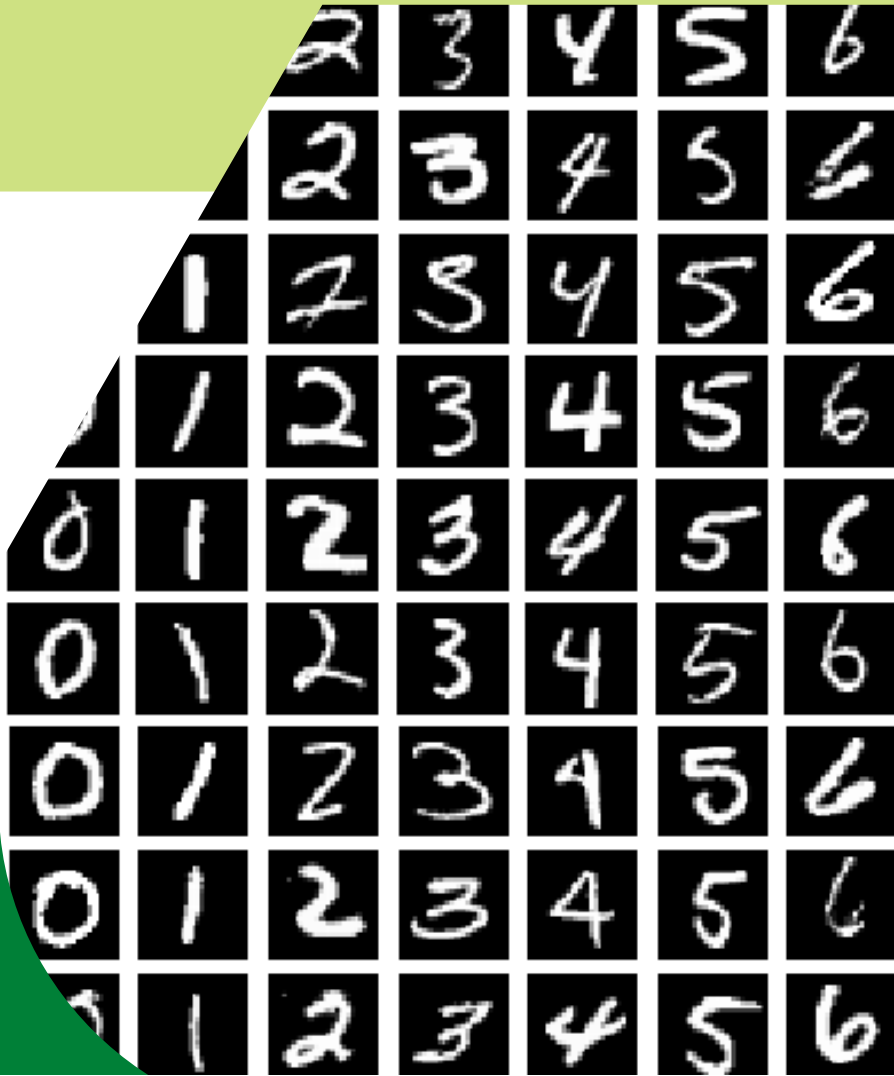
كلية العلوم والتقنيات – مراكش
FACULTÉ DES SCIENCES ET TECHNIQUE – MARRAKECH

Filière
SDAD S3
Année Universitaire
2022 /2023

RAPPORT DE CLASSIFICATION

de jeu donnée MNIST



Réaliser par:
ASAKOUR Ihsane
NAIM Kawtar

NOVEMBRE 2022

CHAPITRE I

Ce chapitre est consacré à donner une représentation générale à les notions qui ont une relation avec le sujet, en plus une description de notre jeu de données.

Optical Character Recognition (OCR)

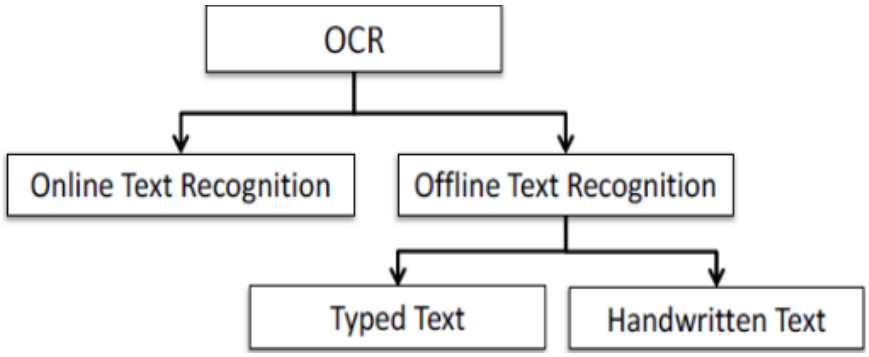
Introduction :

La reconnaissance optique de caractères (OCR) est un vaste domaine de recherche en Soft Computing, en intelligence artificielle (IA), en Pattern Recognition (PR) et en Computer Vision. L'OCR est une technique générale de textes manuscrits ou de numérisation d'images imprimées afin qu'elles puissent être traitées électroniquement, stockées et recherchées plus efficacement et correctement.
L'objectif d'une méthode OCR est la reconnaissance des manuscrits (comme les humains) dans un article difficile.

Domaines d'Application :

- Accompagnement des personnes aveugles et/ou malvoyantes.
- Reconnaissance automatique du code postal.
- Convertissez les documents numérisés en fichiers texte.

Types de L'OCR :



- Dans la reconnaissance de texte en ligne (Online Text Recognition), les mots et les caractères sont reconnus au moment de l'exécution aussi rapidement qu'ils sont écrits et, par conséquent, disposent d'informations temporelles.
- Une méthode de reconnaissance de texte hors ligne (Offline Text Recognition) traite une représentation statique d'un article. Ce type divisé en deux sous-catégories Typed Text et Handwritten Text. Dans les deux sous-catégories, une image de l'article obtenue à partir d'un scanner ou d'un appareil photo est traitée.

MNIST DataBase

Introduction :

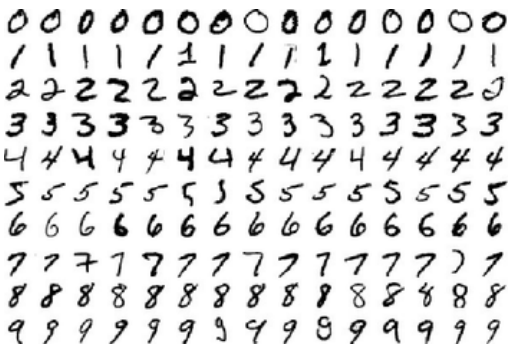
La Base de Données MNIST (Modified National Institute of Standards and Technology database) est une grande base de données de chiffres manuscrits qui est couramment utilisée pour l'entraînement de divers systèmes de traitement d'images. Étant donné que MNIST est un ensemble de données étiqueté qui associe des images de chiffres écrits à la main avec le nom du chiffre respectif, il peut être utilisé dans un apprentissage supervisé pour former des classificateurs
Il peut être le considérer comme le "Hello World" de l'apprentissage automatique

Historique :

Il a été créé en "remixant" les échantillons des ensembles de données originaux du NIST. Les créateurs ont estimé que puisque l'ensemble de données d'entraînement du NIST provenait d'employés de l'American Census Bureau, tandis que l'ensemble de données de test provenait d'étudiants américains du secondaire, il n'était pas bien adapté aux expériences d'apprentissage automatique.
La base de données MNIST contient 60 000 images d'entraînement et 10 000 images de test. La moitié de l'ensemble d'entraînement et la moitié de l'ensemble de test ont été extraites de l'ensemble de données d'entraînement du NIST, tandis que l'autre moitié de l'ensemble d'entraînement et l'autre moitié de l'ensemble de test ont été extraites de l'ensemble de données de test du NIST.

