

Machine learning in python

Réalisé par:

Elsa Catteau

Wissal Elfiguigui

Charlotte Prouzet

SOMMAIRE

1. Transformation des données
2. Classification des données
3. Réseau de neurones
4. Conclusion

I. Transformation des données

Utilisation du dataset digits de scikit-learn contenant :

- digits.data (images)
- digits.target (étiquettes de 0 à 9)

Chaque image correspond à un chiffre manuscrit représenté en 8x8 pixels

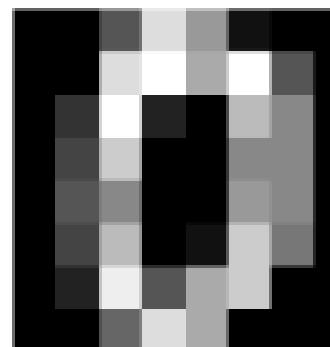
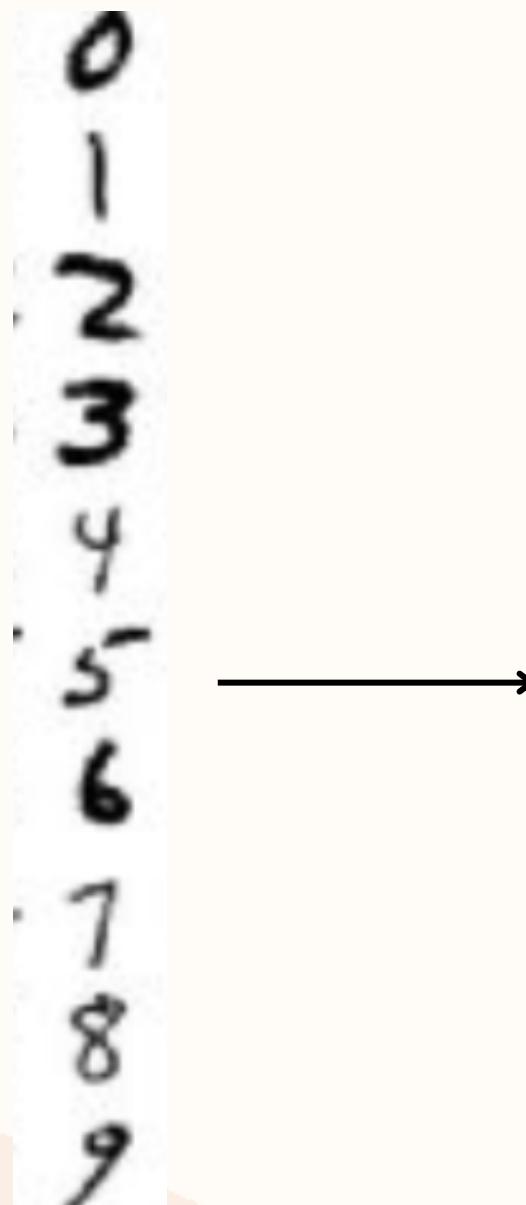


Figure 1: Visualisation de la 1ère image de notre data base : 0

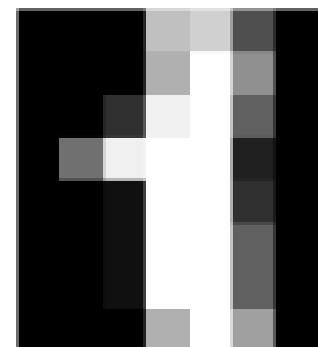


Figure 2: Visualisation de la 2ème image de notre data base : 1

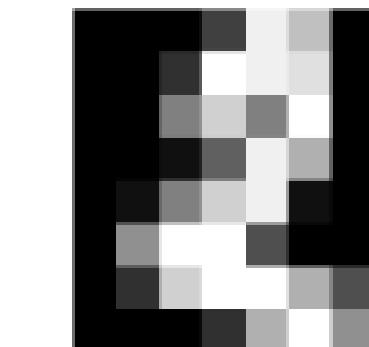


Figure 3: Visualisation de la 3ème image de notre data base : 2

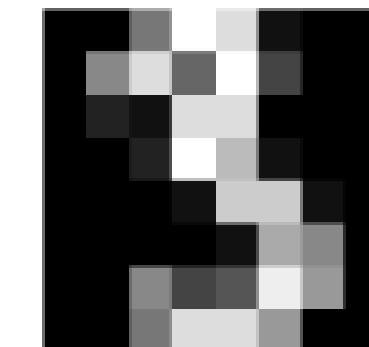
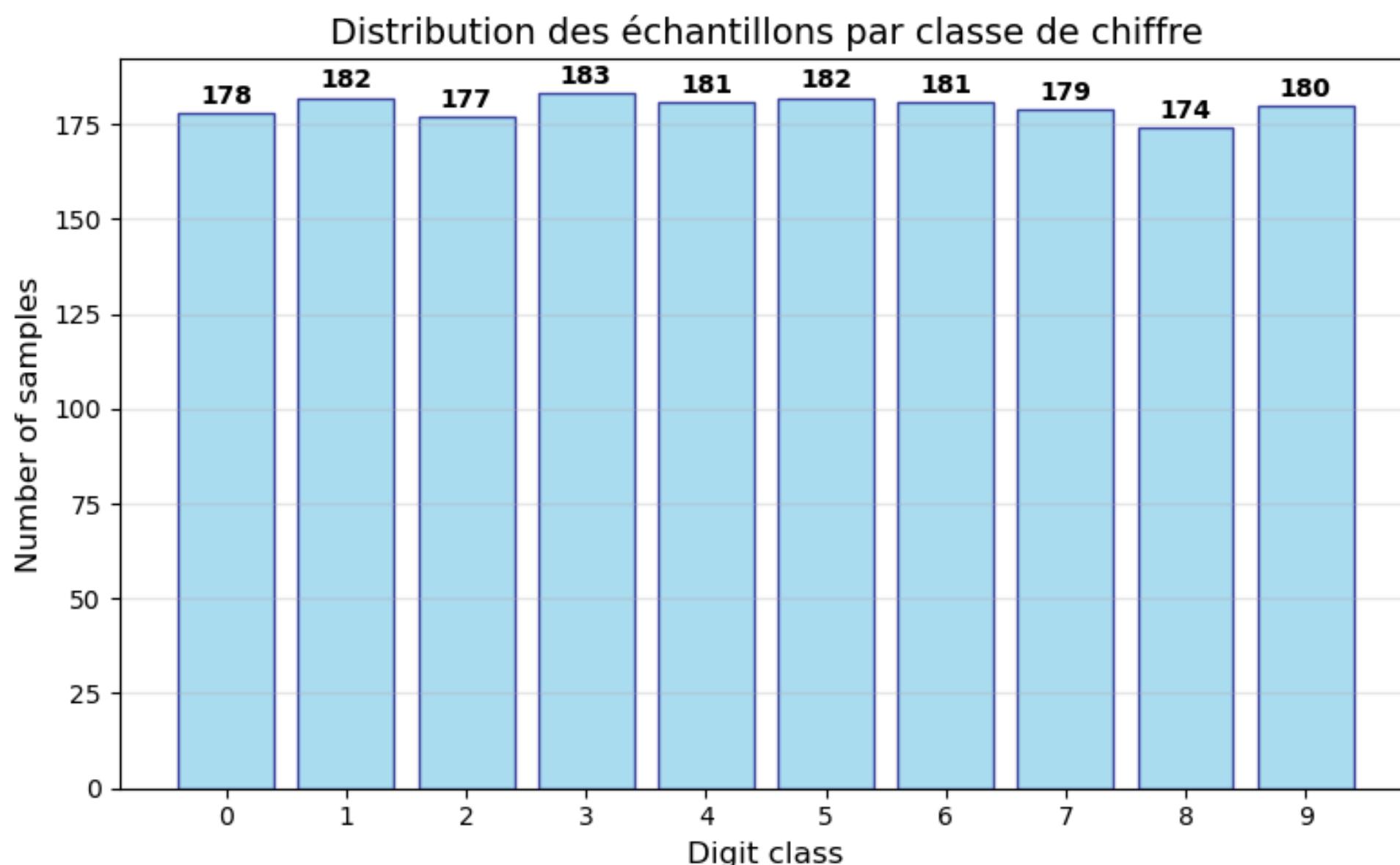


Figure 4: Visualisation de la 4ème image de notre data base 3

Analyse de la distribution



- › Un dataset équilibré pour la classification
- › fonction `get_statistics_text` pour analyser la fréquence des chiffres dans la base.
- › Le dataset présente une répartition homogène entre 174 et 183 échantillons par chiffre (0-9).
 - › Cet équilibre facilite l'entraînement des modèles en évitant les biais de classification

