# Et si la Poste se mettait à l'IA?

Présentation Eric, Antoine et Arina

22/02/21

Objectif : développer un modèle capable d'identifier correctement le chiffre (entre 0 et 9) écrit dans une image

# Description

Taille du dataset entrainement : (42000, 785)

Taille du dataset test : (28000, 784)

Pas de valeurs nulles

Valeur minimale pour les pixels : 0
Valeur maximale pour les pixels : 255

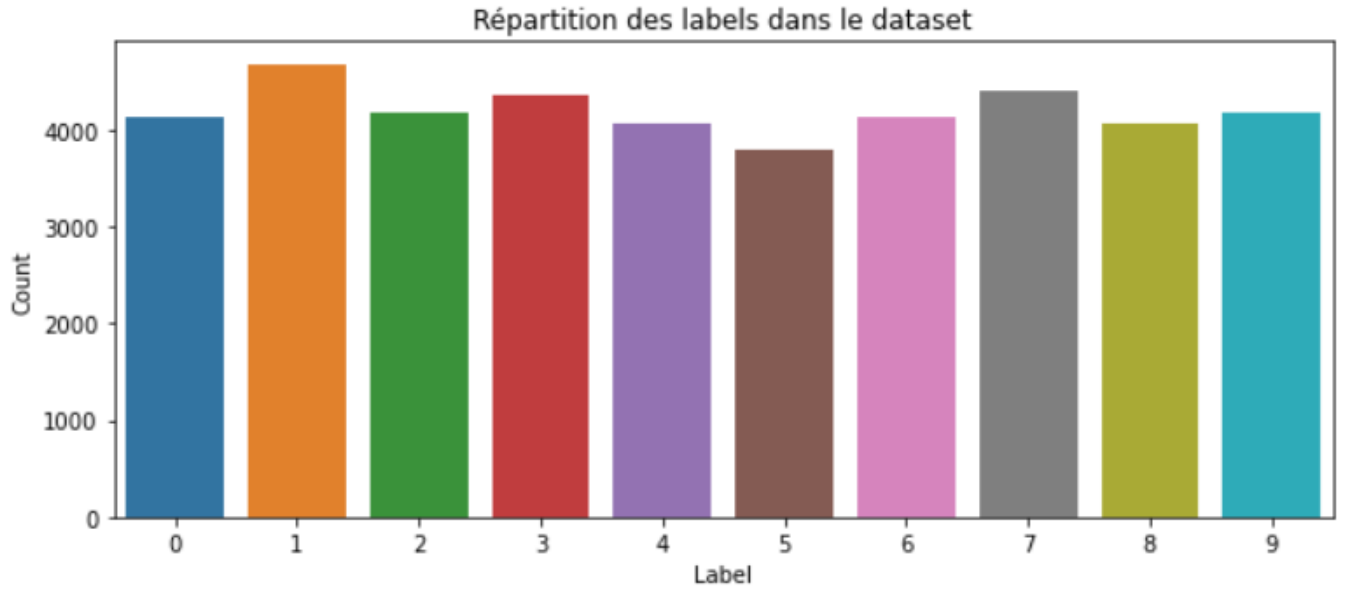
Variable expliquée : label

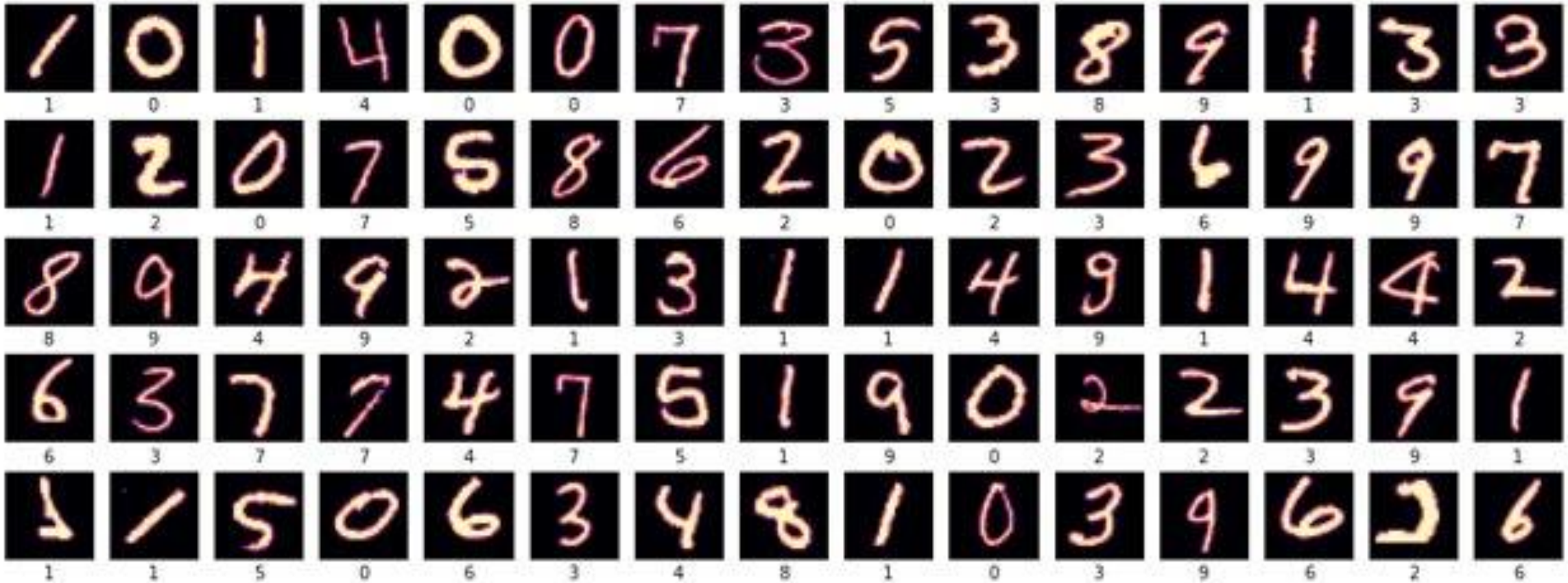
# Visualisation

Les digits sont tous bien représentés, proche de 10% pour chacun

grâce au stratify, les proportions sont respectées dans les deux jeux