Description :

la base de données MNIST (Changed National Organization of Benchmarks and Innovation database) est une base de données étiquetée propice pour un apprentissage supervisé. Dans l'image ci-dessus, pour chaque chiffre, on a sa représentation sous forme d'image ainsi que son étiquette. Par exemple, pour le dernier chiffre en bas à droit, l'étiquette vaut 9 vu qu'il s'agit du chiffre 9. La représentation de ces chiffres est normalisée à travers tout le jeu de données MNIST. Ainsi, chaque chiffre est codé dans un format 28x28 pixels. En plus, chaque pixel peut prendre une valeur de 0 à 255. Cette plage de valeurs représente le niveau de gris Grayscale. En d'autres terme, chaque représentation d'une image est une matrice de dimension .



MNIST dataset

Chaque point de données MNIST, chaque image, peut être considéré comme un tableau de nombres décrivant le degré d'obscurité de chaque pixel. Par exemple, on pourrait penser àcomme quelque chose comme :

Puisque chaque image a 28 par 28 pixels, nous obtenons un tableau 28x28. Alors, nous pensons généralement que MNIST est une collection de vecteurs à 28x28= 784 dimensions.

Le jeu de données MNIST présent par défaut dans la librairie Scikit Learn, Le sous-ensemble comporte 70 000 chiffres que nous diviserons par la suite en deux sous ensembles :

- **L'ensemble d'entrainement** contient **60 000** images.
- **L'ensemble de test** contient **10 000** images.

Handwritten Character Recognition (HCR)

Introduction :

De nos jours, la reconnaissance de caractères manuscrits (HCR) est un domaine de recherche majeur remarquable et difficile dans le domaine du traitement d'images. La reconnaissance des alphabets manuscrits a été largement étudiée au cours des années précédentes. Actuellement, diverses méthodologies de reconnaissance sont bien connues et utilisées pour la reconnaissance d'alphabets manuscrits (caractères).
Les caractères manuscrits sont compliqués à reconnaître en raison des diverses techniques d'écriture manuscrite humaine et des différences de taille, de forme des lettres et d'angles.

Définition :

Comme son nom l'indique, la reconnaissance des caractères manuscrits est la capacité des ordinateurs à reconnaître les caractères manuscrits humains. Ou en termes simples, nous pouvons dire que c'est la capacité des ordinateurs à détecter le caractère présent dans une image particulière et à reconnaître ce caractère.

Domaines d'Application:

Le domaine d'application du HCR est le traitement de documents numériques tels que l'extraction d'informations issues de la saisie de données, de chèques, de demandes de prêts, de cartes de crédit, d'impôts, de formulaires d'assurance maladie, etc.

CHAPITRE II

Ce chapitre est consacré à indiquer les méthodes applicables pour notre problème de classification du jeu de données MNIST, puis ce que on a un problème de classification supervisé alors on a choisi les méthodes qui appartient à un apprentissage supervisées.

Les Méthodes Applicables

Decision Tree

Un algorithme d'apprentissage supervisé, qui on peut utiliser dans la classification, et se compose de nœud racine et des nœuds internes et feuilles. L'algorithme du Decision Tree analyse nos données. Il se base sur les features (le nombre de pixel des images qui est entre [0-255]) pour prédire à quelle classe appartient chaque chiffre [0-9], Il commence avec le feature que son algorithme trouve le plus pertinent c-à-d. le pixel qui divise notre data a de maximum nombres des classes pures , si on a un nœud contient une mélange des classes alors il faut poser un test sur un feature pour classer le dataset , Le Decision Tree continue ce processus jusqu'à obtenir des groupes qui correspondent le mieux possible à chacune de nos classes et ainsi classifier l'ensemble du dataset.

Le Decision Tree possède plusieurs hyperparamètres. Les plus basiques étant :

- criterion (« gini » ou « entropy ») – La fonction (« gini » ou « entropy ») a prendre en compte pour calculer l'incertitude sur la règle de discrimination utilisée.
- random_state (nombre entier) – permet de contrôler l'aléatoire dans l'algorithme.
- max_features (nombre entier) – Le nombre de feature à prendre en compte lors de la recherche du meilleur split
- max_depth (nombre entier) – la profondeur maximale de l'arbre , pour controler le overfitting.

Random Forest

Le terme Forêts Aléatoires désigne une famille de méthodes de classification, composée de différents algorithmes d'induction d'ensemble d'arbres de décision. Les étapes a fin de construire un modèle pour notre problème de classification sont :

- Commençant par l'étape d'apprentissage, consiste à construire un ensemble d'arbres de décision, chacun entraîné à partir d'un sous-ensemble "bootstrap" issu de l'ensemble d'apprentissage original (MNIST). En donnant 60 000 images pour l'ensemble d'entrainement et 10 000 pour l'ensemble de test.

Les principaux hyper paramètres qui sont présents implicitement dans le classificateur de forêt aléatoire qui doit être réglé afin d'augmenter la précision de notre modèle d'apprentissage sont :

n_estimators : est le nombre d'arbres à utiliser dans la forêt.

max_depth : la profondeur maximale de l'arbre , pour contrôler le sur-apprentissage.

- Alors dans l'étape suivant, on peut définir une boucle qui peut tester le modèle avec multiple valeur de n_estimators afin de définir la valeur de ce hyper paramètre qui donne la meilleur précision.
- Après, On applique le modèle, on créant un instance de RandomForestClassifier qui prend comme paramètre la valeur de n_estimators qu'on a construire à partir de la boucle précèdent
- En suite on entraine sur notre ensemble d'apprentissage.
- On Fin, on prédite sur l'ensemble de test, que le modèle n'a pas vue dans la phase d'apprentissage
- Après, dans la phase d'évaluation, on peut tester la performance du modèle on calculant la précision, le taux d'erreur, le taux de réussie, Ainsi la matrice de confusion.

Naïve Bayes :

Dans notre dataset les pixels X appartient à {0,1} (Pixels noirs et blancs), et Y prédire si un chiffre est partient a un classe parmi les 10 classes [0-9].

Etant donné un objet X, la méthode consiste à calculer la probabilité d'appartenance de X à chaque classe, puis choisir celle qui maximise cette valeur.

Nous supposons que le modèle de probabilité pour chaque pixel est gaussien et que la probabilité de chaque classe, c'est-à-dire chiffre, est égale.

Autrement dit, P(c = 0) = P(c = 1) = : : : = P(c = 9). Soit x = (x1; x2; : : : ; x784) le vecteur des valeurs de pixels pour une image donnée et c est la classe ou le chiffre, de 0 à 9. Par conséquent, pour comparaison, nous éliminons le Pr(C).

Il existe trois types de modèles Naïve de Bayes : Gaussien, Multinomial et Bernoulli.

Gaussien Naïve Bayes : Peut effectuer des mises à jour en ligne des paramètres du modèle

- Commençant par l'étape d'apprentissage, consiste à deviser notre jeu de données à l'ensemble d'apprentissage et de test, En donnant 60 000 images pour l'ensemble d'entrainement et 10 000 pour l'ensemble de test.
- Apres dans la pahse d'appretissage, On va crée un instace de GaussianNB on spécifiant comme parametre var_smooting

ce paramètre spécifie la portion de la plus grande variance de toutes les caractéristiques qui est ajoutée aux variances pour la stabilité du calcul.

- En suite on entraine sur notre ensemble d'apprentissage X_train, y_train
- On Fin, on prédite sur l'ensemble de test, X_test, y_test.
- Après, dans la phase d'évaluation, on peut tester la performance du modèle on utilisant les mêmes méthodes utilisées pour le module Random Forest

Multinomial Naïve Bayes : convient à la classification avec des caractéristiques discrètes

Le classificateur multinomial Naïve Bayes, va suivre les mêmes étapes que gaussien Naïve Bayes, Soit à la phase d'apprentissage ou bien dans l'application du modèle.
la seule déférente est au niveau des paramètres utilisé dans la fonction MultinomialNB, qui prend alpha comme paramètre.
alpha (valeur réelle) : Paramètre de lissage additif

Bernoulli Naïve Bayes : Comme MultinomialNB, ce classifieur convient aux données discrètes. La différence est que BernoulliNB est conçu pour les fonctionnalités binaires.