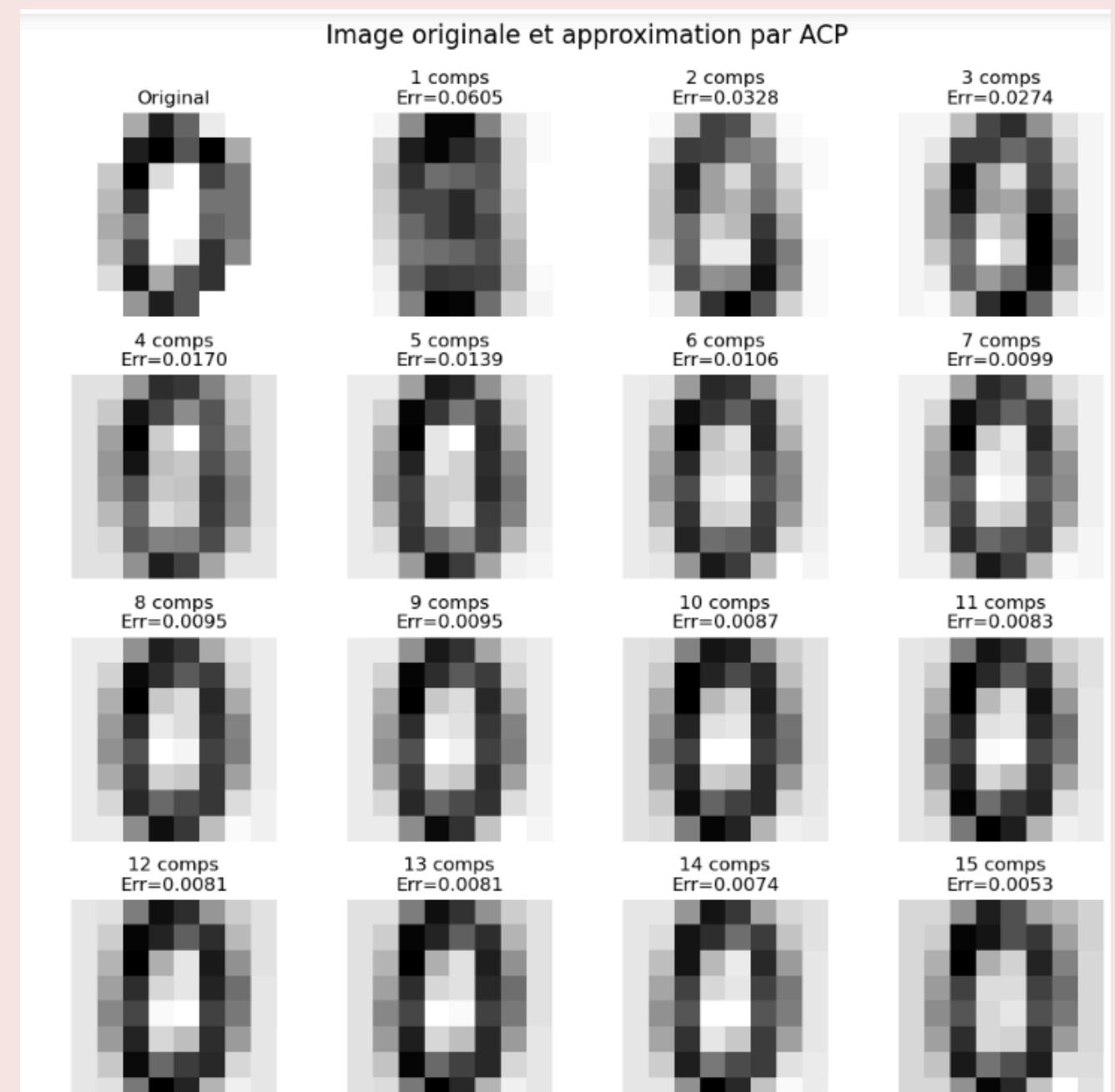
Réduction de dimension avec une ACP pour la visualisation

Définition :

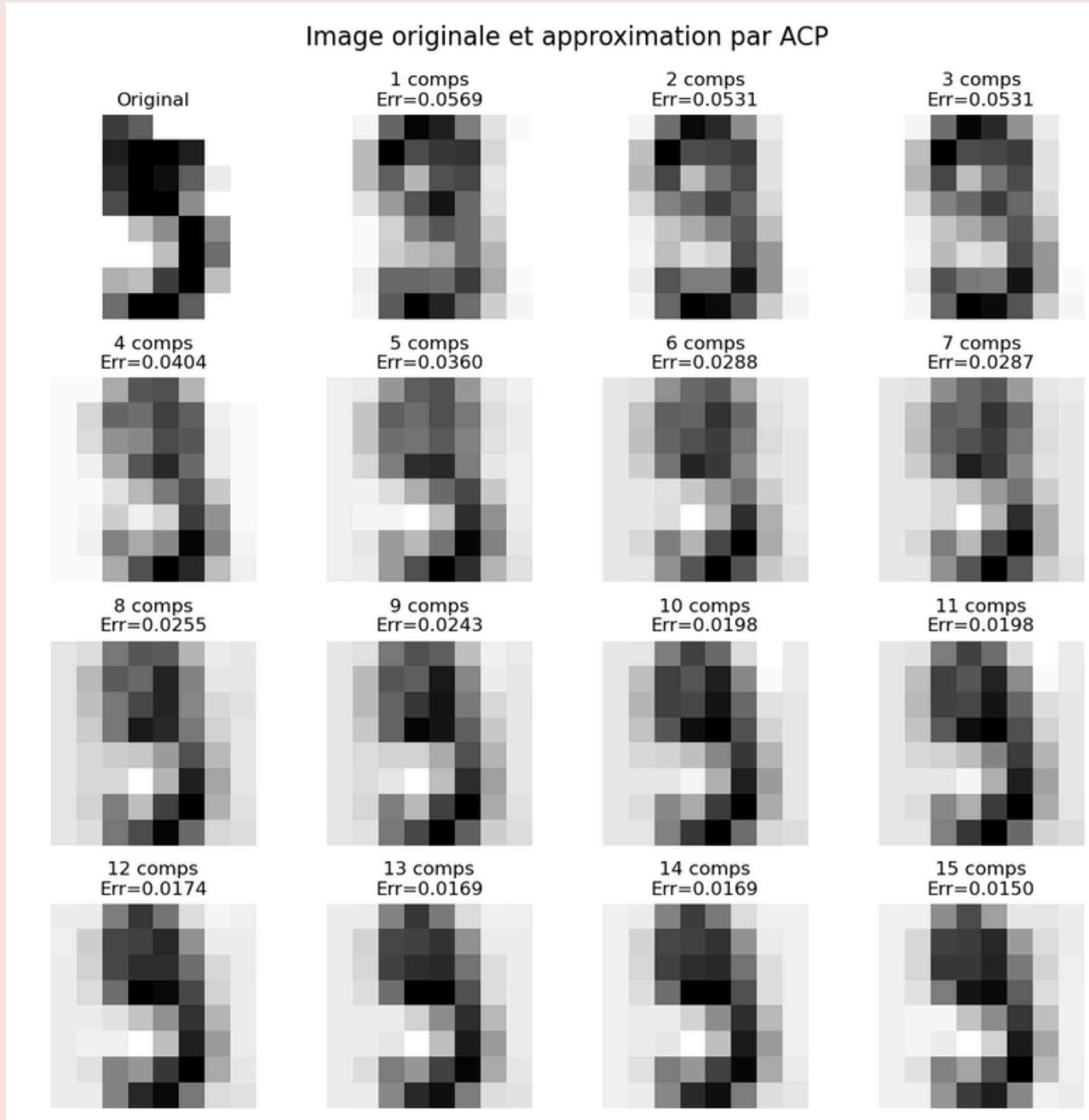
L'ACP réduit la dimension des données en les projetant sur des composantes orthogonales classées par variance expliquée, afin de minimiser l'erreur de reconstruction.