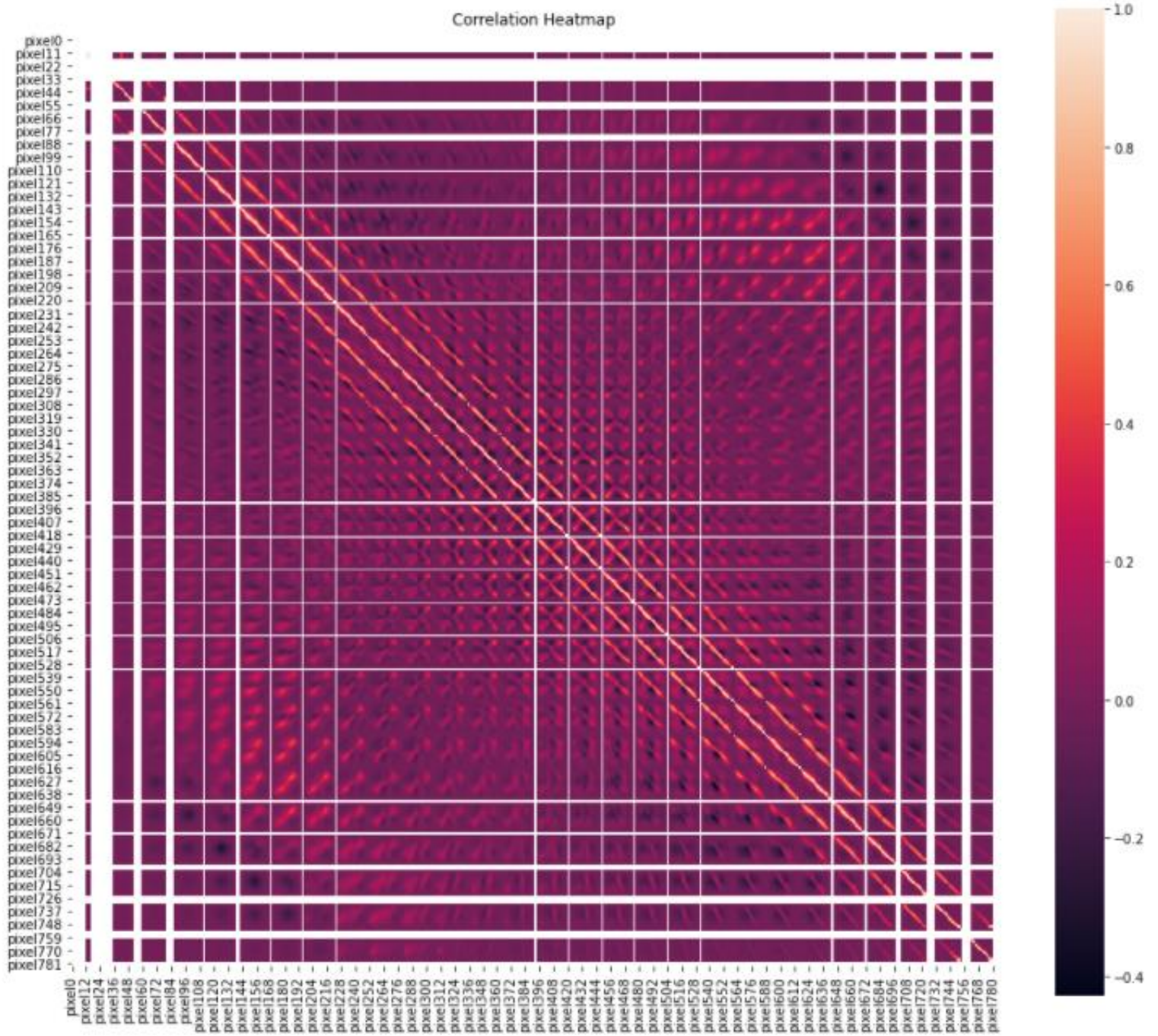
| label | count | percent | cumulative_count | cumulative_percent |
|---|---|---|---|---|
| 1.0 | 4684.0 | 11.152380952380952 | 4684.0 | 11.152380952380952 |
| 7.0 | 4401.0 | 10.47857142857143 | 9085.0 | 21.63095238095238 |
| 3.0 | 4351.0 | 10.359523809523811 | 13436.0 | 31.99047619047619 |
| 9.0 | 4188.0 | 9.971428571428572 | 17624.0 | 41.96190476190476 |
| 2.0 | 4177.0 | 9.945238095238096 | 21801.0 | 51.90714285714285 |
| 6.0 | 4137.0 | 9.85 | 25938.0 | 61.75714285714285 |
| 0.0 | 4132.0 | 9.838095238095237 | 30070.0 | 71.5952380952381 |
| 4.0 | 4072.0 | 9.695238095238096 | 34142.0 | 81.29047619047618 |
| 8.0 | 4063.0 | 9.673809523809524 | 38205.0 | 90.96428571428572 |
| 5.0 | 3795.0 | 9.035714285714286 | 42000.0 | 100.0 |




1 0 1 4 0 0 7 3 5 3 8 9 1 3 3
1 2 0 7 5 8 6 2 0 2 3 6 9 9 7
8 9 4 9 2 1 3 1 1 4 9 1 4 4 2
6 3 7 7 4 7 5 1 9 0 2 2 3 9 1
1 1 5 0 6 3 4 8 1 0 3 9 6 2 6

# Matrice de corrélation

Difficilement exploitable



# Pre-processing

- Réduction du dataset
- Réduction de dimensionnalités

# Train_test_split pour la réduction du dataset

Le volume du dataset pose problème au niveau du temps d'execution. Il est possible de le réduire en découpant le dataset plusieurs fois avec la fonction train_test_split.