Le classificateur bernoulli Naïve Bayes, va suivre les mêmes étapes que multinomial Naïve Bayes, Soit à la phase d'apprentissage ou bien dans l'application du modèle.
Même le paramètres utilisé dans la fonction BernoulliNB, il va utilisé alpha comme paramètre.
alpha (valeur réelle) : Paramètre de lissage additif

CHAPITRE III

Ce chapitre est consacré à l'implémentation de la méthode choisi, Random Forest, Ainsi l'évaluation du modèle en utilisant la précision, le taux d'erreur, le taux de réussite, la matrice de confusion...

Nous avons éliminé la méthode de l'arbre de décision (Decision Tree), car elle peut mal classer un nombre, l'utilisation cette méthode entraînera une seule prédiction de la classe dans le résultat. Et si un arbre est long, nous pouvons obtenir un sur-apprentissage de l'ensemble de données MNIST.

Et pour la méthode naïve de Bayes, a également un inconvénient tel que supposer que 95% sont des pixels correctement classés. Seuls 5% des pixels sont en fait des bords, alors la plupart des pixels de bord sont manquants.

Pour Random forest, il est stable pour des problèmes étiquetés.

Random Forest

Description du jeu de données avec Python :

Nous avons importer Le jeu de donnée MNIST à partir de librairie Scikit Learn, Le sous-ensemble comporte 70 000 chiffres.

```
from collections import Counter # To count the Items in each classe
from sklearn.datasets import fetch_openml # Used to load the MNIST Original dataset
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
```

```
X, y = fetch_openml('mnist_784', version=1, return_X_y=True)
print('Overall # of samples is', y.shape[0])
print('Size of the features is:', X.shape)
```

```
Overall # of samples is 70000
Size of the features is: (70000, 784)
```

Dans notre jeu de donnée MNIST, On a l'équilibre entre les classes , chaque classe contient presque 7877 individus.

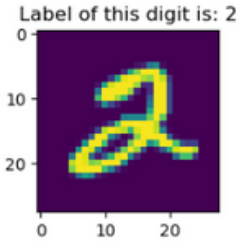
```
er = Counter(y)
(counter)
```

```
-({'1': 7877, '7': 7293, '3': 7141, '2': 6990, '9': 6958, '0': 6903, '6': 6876, '8': 6825, '4': 6824, '5':
```

Affichage un chiffres

La fonction show_digit permet d'afficher un chiffre de jeu de données MNIST.

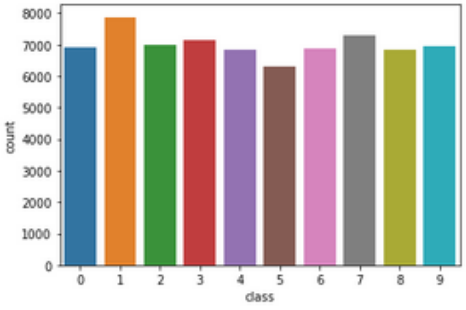
```
def show_digit(x_vec, label):
    x_vec = x_vec.to_numpy()
    x_mat = x_vec.reshape(28, 28)
    plt.figure(figsize=(2,2))
    plt.imshow(x_mat)
    plt.title('Label of this digit is: ' + label)
show_digit(X.loc[5,:], y[5])
```



Nous diviserons MNIST en deux sous ensembles : L'ensemble d'entrainement 60 000 images, et L'ensemble de test 10 000 images

```
# create train test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/7.0)
print(X_train.shape)
print(y_train.shape)
```

```
(60000, 784)
(60000,)
```



Importer les bibliothèques

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix,
ConfusionMatrixDisplay, precision_recall_fscore_support
```

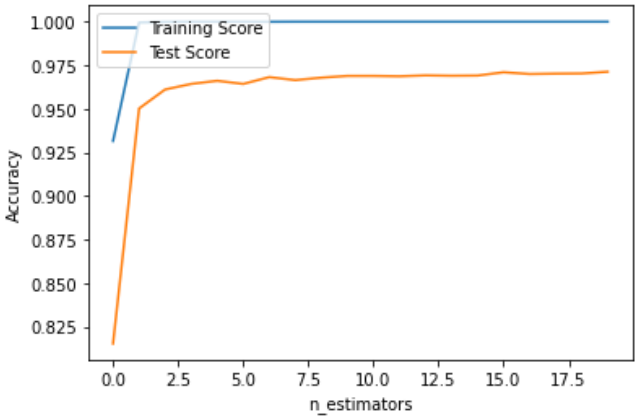
L'estimation du paramètre n_estimators

Pour Random forest nous utilisons un boucle pour entrainer notre modèle par n_estimators , qui indique le nombre des arbres à créés, et on a affiché la précision dans l'ensemble d'entrainement et l'ensemble de test

```
val_score_train=[]
val_score_test=[]
for i in range(1,201,10):
    rfc=RandomForestClassifier(n_estimators=i)
    rfc.fit(X_train, y_train)
    score_train=rfc.score(X_train,y_train)
    val_score_train.append(score_train)
    score_test=rfc.score(X_test,y_test)
    val_score_test.append(score_test)
    print('ITER ',i,' TRAINING SCORE ',score_train,' TEST SCORE ',score_test)
```

Ce code permet d'afficher le courbe accuracy test et train pour chaque nombre d'arbre utilisé(n_estimators)

```
plt.plot(val_score_train)
plt.plot(val_score_test)
plt.xlabel('n_estimators')
plt.ylabel('Accuracy')
plt.legend(['Training Score','Test Score'],loc='upper left')
plt.show()
```



Résultat

```
ITER  1  TRAINING SCORE  0.9315333333333333  TEST SCORE  0.8153
ITER  11  TRAINING SCORE  0.99955  TEST SCORE  0.9502
ITER  21  TRAINING SCORE  0.9999166666666667  TEST SCORE  0.9611
ITER  31  TRAINING SCORE  0.9999833333333333  TEST SCORE  0.9643
ITER  41  TRAINING SCORE  1.0  TEST SCORE  0.9661
ITER  51  TRAINING SCORE  1.0  TEST SCORE  0.9643
ITER  61  TRAINING SCORE  0.9999666666666667  TEST SCORE  0.9682
ITER  71  TRAINING SCORE  1.0  TEST SCORE  0.9665
ITER  81  TRAINING SCORE  1.0  TEST SCORE  0.9679
ITER  91  TRAINING SCORE  1.0  TEST SCORE  0.9689
ITER  101  TRAINING SCORE  1.0  TEST SCORE  0.9689
ITER  111  TRAINING SCORE  1.0  TEST SCORE  0.9687
ITER  121  TRAINING SCORE  1.0  TEST SCORE  0.9692
ITER  131  TRAINING SCORE  1.0  TEST SCORE  0.969
ITER  141  TRAINING SCORE  1.0  TEST SCORE  0.9691
ITER  151  TRAINING SCORE  1.0  TEST SCORE  0.971
ITER  161  TRAINING SCORE  1.0  TEST SCORE  0.97
ITER  171  TRAINING SCORE  1.0  TEST SCORE  0.9702
ITER  181  TRAINING SCORE  1.0  TEST SCORE  0.9703
ITER  191  TRAINING SCORE  1.0  TEST SCORE  0.9713
```

Le résultat de la boucle est affiché sous forme de : nombre d'arbre avec la précision dans l'ensemble d'entrainement et l'ensemble de test

L' appliquation de l' algorithme

D'après les figures précédentes on a que best nombre n_estimators est 191 qui donne une précision de 100% dans l'ensemble d'entrainement et 97% dans l'ensemble de test.