Observations :

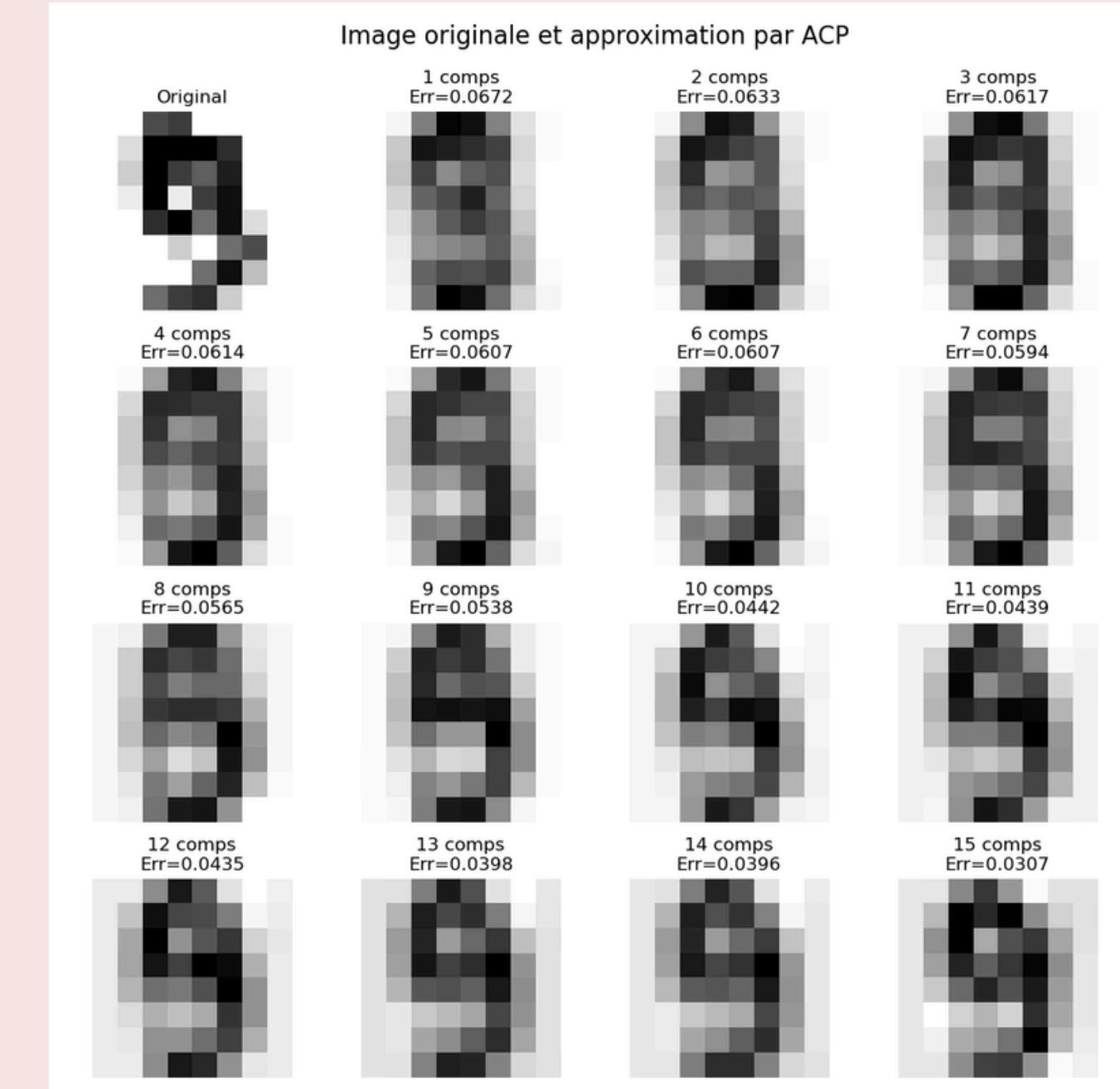
- > À vue d'œil, à partir de 10 composantes principales, la représentation devient satisfaisante
- > Ajouter plus de composantes n'améliore que très peu la visualisation.



Visualisation du chiffre 5

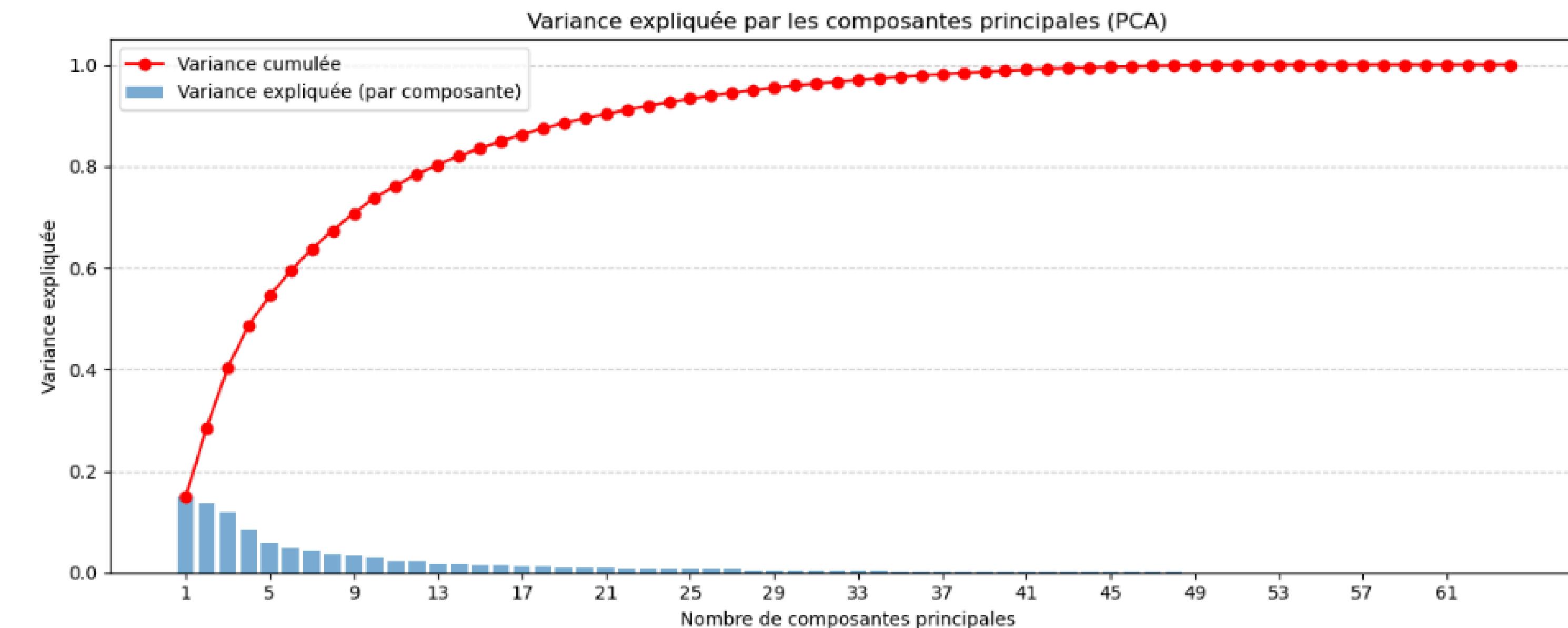


Visualisation du chiffre 9



- > Les chiffres 5 et 9 sont souvent mal distingués dans l'espace projeté.
 - > Même avec un nombre élevé de composantes, leurs points se mélangent, ce qui rend leur classification difficile visuellement.
- => La PCA aide à compresser et visualiser les données, mais elle ne préserve pas toujours la séparabilité entre toutes les classes !