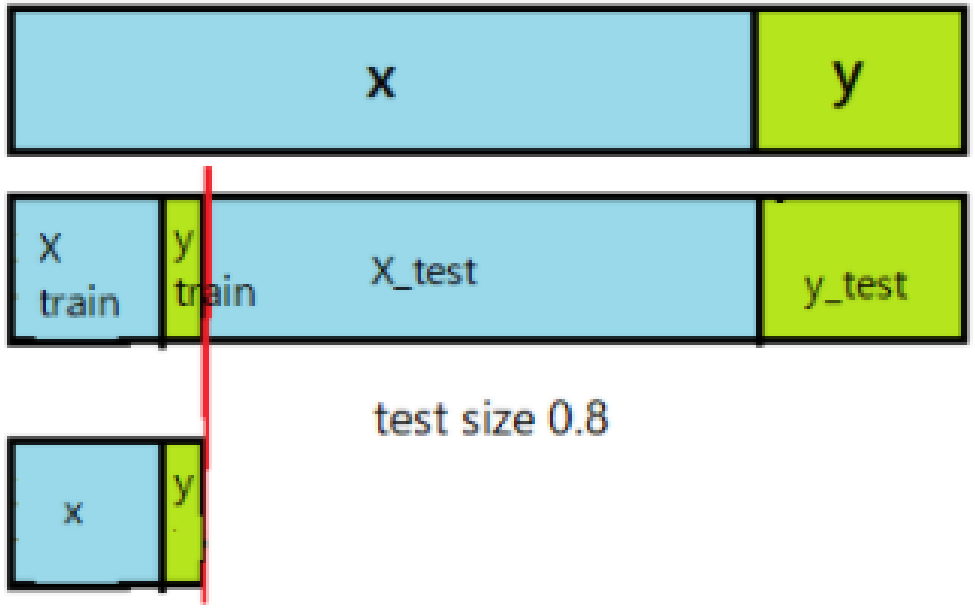
**Etape 1** : on indique la taille du test_size à 0.8, ce qui veut dire qu'on prend que 20% du dataset.

**Cette étape permet de réduire le dataset initial de 80%.**

**Etape 2** : on prend le X_train et le y_train de l'étape 1 (avec la taille du test_size normale : 0.2)

Cette étape correspond au train_test_split normal.