```
rfc=RandomForestClassifier(n_estimators=191)
```

```
rfc.fit(X_train, y_train)
y_pred=rfc.predict(X_test)
```

Alors on a appliqué Random Forest sur ce paramètre et on a entrainé notre modèle sur l'ensemble d'entrainement et prédire le résultat sur l'ensemble de test

L'Evaluation

Aprés avoir appliqué l' algorithme de Random Forest, Nous avons besoins d'outils pour savoir s'il est bien entrainé ou pas, Et pour cela nous avons calculé le taux d'erreur, le taux de réussite, , précision, taux de détection de Random Forest, et F1 score de RF.

Résultat

```
precision,recall,fscore,none= precision_recall_fscore_support(y_test, y_pred, average='weighted')
print("Taux d'erreur : {}".format(1-accuracy_score(y_test,y_pred)))
print("Taux de réussite : {}".format(accuracy_score(y_test,y_pred)))
print('Précision : '+(str(precision)))
print('Taux de détection : '+(str(recall)))
print('F1-score : '+(str(fscore)))
```

```
Taux d'erreur : 0.02959999999999996
Taux de réussite : 0.9704
Précision : 0.9703853649942339
Taux de détection : 0.9704
F1-score : 0.9703683216413936
```

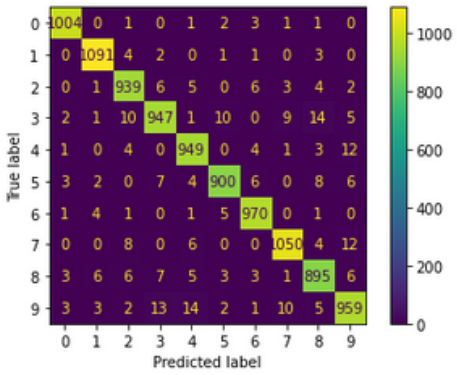
Random forest est bien prédit sur le jeu de données MNIST avec un taux de réussite de 97%

Matrice de Confusion

```
cm=confusion_matrix(y_test, y_pred)
cm
```

```
array([[1004,    0,    1,    0,    1,    2,    3,    1,    1,    0],
       [   0, 1091,    4,    2,    0,    1,    1,    0,    3,    0],
       [   0,    1,  939,    6,    5,    0,    6,    3,    4,    2],
       [   2,    1,   10,  947,    1,   10,    0,    9,   14,    5],
       [   1,    0,    4,    0,  949,    0,    4,    1,    3,   12],
       [   3,    2,    0,    7,    4,  900,    6,    0,    8,    6],
       [   1,    4,    1,    0,    1,    5,  970,    0,    1,    0],
       [   0,    0,    8,    0,    6,    0,    0, 1050,    4,   12],
       [   3,    6,    6,    7,    5,    3,    3,    1,  895,    6],
       [   3,    3,    2,   13,   14,    2,    1,   10,    5,  959]],
      dtype=int64)
```

```
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```



On peut remarquer dans la matrice de confusion que le chiffre 1 et 7 qui bien prédit et le chiffre 8 est celui qui n'a pas été bien prédit par le modèle.

CHAPITRE IV

Ce chapitre est axé a donnée une description à notre jeu de donnée ainsi présenter les déférents étapes pour avoir a la fin des caractères distingues.
En suite expliquer chaque étape et donner le code python utilisé.
En fin, implémenter une méthode de classification sur l'ensemble des caractères obtenu a partir de notre jeu de données.

Le jeu de données

Déscription:

Le jeu de données utilisé dans cette phase est sous forme d'un ensemble de mots en arabe à reconnaître, dont chacun des étudiants de master SDAD doivent écrits dans une feuille les noms et les prénom des autres étudiants, a fin d'avoir des tableau contient des mots en arabe manuscrites

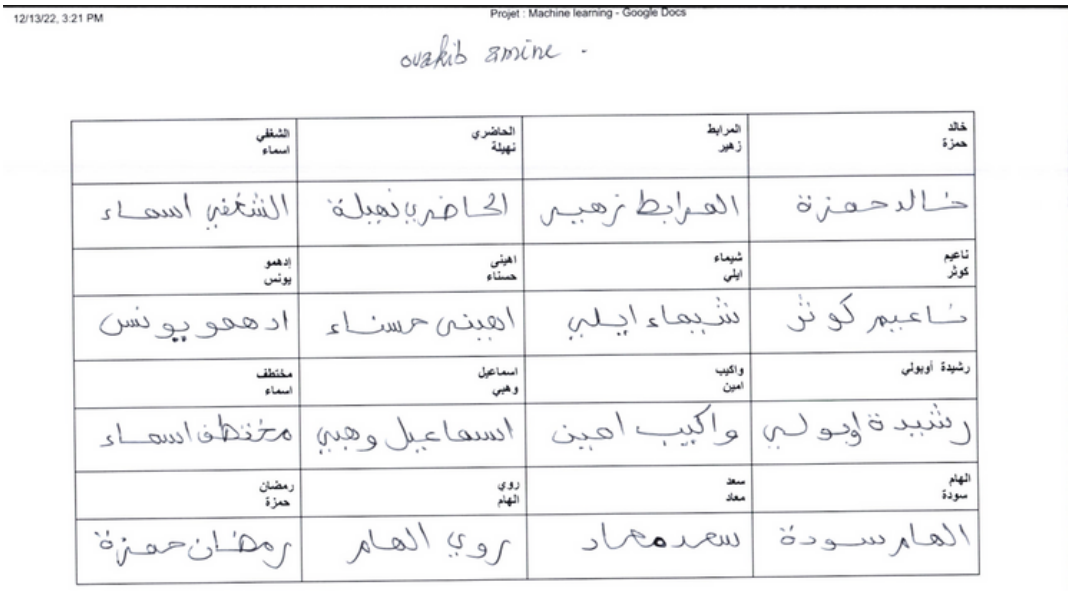


Figure 1 : Exemple d'un tableau de jeu de données

Les étapes:

Le prétraitement de ce jeu de données consiste à préparer les données de manière à ce qu'elles puissent être utilisées efficacement dans un modèle de machine learning.

- Avant de commencer, il est nécessaire de sélectionner les cases qui contient les mots manuscrits.
- Ensuite, on va faire une segmentation des lignes/ des mots, a fin d'avoir des mots individuels, pour faciliter la partie de la segmentation de caractères
- Après qu'on obtient les mots individuels, on va faire une segmentation des caractères.
- Finalement, On va appliquer un modèle de classification supervisé pour faire une reconnaissance des caractères.

Sélectionner les cases

Comme il est indiqué dans la partie qui donne une description de notre jeu de données, le je de données est sous forme d'un ensemble de tableau contient les noms et les prénoms manuscrites et non manuscrites.
on s'intéresse plus sur les mots manuscrites, donc il est nécessaire de deviser le tableau a fin d'avoir seulement les cases contient les mots manuscrites.

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
from split_image import split_image
from PIL import Image
import image_slicer   # Importing Image class from PIL module
import pathlib
import os
initial_count = 0
#method is then called on this Path object to iterate over the files and subdirectories within the directory.
for path in pathlib.Path(r"C:\Users\kawtar\Desktop\ML\projet\image_bruts").iterdir():
    if path.is_file():
        initial_count += 1
nombre_elt = initial_count # count number of file exist
```

ce code permet de parcourir les images dans le dossier images_bruts qui contient originale
tableau puis il calcule le nombres des images

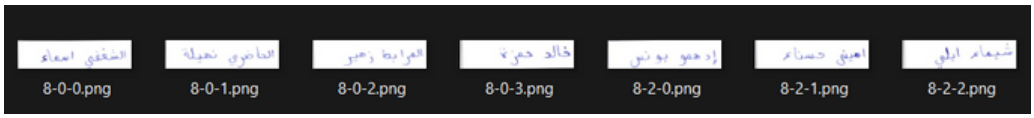
```
for i in range(1,nombre_elt+1): # range number of files
    # read image par image and store it in the variable "im"
    im = Image.open(r"C:\Users\kawtar\Desktop\ML\projet\image_bruts\machine learning0%s.bmp"%i)
    # Size of the image in pixels (size of original image)
    # (This is not mandatory)
    width, height = im.size #  get width et size for each image
    # Set the coordinates for the top-left and bottom-right corners
    # of the cropped image, depending on the value of i because  we have a different size for page 1 and page 2
    if(i%2==1):
        left = 320
        top = 530
        right = 2139
        bottom = height - 299
    # Cropped image of above dimension
    if(i%2==0):
        left = 320
        top = 350
        right = 2139
        bottom = height - 480
    # Crop the image using the coordinates set above
    im1 = im.crop((left, top, right, bottom))
    plt.imshow(im1)
    # Save the cropped image to a new file
```

Le 2ème code ayant pour objectif d'éliminer les écritures qui ne sont pas importantes dans notre jeu de données tels que le titre d'image
et il garde Just le tableau encadrer par les noms et les prénoms enregistrant les images recadrées dans le répertoire "image_tab".
C' est pour cela on a utiliser la fonction crop qui prend comme paramètre les valeurs des dimensions (left , top , right , bottom) .
La taille de l'image (page 1 et page 2) est différentes . Si i est impair, les coordonnées en haut à gauche et en bas à droite sont définies sur certaines valeurs. Si i est pair, les coordonnées sont définies sur des valeurs différentes.

Résultat