Variance Expliquée et Nombre Optimal de Composantes

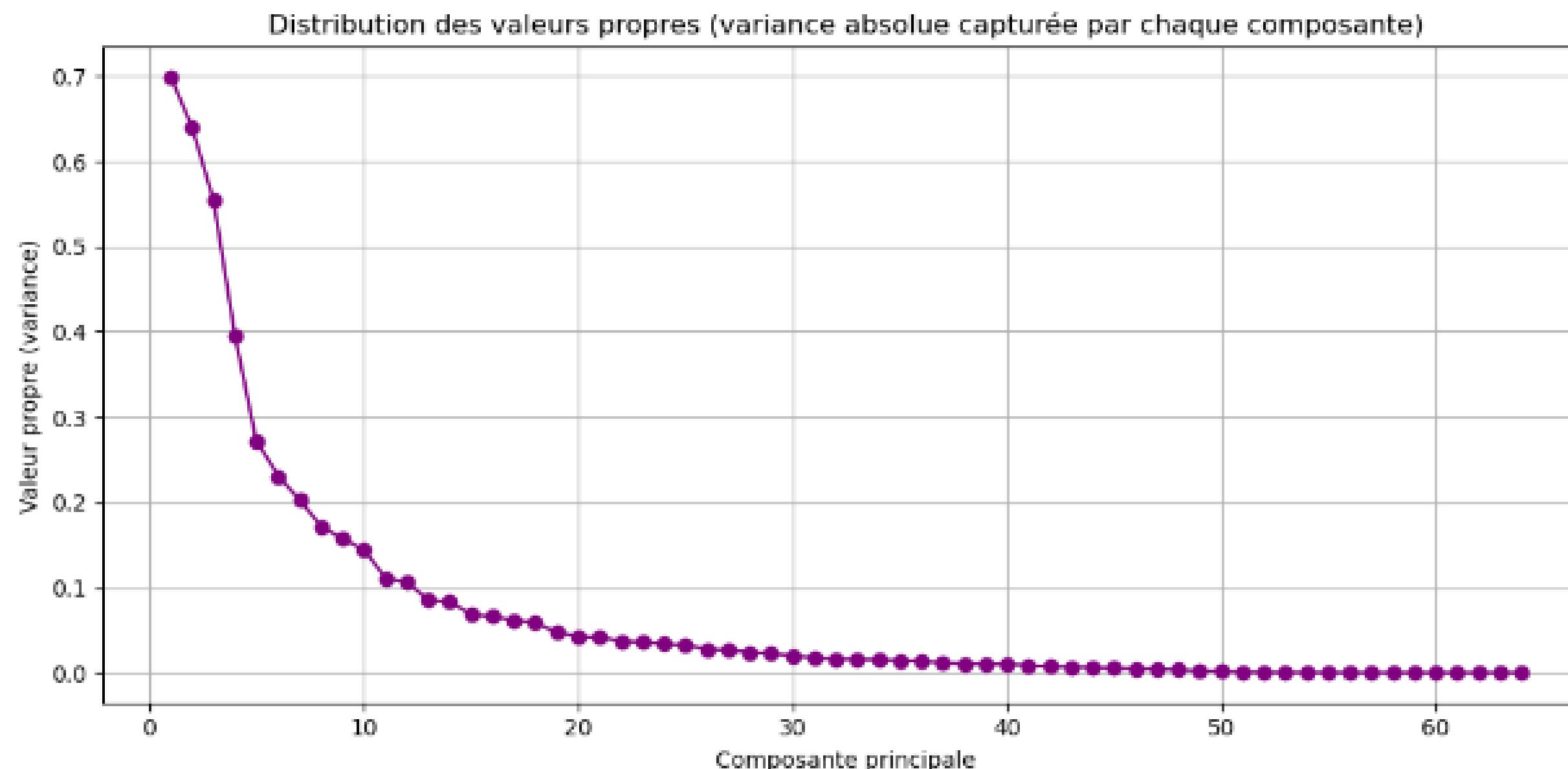


> Résultat obtenu : **29 composantes** suffisent pour expliquer au moins 95 % de la variance.

Analyse de la distribution des valeurs propres

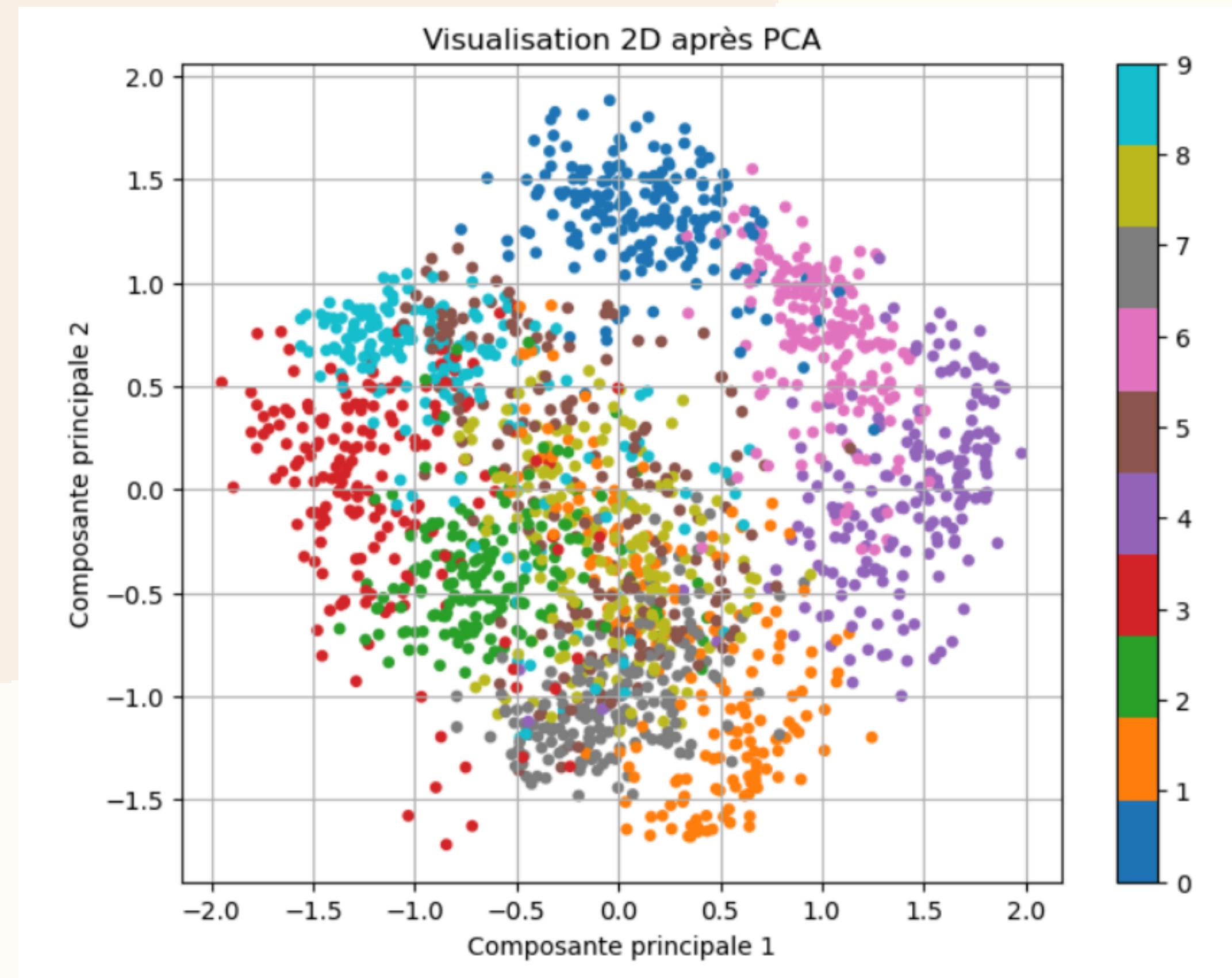
Pour approfondir l'analyse, on examine la distribution des valeurs propres.

> Cela revient à étudier la variance expliquée brute par chaque composante principale.



- Forte décroissance des valeurs propres au début
=> les premières composantes capturent l'essentiel de la variance.
- Les dernières composantes ont des valeurs proches de zéro, donc peu informatives.