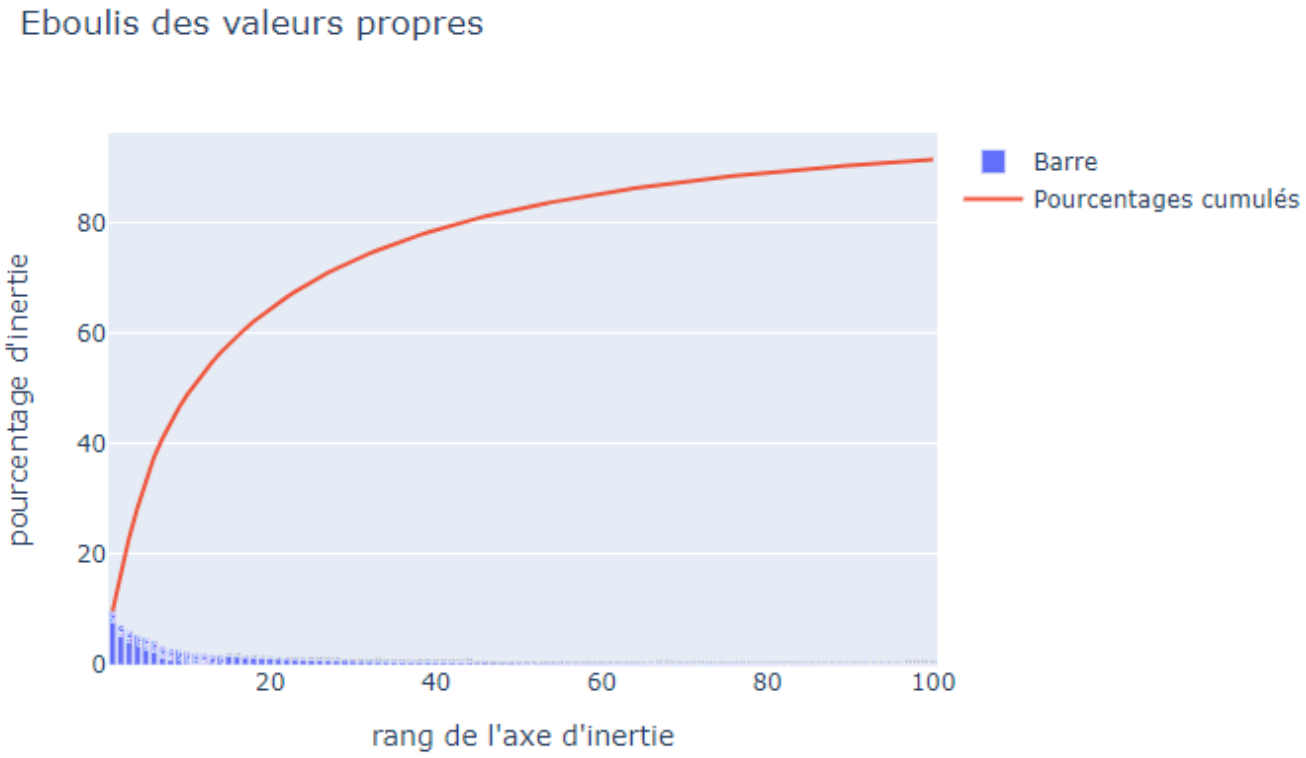
Avec les 20% de données restantes,
en veillant à ce qu'elles gardent la bonne proportion d'éléments avec stratify=y, on a un échantillon fonctionnel pour pouvoir essayer des modèles.



# ACP pour la réduction des dimensions

**Etape très importante dans un cas ou le nombre de features est important, comme c'est souvent le cas dans la pratique.**

**Rappel :** 1 feature = 1 dimension




C'est une méthode qui utilise des opérations matricielles simples à partir de l'algèbre linéaire et des statistiques pour calculer une projection des données d'origine dans le même nombre ou moins de dimensions.