```
# function for crop eatch table to the case
def crop(infile,height,width,nombre_elt):
    im = Image.open(infile) #open tab image
    imgwidth, imgheight = im.size # set size
    for k in range(imgheight//height):
        for j in range(imgwidth//width):
            box = (j*width, k*height, (j+1)*width, (k+1)*height) #set var of box
            if(k%2 == 0):  # we have a cases manuscrit in ligne binairy
                path=os.path.join(r'C:\Users\kawtar\Desktop\ML\projet\image','%s-%s-%s.png'%(nombre_elt,k,j))
                # crop just manuscrit image in the path
                im.crop(box).save(path)
# for each number of table image
for i in range(1,nombre_elt+1):
    # crop case within height =114 and width = 452
    crop(r'C:\Users\kawtar\Desktop\ML\projet\image_tab\%s.png'%i,114,452,i)
```

Ce code permet de recadrer un fichier image en images plus petites, chaque image plus petite représentant une cellule de grille dans un tableau.
Le code le fait en parcourant l'image et en divisant sa largeur et sa hauteur en blocs égaux. Les dimensions de chaque bloc sont spécifiées par les variables de hauteur et de largeur.
Les blocs résultants sont ensuite enregistrés juste les noms et prénoms manuscrites sous forme de fichiers image séparés, avec leurs noms de fichiers contenant l'index de la table d'où ils proviennent et leur position dans la table.



Segmentation des Lignes / des mots

La segmentation de lignes/des mots consiste à diviser une image contenant du texte en différentes lignes/mots. Cela peut être utile dans le cadre de l'analyse de données d'images, par exemple pour extraire du texte à partir d'une image pour l'utiliser dans un traitement de texte ou pour l'analyse de texte.
mais pour notre objectif il est utile pour extraire des mots individuels a fin de les utiliser pour faire une segmentation des caractères pour le problème de la reconnaissance des caractères

```python
import cv2
import matplotlib.pyplot as plt

# Read the image file
image = cv2.imread('test_image.jpg')

# Convert the image to grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Apply Otsu's thresholding to the grayscale image to create a binary image
thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)[1]

# Find the contours in the binary image
cnts = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# Check the format of the output from cv2.findContours
cnts = cnts[0] if len(cnts) == 2 else cnts[1]

# Create an empty list to store the coordinates of the bounding boxes
list1 = []

# Iterate through the contours
for c in cnts:
    # Get the bounding box coordinates for the contour
    x,y,w,h = cv2.boundingRect(c)
    # Draw the bounding box on the image
    cv2.rectangle(thresh, (x, y), (x + w, y + h), (100,255,12), 1)
    # Add the bounding box coordinates to the list
    list1.append((x, y, x + w, y + h))

# Display the image with the bounding boxes
plt.figure(figsize=(5,5))
plt.imshow(thresh,cmap="gray")
```

cv2.cvtColor : cette fonction prend deux paramètre:
l'image
cv2.COLOR_BGR2GRAY :qui indique on veut convertir cette image au niveaux de gris

cv2.threshold : cette fonction prend l'image au niveaux de gris et le convertir à un image binaire

cv2.findContours : cette fonction a pour objectif de trouver des contours dans une image, elle prend 3 paramètres :

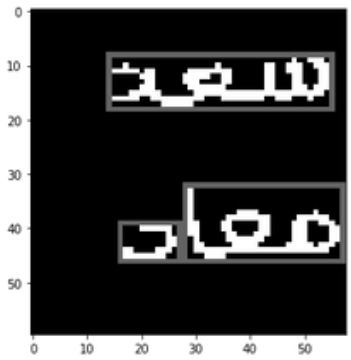
l'image binaire

cv2.RETR_EXTERNAL: qui indique à la fonction de ne récupérer que les contours extérieurs extrêmes des objets dans une image.

cv2.CHAIN_APPROX_SIMPLE : Il est utilisé dans la fonction pour spécifier la manière dont les points de contour doivent être représentés.

Le code : cnts=cnts[0] if len(cnts)==2 else cnts[1] permet de vérifier si l'image contient un seule ligne ou bien deux lignes

Résultat



la boucle permet de dessiner des cadres de délimitation autour des contours d'une image et stocker les coordonnées du cadre de délimitation dans une liste appelée list1.
Voici une ventilation de ce qui se passe :

- Le code itère sur chaque contour de la liste cnts.
- Il utilise la fonction cv2.boundingRect pour obtenir les coordonnées de la boîte englobante pour le contour actuel. Cette fonction renvoie un tuple contenant les coordonnées (x, y) du coin supérieur gauche de la boîte englobante, ainsi que la largeur (w) et la hauteur (h) de la boîte englobante.
- Il utilise la fonction cv2.rectangle pour dessiner un rectangle sur le seuil de l'image en utilisant les coordonnées de la boîte englobante. Le rectangle est dessiné avec une épaisseur de 1 pixel et une couleur de (100, 255, 12).
- Il ajoute les coordonnées de la boîte englobante à la liste list1.

Ce code dessinera des boîtes englobantes autour de tous les contours de l'image et stockera les coordonnées de la boîte englobante dans la liste list1.

Segmentation des Caractères

La segmentation des caractères est un processus qui consiste à séparer les différents caractères d'un texte en utilisant des algorithmes de traitement de l'image ou du langage naturel. Elle est souvent utilisée dans les applications de reconnaissance de caractères, comme les systèmes de reconnaissance de la parole ou les systèmes de reconnaissance de texte dans les images.
Pour cette étape, on a essaye plusieurs méthodes comme l'utilisation de OPENCV et PYTESSERACT

Limitation de OPENCV:

OpenCV est une bibliothèque logicielle de traitement d'image et de vision par ordinateur très populaire. Elle permet de réaliser une grande variété d'opérations sur des images, notamment la détection de contours et de formes géométriques spécifiques, la reconnaissance de visages et de caractères, la détection de mouvement et la segmentation d'images.

Le problème qu'on a rencontré en utilisant OPENCV, Il est effectivement possible seulement pour détecter l'espace dans une image. Une fois que l'espace a été détecté, il découpe l'image en utilisant les coordonnées de l'espace comme guide, comme monté dans la figure 2.

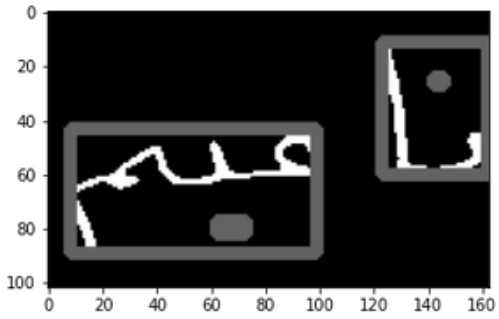


Figure 2 : Résultat retourné par OPENCV

Limitation de PYTESSERACT:

Pytesseract est une bibliothèque Python qui permet d'utiliser le moteur de reconnaissance optique de caractères Tesseract dans des scripts Python. Elle est conçue pour extraire du texte à partir d'images en utilisant la reconnaissance optique de caractères (OCR).

Le problème qu'on a rencontré en utilisant PYTESSERACT, Il est effectivement utile pour détecter les caractères dans une image, mais le problème il est impossible de deviser l'image pour récupérer seulement les segment des images contient les caractères reconnaître par PYTESSERACT.