Application de la PCA en 2 dimensions



Extraction des caractéristiques

> Réduction de la matrice à 20 dimension

1. Extraction en appliquant la PCA > **F_pca shape : (1797, 20)**

2. Extraction par zone : On découpe la matrice en 3 zones > **F_zones shape : (1797, 3)**

3. Extraction basée sur la détection des contours : Filtre de Sobel > **F_edges shape : (1797, 1)**

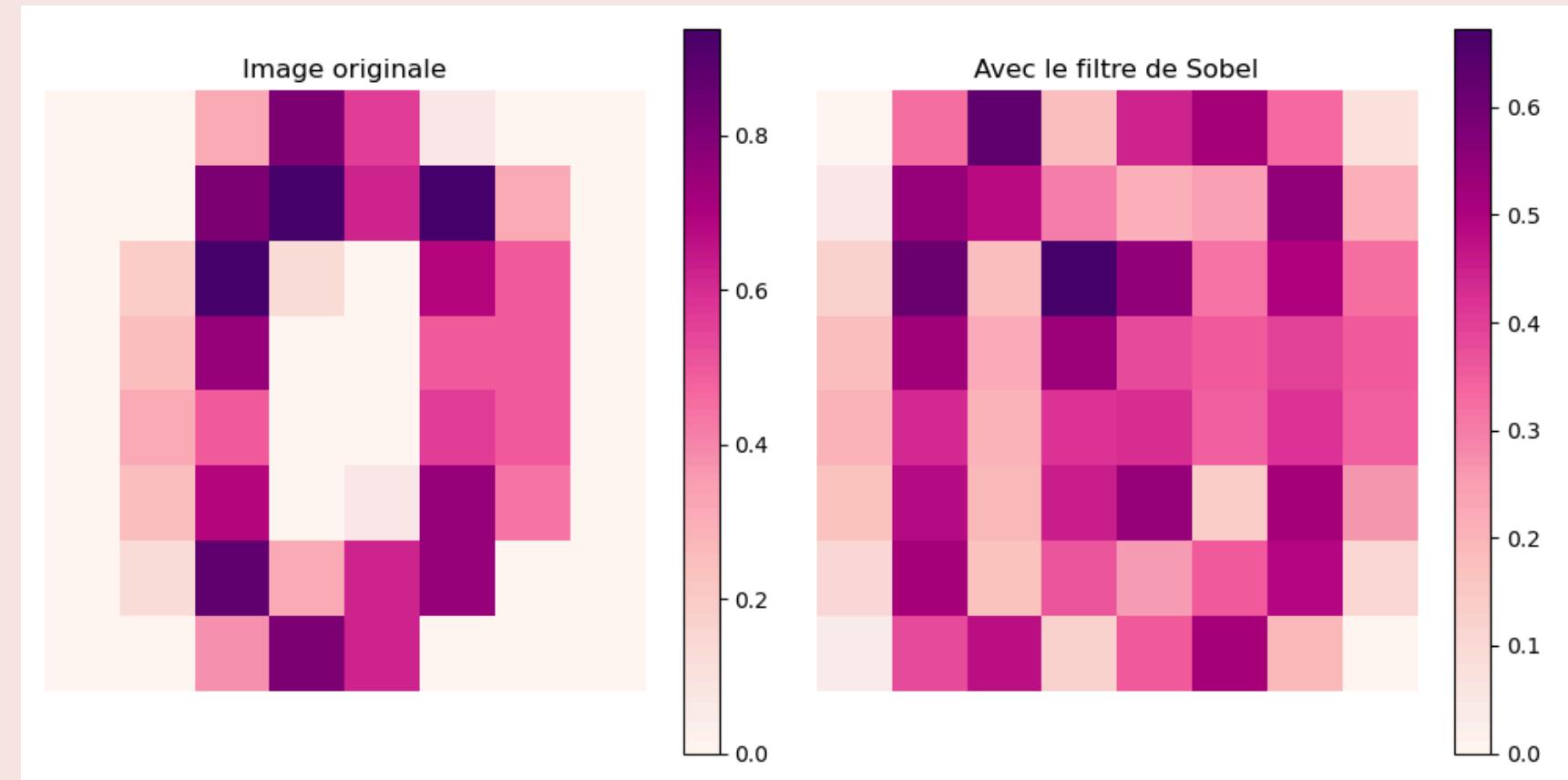


Image originale du chiffre 0 VS Image avec le filtre de Sobel

> Combinaison des 3 caractéristiques