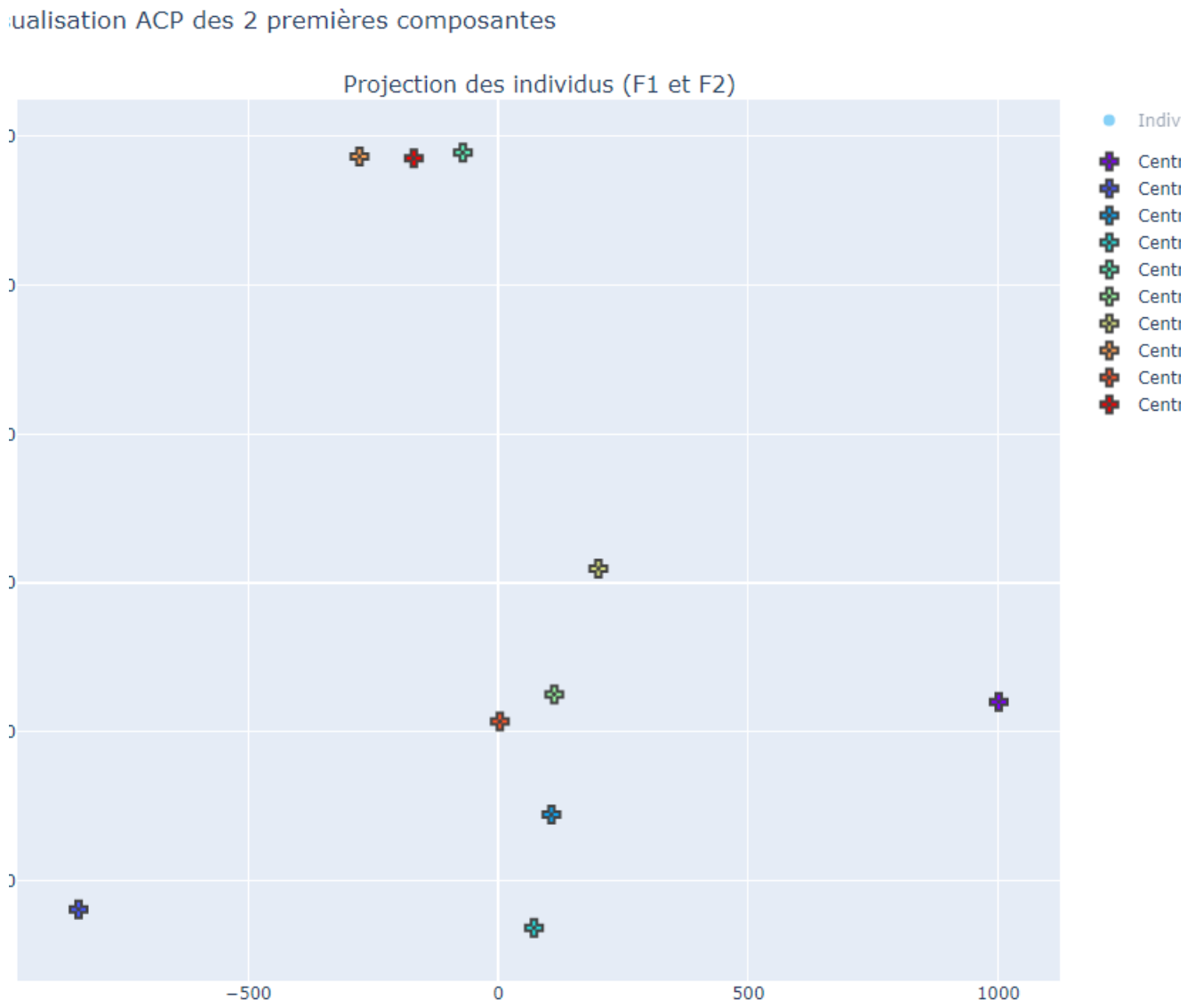
**Avec 86 composantes, on arrive à 90% de la variance expliquée.**

- **Raisons d'utiliser ACP :**

- Résume les données avec le moins d'information possible.
- Permet de surmonter la redondance des fonctionnalités dans l'ensemble de données.
- Capture des informations précieuses expliquant une variance élevée, ce qui donne la meilleure précision. Cela rend les visualisations de données faciles à gérer.
- Diminue la complexité du modèle et augmente l'efficacité du calcul.

- **Pour utiliser ACP :**

- Le dataset doit avoir beaucoup de features.
- Les données doivent être mises à l'échelle.



# Avantages de l'algorithme Support Vector Classifier

Nous devons prendre une décision non linéaire. Pas une simple ligne, plus complexe comme une courbe.

Le modèle SVC (Support Vector Classifier) était pertinent d'utilisation pour sa précision dans le traitement des données de type spatiale. Nous avons ainsi comparer 9 modèles disposant chacun de paramètres différents pour nous assurer les prédictions les plus précises relativement à un temps de travail le plus court.

SVC convertit des données dans un espace dimensionnel plus élevé et effectue une séparation assez compliquée d'éléments qui ne sont pas manifestement séparables.