```
['0' ,'44' ,'23' ,'5' ,'2' ,'ي']
['0' ,'44' ,'41' ,'5' ,'23' ,'ل']
['0' ,'44' ,'56' ,'5' ,'41' ,'ي']
['0' ,'44' ,'71' ,'5' ,'56' ,'إ']
['0' ,'40' ,'169' ,'9' ,'89' ,'ء']
['0' ,'44' ,'113' ,'5' ,'101' ,'ا']
['0' ,'44' ,'128' ,'5' ,'113' ,'م']
['0' ,'44' ,'146' ,'5' ,'128' ,'ي']
['0' ,'44' ,'155' ,'5' ,'146' ,'ق']
```

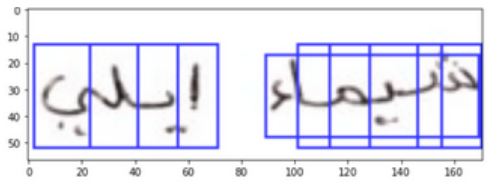


Figure 3 : Résultat retourné par PYTESSERACT

Solutions

A fin de ne pas être bloquer dans cette partie, on a essayé de faire la segmentation des caractères manuellement en utilisant un site web : https://toolxox.com/splitter.php

Implémentation de méthode de classification

Modèle choisi :

Pour ce problème de reconnaissance des caractères, on a choisi d'appliquer le modèle Random Forest parce que :

- On a un problème de classification supervisé.
- On a déjà utilisé ce modèle pour la classification du jeu de données MNIST, c'est celui qui a donné les meilleurs résultats

Les étiquettes:

Après la segmentation des caractères que nous avons faite manuellement, nous définirons les classes de notre jeu de données qui contient 61 dossier au total.
Les classes seront définies de manière à ce que chaque lettre dans une position soit comptée comme une étiquette, par exemple, la lettre bae (ب) qui est écrite au début du mot est différente de celle au milieu du mot. ils seront comptés comme des classes déférents

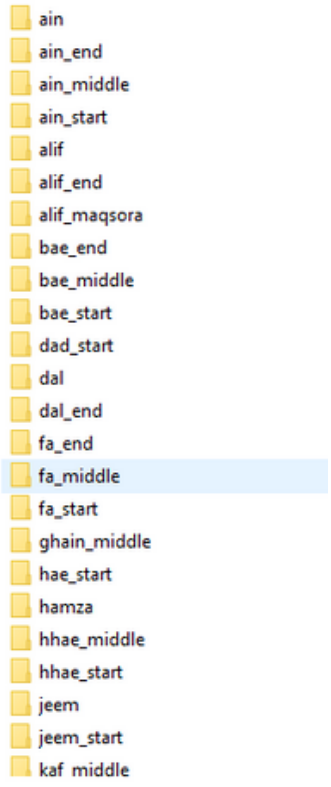


Figure 4 : Les étiquettes

comme illustré ici, chaque dossier présente des classes différentes à l'autre, et dans chaque dossier, il y a un ensemble de caractères du même type indiqué dans le nom du dossier.

Chaque lettre est écrite avec 4 méthodes différentes

- au début
- entre
- à la fin
- isolée

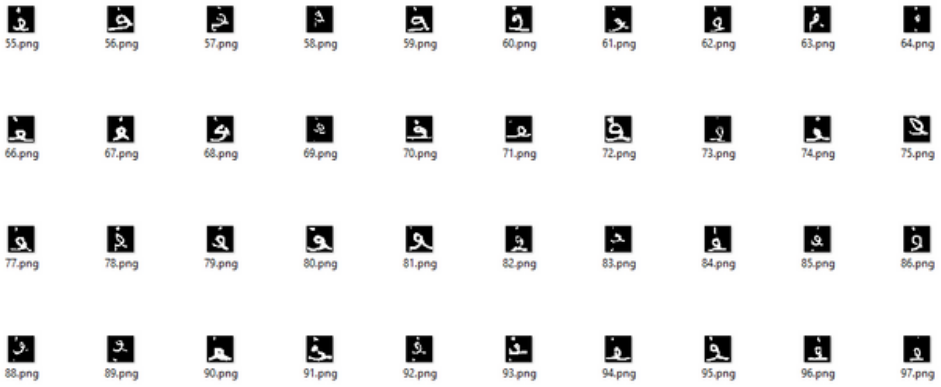


Figure 5 : Les lettres dans le dossier fa_middle (ف وسط الكلام)

L'application du modèle

On commence par l'importation des bibliothèques nécessaires

```
import os
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import cv2
import skimage
from   imblearn.over_sampling import SMOTE
from   collections import Counter
from   PIL import Image
from   skimage.transform import resize
from   skimage import io
from   sklearn.model_selection import train_test_split
from   sklearn.ensemble import RandomForestClassifier
from   sklearn.metrics import accuracy_score, precision_recall_fscore_support, confusion_matrix, ConfusionMatrixDisplay
```

Après on va lire tous les dossier contient les caractères, on va récupérer liste résultante des noms de fichiers est les stockée dans une variable appelée dos.

```
DATADIR = r"C:/Users/hp/Desktop/SDAD/S3/Machine_Learning/Projet/Phase4/dataset/"
dos= os.listdir(DATADIR)
```

Ensuite, On va lire les images dans chaque dossier de nos étiquettes

```
X = []
y = []
clas = 0

for path, dir, files in os.walk(DATADIR):
    if dir:
        for p in dir :
            for path, dir, files in os.walk(DATADIR + '\\' + p) :
                for f in files :
                    imagePaths = path + '\\' + f
                    img = cv2.imread(imagePaths,2)
                    ret, bw_img = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
                    bw_img = resize(bw_img, (28, 28))
                    X.append(bw_img)

                    label = dos[clas]
                    y.append(label)
                clas = clas + 1

X = np.array(X,dtype=int)
y = np.array(y)
dataset_size = X.shape[0]
X = X.reshape(dataset_size,-1)
```

Dans cet extrait de code, les listes X et y sont remplies avec des données provenant d'une structure de répertoires comportant une série de sous-répertoires, chacun contenant des fichiers image.

Tout d'abord, la fonction os.walk() est utilisée pour parcourir le répertoire de niveau supérieur spécifié par DATADIR. Pour chaque sous-répertoire du répertoire de niveau supérieur, os.walk() est appelée à nouveau pour parcourir les fichiers de ce sous-répertoire.

Pour chaque fichier, le code lit l'image en utilisant cv2.imread(), applique un seuil binaire à l'image en utilisant cv2.threshold() pour créer une image en noir et blanc (binaire), redimensionne l'image en utilisant la fonction resize() , et ajoute l'image à la liste X. L'étiquette correspondante pour l'image est obtenue à partir du dictionnaire dos et l'étiquette est ajoutée à la liste y.

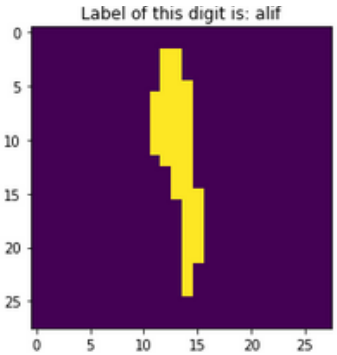
Une fois toutes les images traitées, les listes X et y sont converties en tableaux numériques et la taille de l'ensemble de données est obtenue en obtenant la forme du tableau X. Enfin, le tableau X est remodelé pour avoir la forme (dataset_size, -1). Cela remodèle le tableau afin qu'il ait la forme (dataset_size, num_features), où num_features est le nombre de pixels dans chaque image.