> Normalisation

>> Résultat : **F_final shape : (1797, 24)**

II. Classification des données

Graphique en barres représentant la répartition des classes

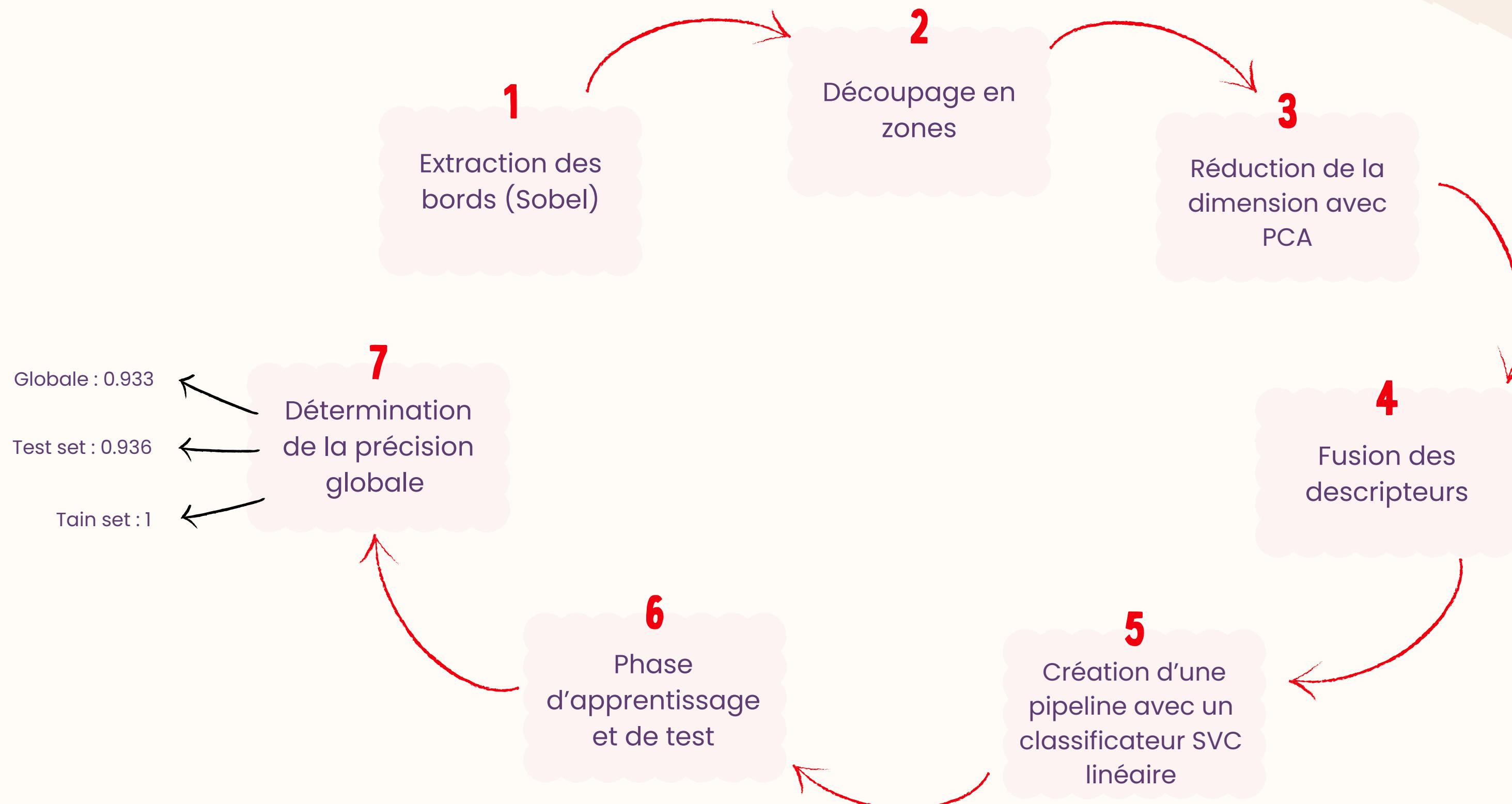


Visualisation de la distribution normalisée des classes



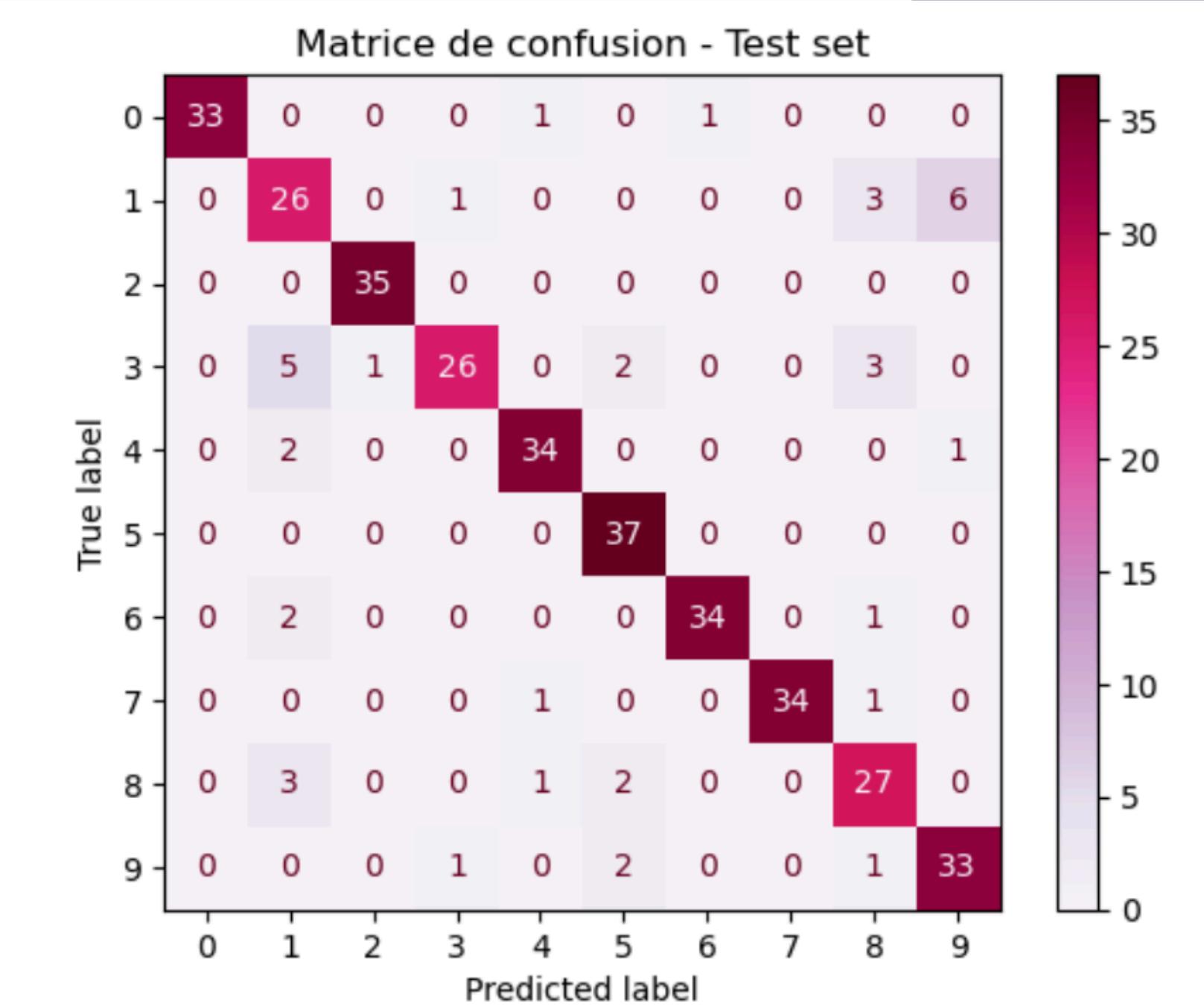
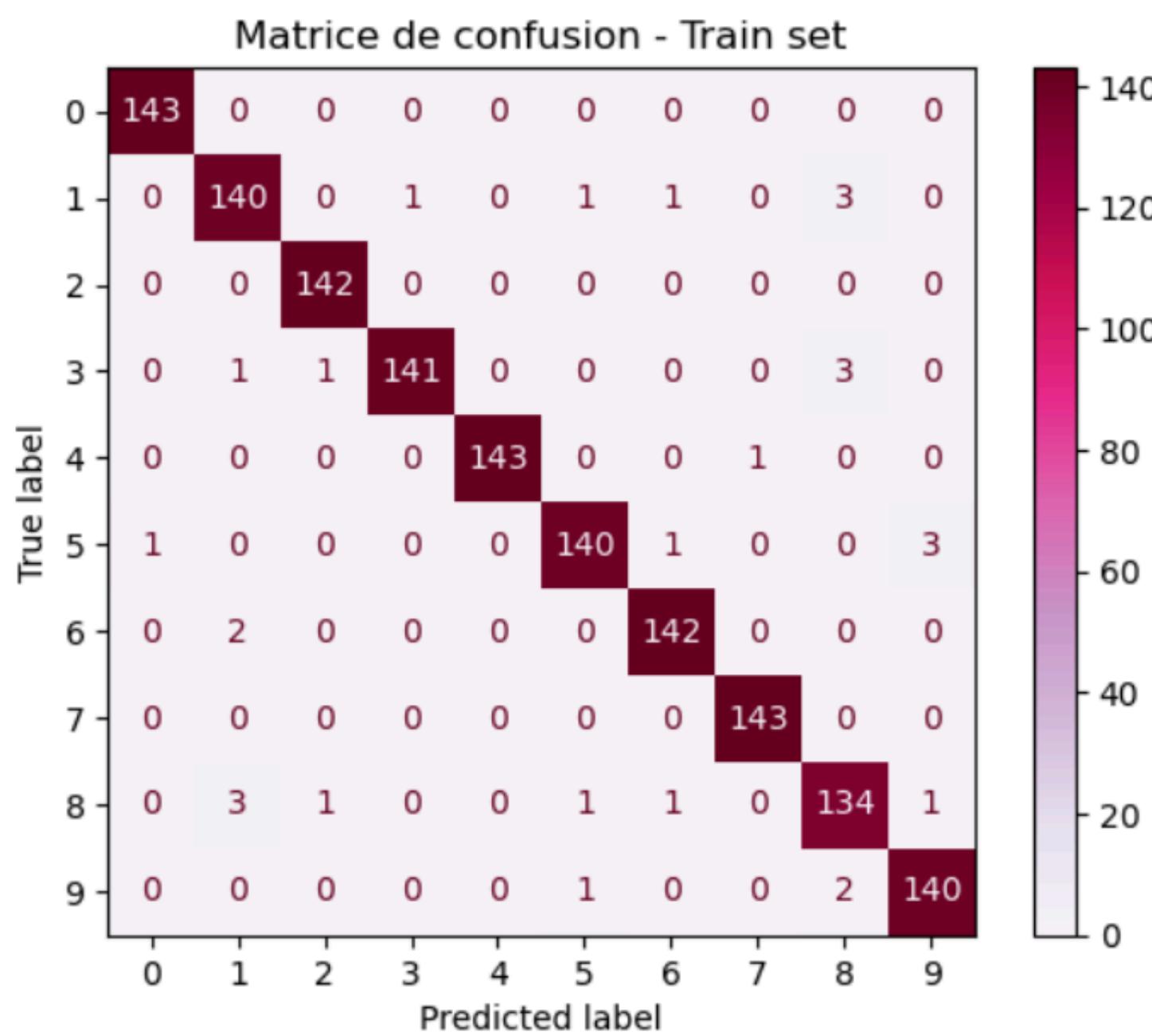
Méthodes de classification avec SVM

Objectif : Classifier les chiffres manuscrits en combinant plusieurs descripteurs dans une seule pipeline:

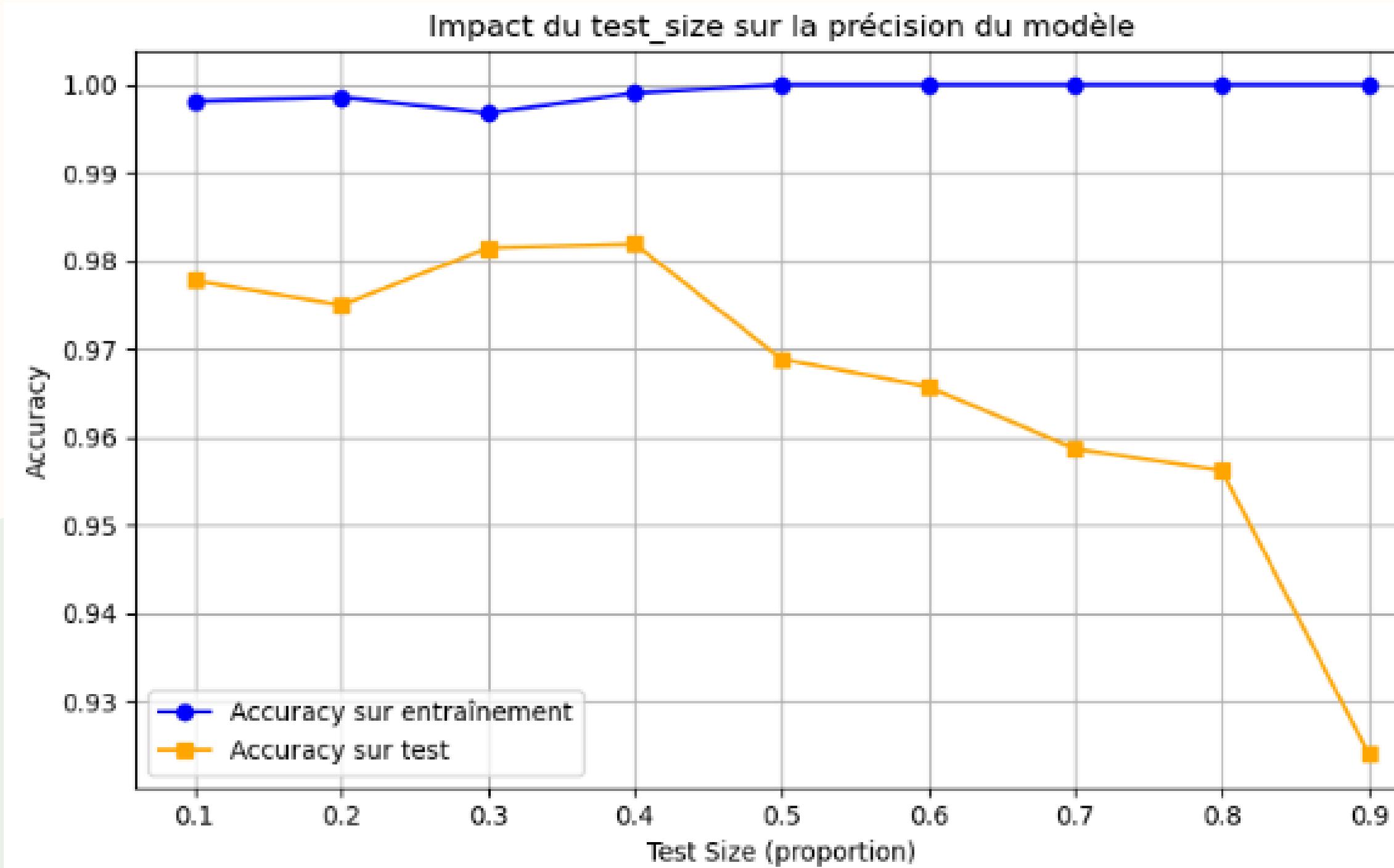


Premier entraînement d'un SVC

Matrice de confusion



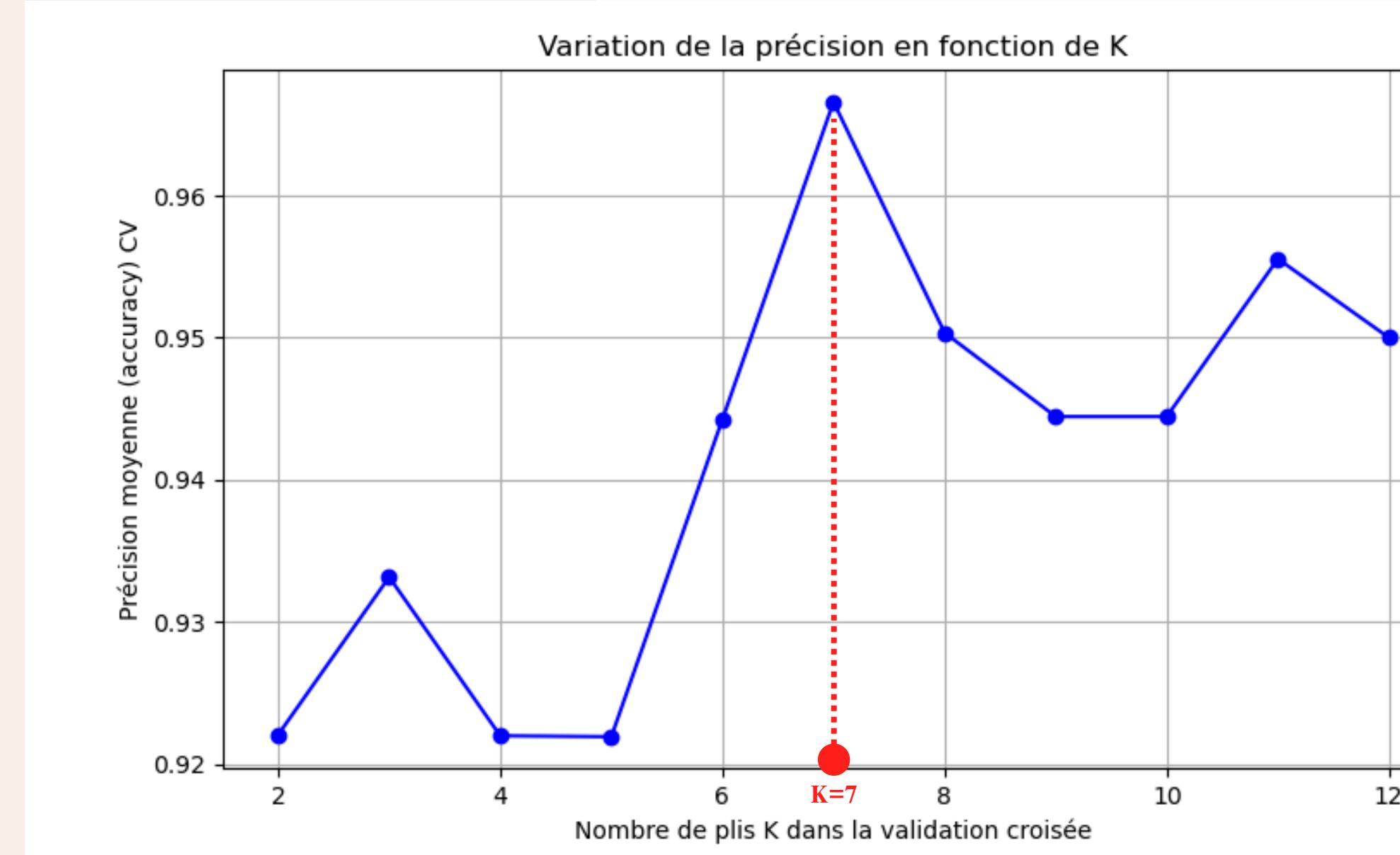
L'influence de la taille vis à vis de la précision de notre modèle



Test : 40% / Entrainement : 60%

Hyperparamètres et validation croisée (CV)

- C : coefficient de régularisation du SVM
- gamma : paramètre du noyau RBF
- K : nombre de plis dans la validation croisée



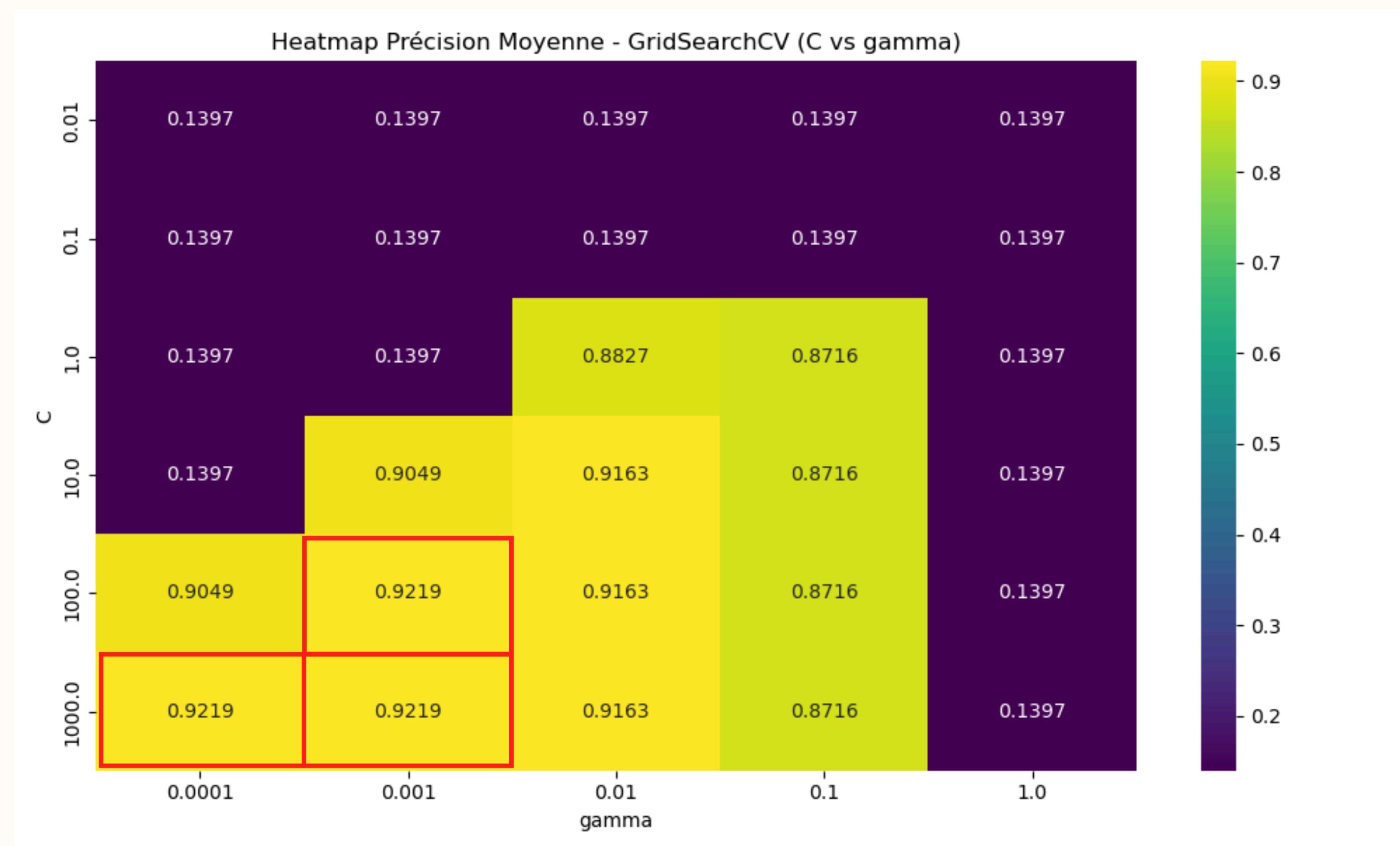
K=5

- Précision : 92.19%
- C=100
- Gamma = 0.001

K=10

- Précision : 94.44%
- C=100
- Gamma = 0.001

Choix des hyperparamètres C et gamma



C=100 et gamma=0.001

One-vs-One vs One-vs-Rest

OvO (One-vs-One) :

- Consiste à entraîner un classifieur pour chaque paire de classes. Si on a K classes, alors on entraîne $K(K-1)/2$ classifieurs.
- Recommandé pour les petits datasets

OvR (One-vs-Rest) :

- Consiste à entraîner un classifieur par classe, en distinguant chaque classe contre toutes les autres.
- Recommandé pour les grands volumes de données
- Moins coûteux en calcul donc plus rapide

Résultats OvO :

- Précision : 0.943
- Temps d'exécution : 0.65 secondes

Résultats OvR :

- Précision : 0.947
- Temps d'exécution : 0.14 secondes

III. Réseau de neurones

Objectif :

- Construire, entraîner et évaluer un réseau de neurones capable de reconnaître des chiffres manuscrits (de 0 à 9) à partir d'images en niveaux de gris.