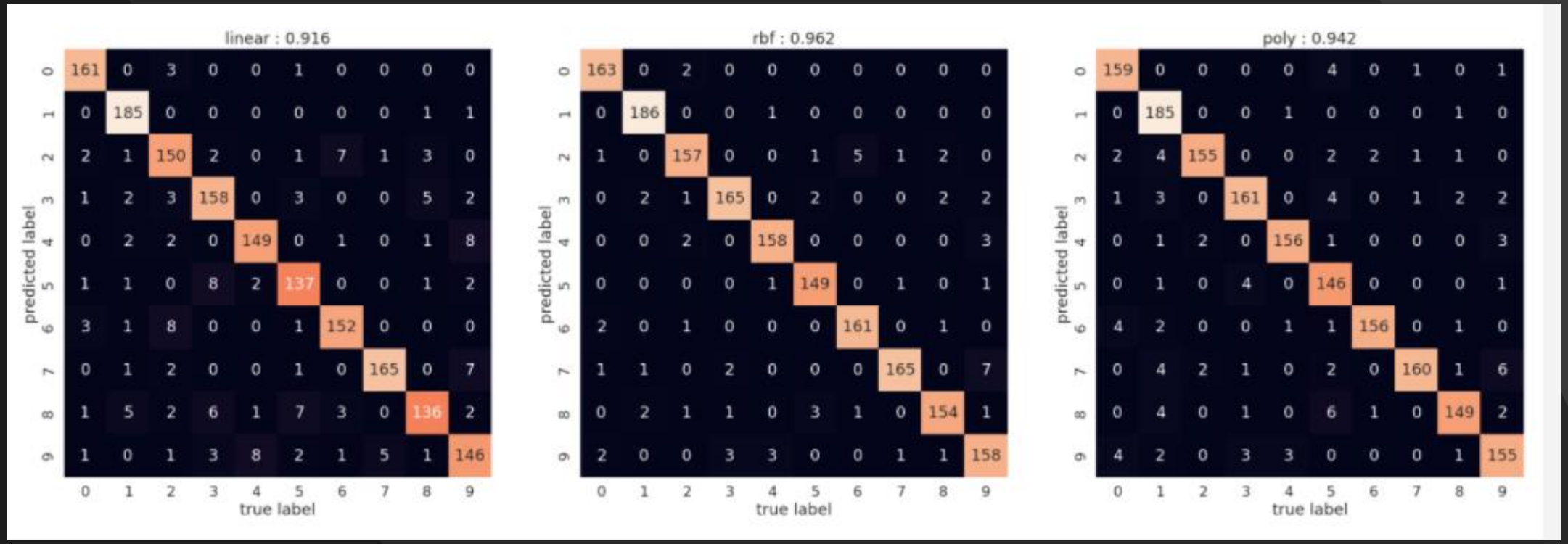
SVC produit une précision significative avec moins de puissance de calcul