```
print('Overall # of samples is', y.shape[0])
print('Size of the features is:', X.shape)
```

```
Overall # of samples is 7401
Size of the features is: (7401, 784)
```

Cela montre que y contient 7401 observations et X contient 7401 observation avec 784 caractéristiques (28 * 28)

```
def show_digit(x_vec, label):
    x_mat = x_vec.reshape(28, 28)
    plt.figure(figsize=(4,4))
    plt.imshow(x_mat)
    plt.title('Label of this digit is: ' + label)
show_digit(X[300,:], y[300])
```

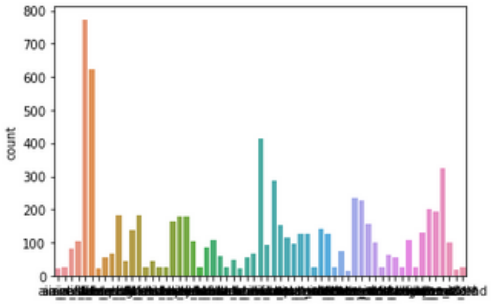


la fonction prend un tableau à une dimension x_vec et une étiquette en entrée et affiche l'image correspondante du chiffre.
La fonction convertit d'abord le tableau à 1 dimension x_vec en un tableau à 2 dimensions à l'aide de la méthode de reshape(), qui réorganise les éléments du tableau dans une nouvelle forme. Dans ce cas, la nouvelle forme comporte 28 lignes et 28 colonnes.

Il est nécessaire de vérifier l'équilibre des classes avant d'appliquer le modèle, a fin d'avoir des résultats performants

```
counter = Counter(y)
print(counter)
```

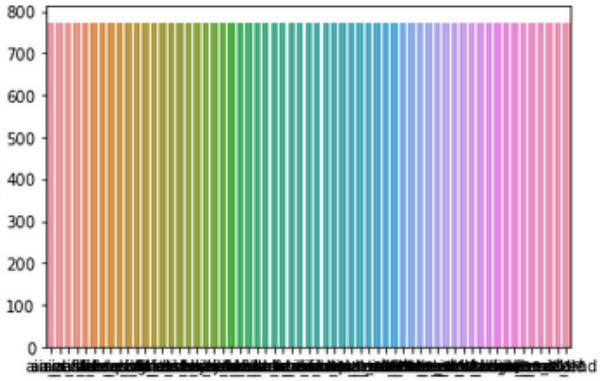
```
Counter({'alif': 773, 'alif_end': 622, 'lam_start': 415, 'yae_middle': 323, 'mim_middle': 288, 'seen_middle': 234, 'seen_star
t': 228, 'waw_end': 203, 'yae_end': 192, 'bae_start': 184, 'dal_end': 184, 'hamza': 179, 'hhae_middle': 177, 'hae_start': 165,
'sheen_middle': 157, 'mim_start': 151, 'rae': 140, 'dal': 137, 'waw': 130, 'noun_start': 126, 'noun_middle': 125, 'rae_end': 12
5, 'noun': 116, 'kaf_middle': 107, 'ta_middle': 106, 'ain_start': 104, 'hhae_start': 104, 'sheen_start': 99, 'yae_start': 99,
'noun_end': 95, 'mim': 92, 'jeem_start': 85, 'ain_middle': 81, 'seen': 76, 'bae_middle': 68, 'lam_middle': 65, 'tae_marbouta':
64, 'kaf_start': 61, 'bae_end': 56, 'lam_end': 55, 'tae_marbouta_end': 54, 'khae_start': 48, 'fa_middle': 45, 'dad_start': 44,
'ain_end': 27, 'fa_end': 27, 'fa_start': 27, 'ghain_middle': 27, 'jeem': 27, 'khae_middle': 27, 'tae_middle': 27, 'thae': 27,
'qua_middle': 26, 'sad_start': 26, 'tae_end': 26, 'zae_end': 25, 'lam_alif': 23, 'ain': 22, 'alif_maqsora': 22, 'zae': 20, 'see
n_end': 13})
```



On peut remarqué que les classes ne sont pas équilibres, Ils doivent donc être équilibrés en utilisant SMOTE :

```
# transform the dataset
oversample = SMOTE()
X, y = oversample.fit_resample(X, y)
```

SMOTE (Synthetic Minority Oversampling Technique) est une technique populaire pour suréchantillonner une classe minoritaire dans un ensemble de données dans le but de former un modèle d'apprentissage automatique. Il fonctionne en générant des échantillons synthétiques de la classe minoritaire, plutôt qu'en dupliquant simplement des échantillons existants comme le font certaines techniques de suréchantillonnage.



Et voilà le résultat, les classes sont maintenant équilibres

Split traditionnelle : nous avons divisé notre jeu de données "arabe" 80% pour l'entrainement et 20% pour le test.

```
# create train test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2  )
print("X_test :",X_test.shape)
print("y_test :",y_test.shape)
print("X_train :",X_train.shape)
print("y_train :",y_train.shape)
```

```
X_test : (9431, 784)
y_test : (9431,)
X_train : (37722, 784)
y_train : (37722,)
```

L'estimation du paramètre n_estimators

```
val_score_train = [] # list pour stocké les vqleurs d'entrainement
val_score_test = [] # list pour stocké les vqleurs de test
k_range = range(999,1010)  # prend range de 999 à 1009
for k in k_range:
    model = RandomForestClassifier(n_estimators=k) # appliquer model random forest
                                                   # avec nbr estimateurs entre 999 et 1009
    model.fit(X_train, y_train) # fit model
    score_train=model.score(X_train,y_train) # calcule accuracy train
    score_test=model.score(X_test, y_test) # calcule accuracy test
    val_score_train.append(score_train) # ajouter résultat en list
    val_score_test.append(score_test)
    # affiche le résultat
    print('ITER ',k,' TRAINING SCORE ',score_train,' TEST SCORE ',score_test)
```

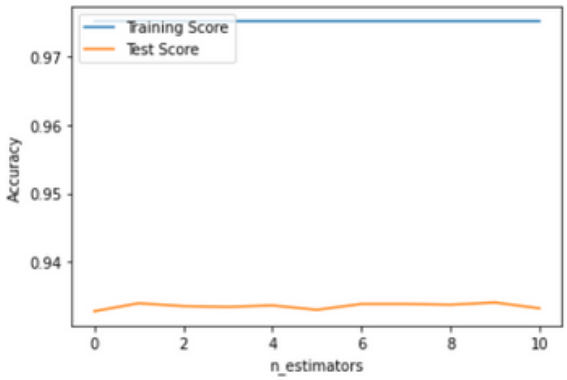
```
ITER  999  TRAINING SCORE  0.9751603838608769  TEST SCORE  0.9327748913158732
ITER  1000  TRAINING SCORE  0.9751603838608769  TEST SCORE  0.9339412575548722
ITER  1001  TRAINING SCORE  0.9751603838608769  TEST SCORE  0.9335171243770544
ITER  1002  TRAINING SCORE  0.9751603838608769  TEST SCORE  0.9334110910825999
ITER  1003  TRAINING SCORE  0.9751603838608769  TEST SCORE  0.9336231576715088
ITER  1004  TRAINING SCORE  0.9751603838608769  TEST SCORE  0.9329869579047821
ITER  1005  TRAINING SCORE  0.9751603838608769  TEST SCORE  0.9338352242604178
ITER  1006  TRAINING SCORE  0.9751603838608769  TEST SCORE  0.9338352242604178
ITER  1007  TRAINING SCORE  0.9751603838608769  TEST SCORE  0.9337291909659633
ITER  1008  TRAINING SCORE  0.9751603838608769  TEST SCORE  0.9340472908493267
ITER  1009  TRAINING SCORE  0.9751603838608769  TEST SCORE  0.933199024493691
```

Dans le modele de random forest on a un hyperparametre n_estimators qui indique le nombre des arbres à créés, et on a affiché la précision dans l'ensemble d'entrainement et l'ensemble de test

Alors nous avons testé notre modele par apport n_estimators pour avoir des bons rédultats dans l' entrainement et dans le test aussi.
Les résultats d'accuracy sont stocké dans des listes.

On peut remarquer dans l 'affichage de boucle et dans le courbe que pour n_estimators 1000 on aura 97% accuracy train et 93% dans de test qui donnera des bonnes résultats.

Le courbe du résultat



L' application de Random Forest avec n_estimators 1000