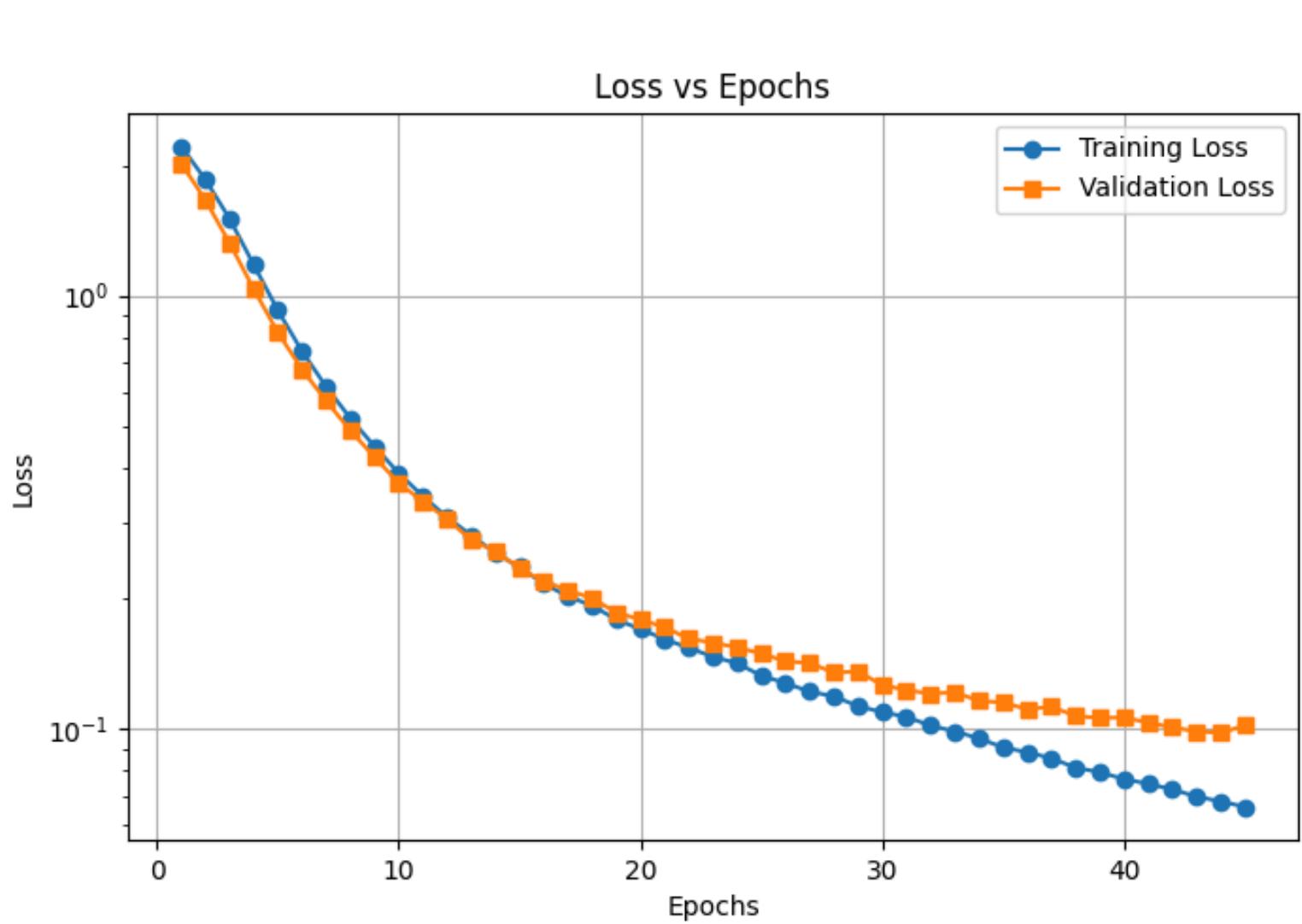
Modèle utilisé :

- une couche d'entrée correspondant aux 64 pixels de l'image,
- une couche cachée avec 32 neurones
- une couche de sortie avec 10 neurones (un par chiffre)

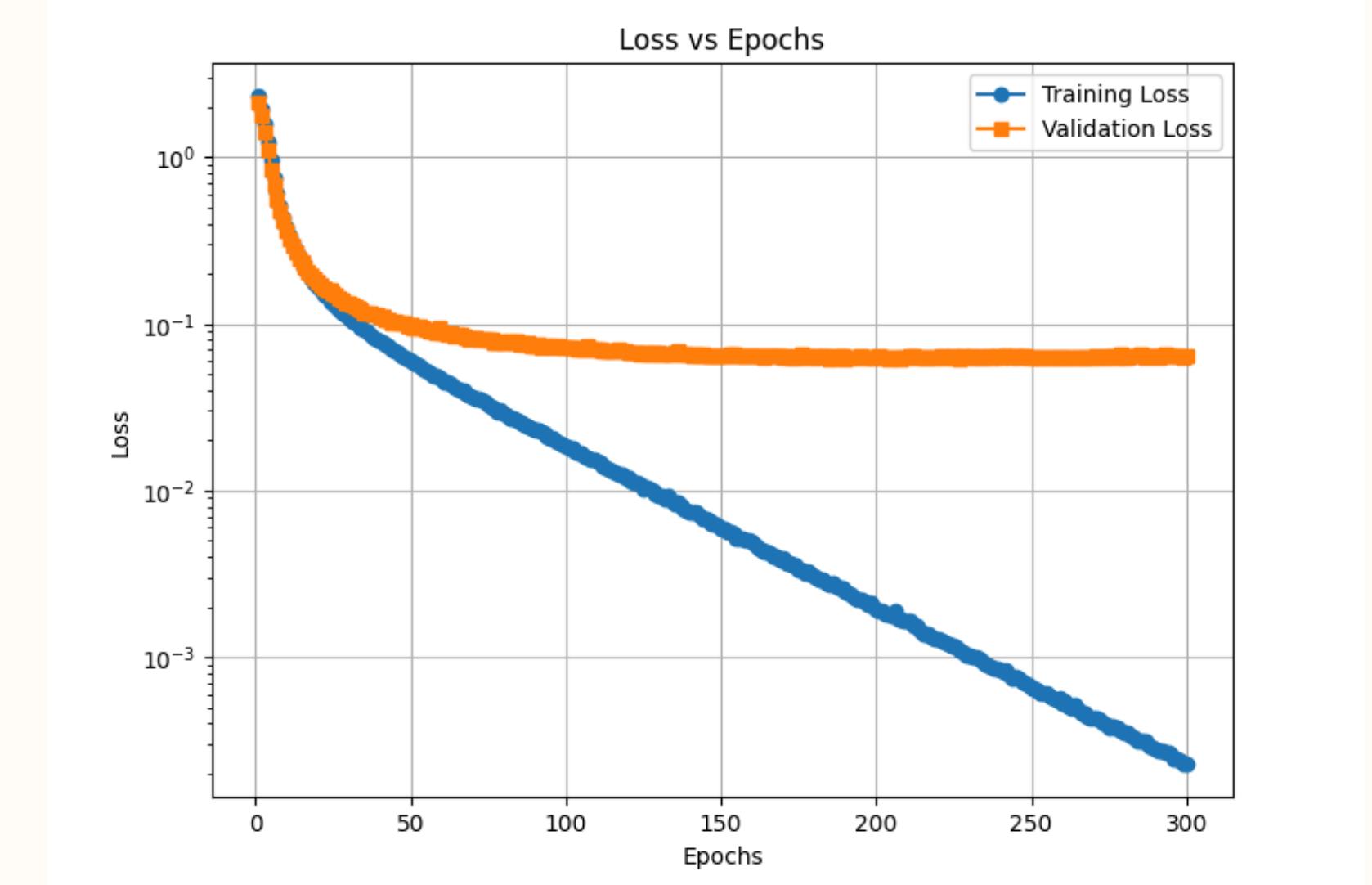
Compilation :

- Optimiseur : Adam

Graphes montrant l'évolution de la perte au cours de l'entraînement



Epochs = 45



Epochs = 300

Conclusion

Méthodes utilisées :

- Réduction de dimension (PCA)
- Extraction de caractéristiques (zones, contours avec Sobel)
- SVM (noyau RBF)
- Réseau de neurones



Merci pour votre attention !