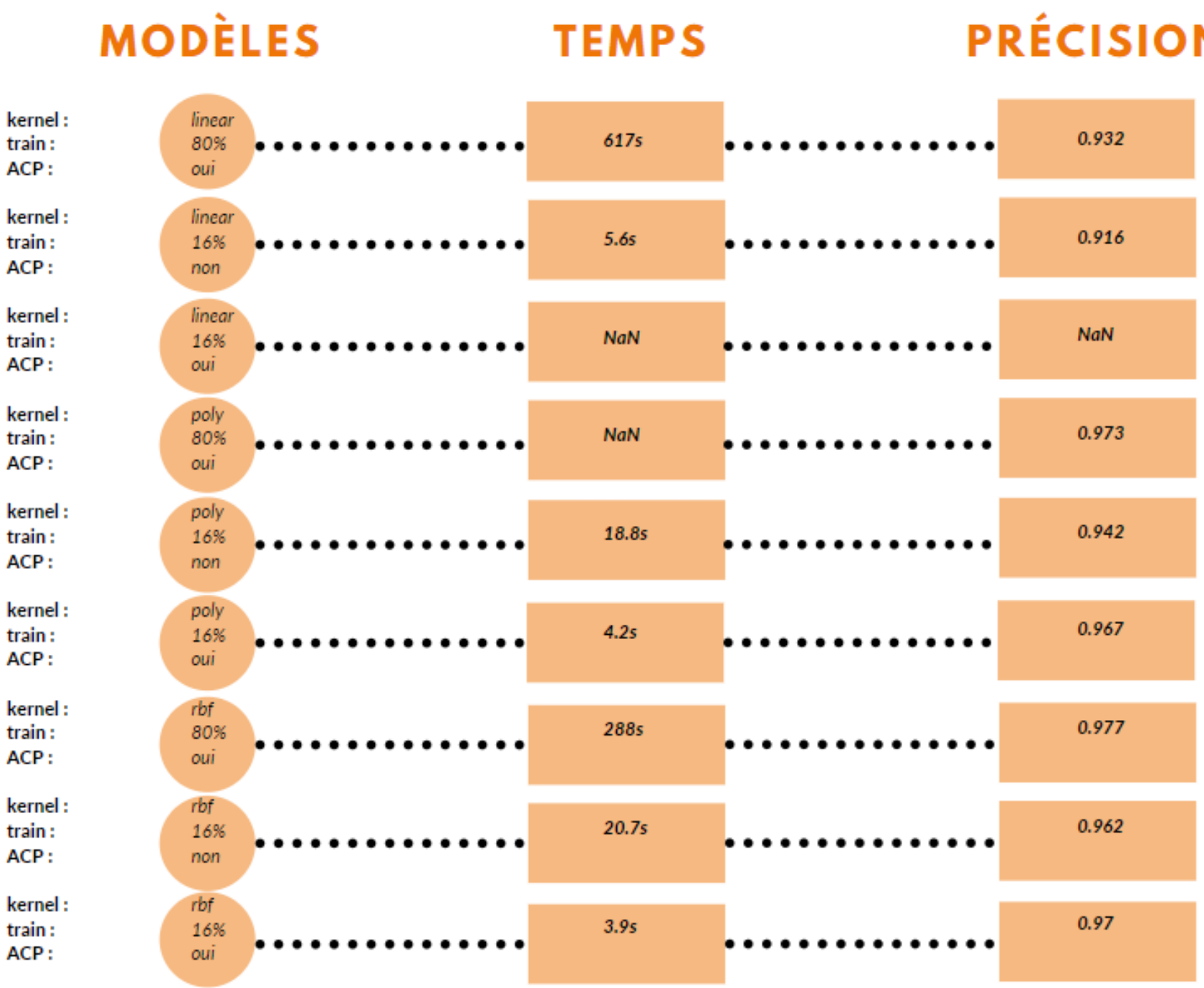
SVC maximise la séparation entre les classes pour rendre la décision plus efficace.

# Comparaison de modèles

- **SVC avec kernel 'linear':** 0.916
- **SVC avec kernel 'rbf':** 0.962
- **SVC avec kernel 'poly':** 0.942

- **Sur 20% du dataset**




MODÈLES
TEMPS
PRÉCISION
kernel : linear
train : 80%
ACP : oui
617s
0.932
kernel : linear
train : 16%
ACP : non
5.6s
0.916
kernel : linear
train : 16%
ACP : oui
NaN
NaN
kernel : poly
train : 80%
ACP : oui
NaN
0.973
kernel : poly
train : 16%
ACP : non
18.8s
0.942
kernel : poly
train : 16%
ACP : oui
4.2s
0.967
kernel : rbf
train : 80%
ACP : oui
288s
0.977
kernel : rbf
train : 16%
ACP : non
20.7s
0.962
kernel : rbf
train : 16%
ACP : oui
3.9s
0.97

# Choix du meilleur modèle

Le meilleur rapport précision/temps de traitement parmi nos modèles a été obtenu avec un kernel RBF, un échantillon d'entrainement à 14% de nos données totales, et une réduction des features par ACP. L'outil GridSearchCV nous a permis par la suite de trouver les hyper-paramètres les plus adaptés afin de gagner encore plus de précision.

Les meilleurs hyper-paramètres déterminer par l'outil GridSearchCV :

kernel : 'rbf'

C : 101

gamma : 1

**Précision gagnée : 0.97 -> 0.971**

# Conclusion

Notre modèle de SVC est capable de déterminer un chiffre avec 97,1% de chance de succès en analysant les niveaux de gris contenue dans une matrice de pixels.

Il a l'avantage d'être rapide et de nécessiter une quantité limitée de donnée.