```
rfc = RandomForestClassifier(n_estimators=1000)
rfc.fit(X_train, y_train)
print('train score:', rfc.score(X_train, y_train))
print('test score:', rfc.score(X_test, y_test))
```

```
train score: 0.9744711309050421
test score: 0.9391368889831407
```

Le code que nous avons fourni entraîne un classificateur de forêt aléatoire sur un ensemble de données d'entraînement (X_train, y_train), puis imprime la précision du modèle sur les ensembles de données d'entraînement et de test.

Le paramètre n_estimators = 1000 détermine le nombre d'arbres de décision que la forêt aléatoire doit contenir. Une valeur plus élevée pour n_estimators se traduira généralement par un modèle plus précis.

Nous testons l'accuracy de modele dans le train et dan le test pour avoir une idée de la façon dont le modèle se généralise aux données invisibles.

```
y_pred=rfc.predict(X_test)
```

rfc. predict(X_test) renvoie une prédiction pour chaque point de données. Dans le cas d'un classificateur de forêt aléatoire, la prédiction sera une étiquette de classe pour chaque point de données dans X_test.

L'Evaluation

Taux d'erreur, Taux de réussite, Précision, Taux de détection, F1-score

```
precision,recall,fscore,none= precision_recall_fscore_support(y_test, y_pred, average='weighted')
print("Taux d'erreur : {}".format(1-accuracy_score(y_test,y_pred)))
print("Taux de réussite : {}".format(accuracy_score(y_test,y_pred)))
print('Précision : '+(str(precision)))
print('Taux de détection : '+(str(recall)))
print('F1-score : '+(str(fscore)))
```

```
Taux d'erreur : 0.06086311101685926
Taux de réussite : 0.9391368889831407
Précision : 0.949831797592206
Taux de détection : 0.9391368889831407
F1-score : 0.941637640480587
```

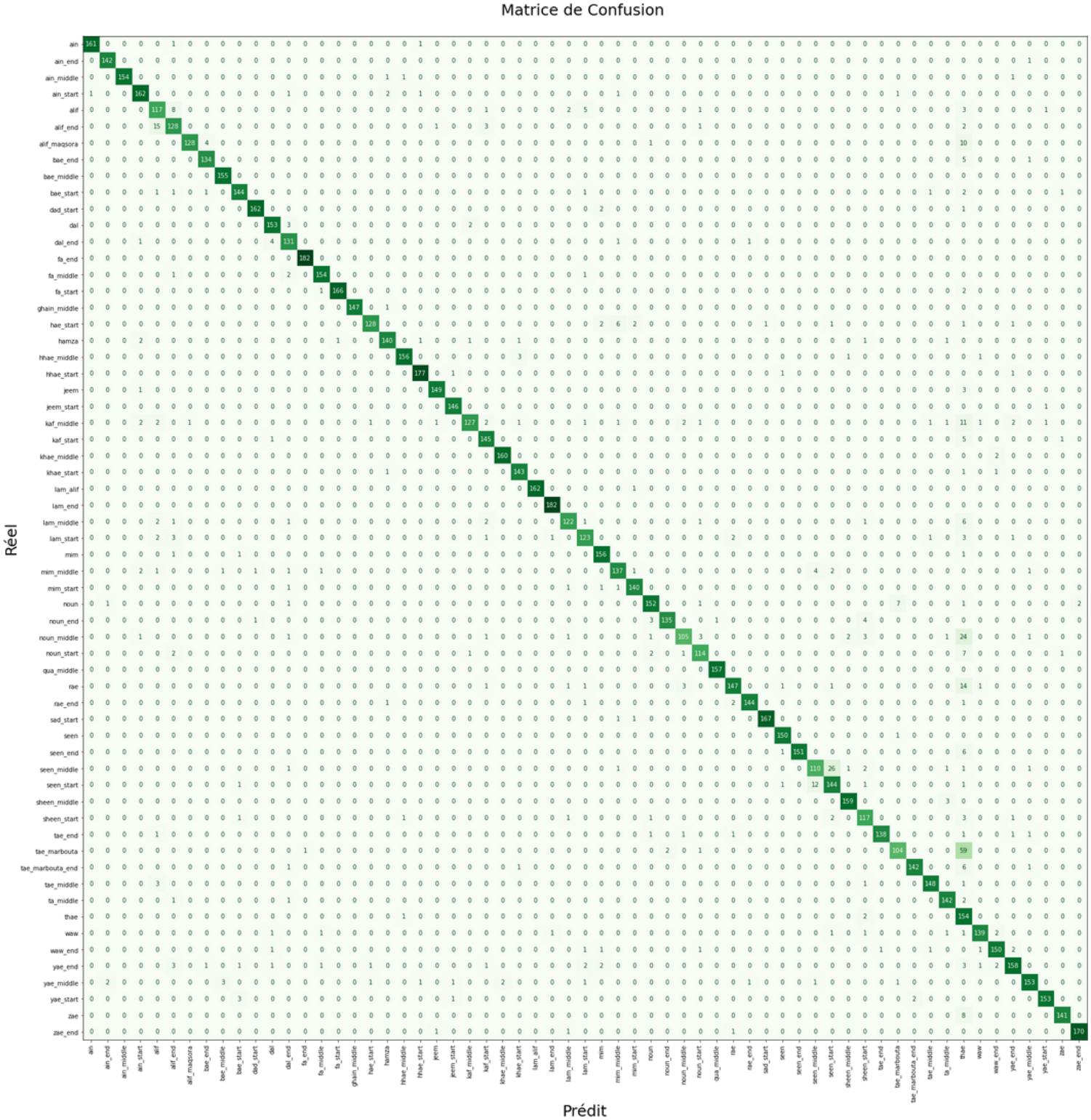
Pour évaluer le modèle Random forest il faut utiliser plusieurs métriques d'évaluation pour évaluer les performances d'un modèle ML

Nous obetenons: Taux erreur 6% , une Accuracy de 94% Précision de 95% , Recall 94% et F1 score 94%

Matrice de Confusion

```
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay
cm=confusion_matrix(y_test,y_pred)

disp=ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=dos)
fig, ax = plt.subplots(figsize=(30,30))
disp.plot(xticks_rotation = 'vertical', ax = ax ,values_format='g', cmap="Greens", colorbar = False)
plt.xlabel("y_pred", fontsize = 25)
plt.ylabel("y_true", fontsize = 25)
plt.xlabel('Prédit', fontsize = 25)
plt.ylabel('Réel', fontsize = 25)
plt.title('Matrice de Confusion \n', fontsize = 25)
plt.show()
```


Matrice de Confusion
Réel
Prédit
ain
ain_end
ain_middle
ain_start
alif
alif_end
alif_maqsora
bae_end
bae_middle
bae_start
dad_start
dal
dal_end
fa_end
fa_middle
fa_start
ghain_middle
hae_start
hamza
hhae_middle
hhae_start
jeem
jeem_start
kaf_middle
kaf_start
khae_middle
khae_start
lam_alif
lam_end
lam_middle
lam_start
mim
mim_middle
mim_start
noun
noun_end
noun_middle
noun_start
qua_middle
rae
rae_end
sad_start
seen
seen_end
seen_middle
seen_start
sheen_middle
sheen_start
tae_end
tae_marbouta
tae_marbouta_end
tae_middle
ta_middle
thae
waw
waw_end
yae_end
yae_middle
yae_start
zae
zae_end