : 0.376 Execution Time : 308 ms
10000	Euclidean	2	Silhouette Score : 0.373 Execution Time : 10415 ms
100	**Manhattan**	**2**	**Silhouette Score : 0.390** **Execution Time : 8 ms**
1000	Manhattan	2	Silhouette Score : 0.373 Execution Time : 97 ms
10000	Manhattan	2	Silhouette Score : 0.310 Execution Time : 3456 ms

- F0

Taux de données dans les 2 premières composantes principales : 97.11% .
Taux de données dans les 3 premières composantes principales : 99.39% .
La classe F0 présente des Silhouette Scores plus élevés, indiquant une meilleure séparation des clusters. Les meilleures performances sont observées avec une distance de Manhattan, aucune réduction de dimension (None), et 10000 itérations, avec un Silhouette Score de 0.458.

Nb Iterations	Distance Type	Components (ACP)	Kmeans Results
100	Euclidean	None	Silhouette Score : 0.443 Execution Time : 900 ms
1000	Euclidean	None	Silhouette Score : 0.406 Execution Time : 4950 ms
10000	Euclidean	None	Silhouette Score : 0.399 Execution Time : 32305 ms
100	Manhattan	None	Silhouette Score : 0.409 Execution Time : 375 ms
1000	Manhattan	None	Silhouette Score : 0.368 Execution Time : 3065 ms
10000	**Manhattan**	**None**	**Silhouette Score : 0.458** **Execution Time : 32775 ms**
100	Euclidean	3	Silhouette Score : 0.425 Execution Time : 63 ms
1000	Euclidean	3	Silhouette Score : 0.349 Execution Time : 338 ms
10000	Euclidean	3	Silhouette Score : 0.426 Execution Time : 10697 ms
100	Manhattan	3	Silhouette Score : 0.406 Execution Time : 16 ms
1000	Manhattan	3	Silhouette Score : 0.420 Execution Time : 117 ms
10000	Manhattan	3	Silhouette Score : 0.420 Execution Time : 3797 ms
100	Euclidean	2	Silhouette Score : 0.336 Execution Time : 49 ms
1000	Euclidean	2	Silhouette Score : 0.370 Execution Time : 341 ms
10000	Euclidean	2	Silhouette Score : 0.419 Execution Time : 10541 ms
100	Manhattan	2	Silhouette Score : 0.430 Execution Time : 11 ms
1000	Manhattan	2	Silhouette Score : 0.425 Execution Time : 102 ms
10000	Manhattan	2	Silhouette Score : 0.437 Execution Time : 3742 ms

- E34

Taux de données dans les 2 premières composantes principales : 99.72% .
Taux de données dans les 3 premières composantes principales : 99.90% .
Les résultats montrent une amélioration des Silhouette Scores avec le nombre d'itérations. La meilleure configuration est obtenues avec une distance de Manhattan, 2 composantes principales, et 10000 itérations, affichant un Silhouette Score de 0.570.

Nb Iterations	Distance Type	Components (ACP)	Kmeans Results
100	Euclidean	None	Silhouette Score : 0.397 Execution Time : 134 ms
1000	Euclidean	None	Silhouette Score : 0.387 Execution Time : 1105 ms
10000	Euclidean	None	Silhouette Score : 0.413 Execution Time : 14975 ms
100	Manhattan	None	Silhouette Score : 0.272 Execution Time : 58 ms
1000	Manhattan	None	Silhouette Score : 0.370 Execution Time : 436 ms
10000	Manhattan	None	Silhouette Score : 0.370 Execution Time : 7167 ms
100	Euclidean	3	Silhouette Score : 0.532 Execution Time : 55 ms
1000	Euclidean	3	Silhouette Score : 0.476 Execution Time : 338 ms
10000	Euclidean	3	Silhouette Score : 0.455 Execution Time : 10703 ms
100	Manhattan	3	Silhouette Score : 0.539 Execution Time : 17 ms
1000	Manhattan	3	Silhouette Score : 0.471 Execution Time : 120 ms
10000	Manhattan	3	Silhouette Score : 0.541 Execution Time : 3822 ms
100	Euclidean	2	Silhouette Score : 0.558 Execution Time : 70 ms
1000	Euclidean	2	Silhouette Score : 0.517 Execution Time : 316 ms
10000	Euclidean	2	Silhouette Score : 0.489 Execution Time : 10521 ms
100	Manhattan	2	Silhouette Score : 0.532 Execution Time : 14 ms
1000	Manhattan	2	Silhouette Score : 0.521 Execution Time : 120 ms
10000	**Manhattan**	**2**	**Silhouette Score : 0.570** **Execution Time : 4535 ms**

- GFD

Taux de données dans les 2 premières composantes principales : 71.83% .
Taux de données dans les 3 premières composantes principales : 80.67% .
La classe GFD présente des Silhouette Scores modérés, mais les résultats sont améliorés avec le nombre d'itérations. Les performances optimales sont obtenues avec une distance euclidienne, 2 composantes principales, et 10000 itérations, affichant un Silhouette Score de 0.431.

Nb Iterations	Distance Type	Components (ACP)	Kmeans Results
100	Euclidean	None	Silhouette Score : 0.369 Execution Time : 719 ms
1000	Euclidean	None	Silhouette Score : 0.383 Execution Time : 4382 ms
10000	Euclidean	None	Silhouette Score : 0.323 Execution Time : 25421 ms
100	Manhattan	None	Silhouette Score : 0.219 Execution Time : 308 ms
1000	Manhattan	None	Silhouette Score : 0.378 Execution Time : 2419 ms
10000	Manhattan	None	Silhouette Score : 0.291 Execution Time : 26370 ms
100	Euclidean	3	Silhouette Score : 0.351 Execution Time : 61 ms
1000	Euclidean	3	Silhouette Score : 0.327 Execution Time : 334 ms
10000	Euclidean	3	Silhouette Score : 0.311 Execution Time : 10723 ms
100	Manhattan	3	Silhouette Score : 0.256 Execution Time : 16 ms
1000	Manhattan	3	Silhouette Score : 0.255 Execution Time : 117 ms
10000	Manhattan	3	Silhouette Score : 0.340 Execution Time : 3575 ms
100	Euclidean	2	Silhouette Score : 0.347 Execution Time : 41 ms
1000	Euclidean	2	Silhouette Score : 0.336 Execution Time : 299 ms
10000	**Euclidean**	2	**Silhouette Score : 0.431** **Execution Time : 10326 ms**
100	Manhattan	**2**	Silhouette Score : 0.4306 Execution Time : 9 ms
1000	Manhattan	2	Silhouette Score : 0.392 Execution Time : 95 ms
10000	Manhattan	2	Silhouette Score : 0.382 Execution Time : 3335 ms

3.3 Synthèse

Les résultats montrent que le modèle GFD se démarque avec les meilleures performances parmi tous les modèles, fournissant des résultats presque parfaits. Cela suggère que la représentation GFD a réussi à capturer toutes les caractéristiques importantes pour différencier entre les classes.
Pour améliorer notre étude, il serait utile de tester le modèle sur un ensemble de données plus

large pour confirmer notre hypothèse. De plus, utiliser l'analyse en composantes principales (ACP) pour visualiser nos regroupements ou classifications pourrait également enrichir notre compréhension des données.

3.4 Conclusion

En résumé, ce chapitre nous a permis d'examiner les résultats obtenus et d'en extraire des informations pertinentes. Nous avons également abordé les perspectives futures de notre étude.