2023/2024

ML : CLASSIFICATION DES IMAGES

RAPPORT DE TP

Zahaf Boualem Nadjib

Rabiai Mehdi Ayoub

description du code :

Ce code démontre efficacement l'application de plusieurs techniques d'apprentissage automatique, y compris les Réseaux de Neurones Convolutionnels (CNN), les K-Plus Proches Voisins (KNN), les Machines à Vecteurs de Support (SVM) et les classificateurs Bayésiens Multinomiaux Naïfs.

1. **Prétraitement des données** : Initialement, le jeu de données MNIST est chargé et prétraité. Cela implique de remodeler les données pour ajouter une dimension de canal, de normaliser les valeurs des pixels et de convertir les étiquettes en vecteurs encodés en one-hot.
2. **Développement du modèle CNN** : Un modèle CNN est construit en utilisant Keras. Ce modèle comprend deux couches convolutionnelles, chacune suivie par un max-pooling, une étape de mise à plat (flattening), une couche entièrement connectée et un dropout pour la régularisation. L'architecture du modèle est adaptée pour traiter et classifier efficacement les images MNIST.
3. **Augmentation des données** : Pour améliorer la capacité de généralisation du modèle, un **ImageDataGenerator** est utilisé pour l'augmentation des données, introduisant des variations telles que la rotation, le zoom et le décalage.
4. **Entraînement et évaluation du CNN** : Le modèle CNN est compilé et entraîné en utilisant l'optimiseur Adam, avec les données augmentées circulant en lots. Après l'entraînement, la précision du modèle est évaluée en utilisant le jeu de données de test.
5. **Implémentation d'algorithmes de ML traditionnels** : En plus du CNN, des algorithmes d'apprentissage machine traditionnels (KNN, SVM et Bayésien Multinomial Naïf) sont mis en œuvre. Ces modèles sont entraînés sur des données d'image aplatis pour faciliter la comparaison avec le CNN.
6. **Évaluation et comparaison des modèles** : La précision de chaque modèle est calculée en utilisant le **accuracy_score** de scikit-learn. Une analyse comparative est réalisée pour évaluer la performance de chaque modèle.
7. **Persistance du modèle** : Enfin, chaque modèle entraîné est sauvegardé sur le disque pour une utilisation future, indiquant une approche pratique de la gestion du pipeline d'apprentissage machine.

resultats du code :

Les résultats obtenus dans cet exercice de machine learning sur le jeu de données MNIST révèlent des performances significativement variées parmi les différents modèles utilisés. Le modèle CNN se distingue nettement avec une précision impressionnante de 99,41 %, soulignant l'efficacité des réseaux de neurones convolutionnels dans le traitement et la classification d'images complexes. En comparaison, le modèle KNN affiche également une bonne performance avec une précision de 97,17 %, indiquant que les méthodes de classification basées sur les voisins peuvent être efficaces, bien qu'inférieures aux techniques d'apprentissage profond. Le modèle SVM suit avec une précision respectable de 94,04 %, ce qui démontre sa compétence dans un contexte de classification d'images,

mais avec une marge d'amélioration par rapport aux méthodes précédentes. Enfin, le modèle Bayésien Multinomial Naïf, avec une précision de 83,57 %, est le moins performant des quatre, ce qui peut être attribué à sa nature simpliste et à son approche moins adaptée aux données d'image. Ces résultats illustrent clairement la supériorité des CNN pour les tâches de classification d'images complexes, tout en mettant en lumière les forces et les faiblesses des différentes méthodologies de machine learning.

Questions :

Création de la Base de Données :

1. **Le jeu de données MNIST** :
    - **Définition** : Le jeu de données MNIST (Modified National Institute of Standards and Technology database) est une collection d'images de chiffres manuscrits, allant de 0 à 9. Il est largement utilisé comme base de référence pour tester les algorithmes de traitement d'images et d'apprentissage automatique.
    - **Usage fréquent** : MNIST est souvent utilisé en apprentissage automatique pour plusieurs raisons. Premièrement, sa simplicité et sa petite taille le rendent idéal pour les tests initiaux et l'enseignement. De plus, comme il s'agit d'un problème relativement simple mais représentatif de tâches de classification d'images plus complexes, il sert de "terrain d'essai" pour de nombreuses méthodes en apprentissage profond.
2. **Format des données dans x_train et x_test** :
    - **Nature des données** : Dans le jeu de données MNIST, **x_train** et **x_test** contiennent respectivement les images d'entraînement et de test. Chaque image est représentée sous forme de tableau 2D (matrice) de pixels.
    - **Représentation d'une image** : Chaque image dans ces variables est représentée par une matrice de pixels de 28x28. Chaque élément de cette matrice correspond à l'intensité d'un pixel en niveaux de gris, allant de 0 (noir) à 255 (blanc).
3. **Variables y_train et y_test** :
    - **Rôle** : Les variables **y_train** et **y_test** représentent les étiquettes (labels) des images dans **x_train** et **x_test**. Elles contiennent les chiffres réels que chaque image est censée représenter.
    - **Encodage des étiquettes** : Dans MNIST, les étiquettes sont encodées sous forme de nombres entiers allant de 0 à 9, correspondant directement au chiffre manuscrit dans l'image.
4. **Dimensions de x_train et x_test** :
    - **Dimensions** : La dimension de **x_train** est typiquement de [60000, 28, 28], représentant 60 000 images de 28x28 pixels chacune. Pour **x_test**, la dimension est généralement de [10000, 28, 28], correspondant à 10 000 images de test.
    - **Correspondance avec la nature des images** : Ces dimensions correspondent à la quantité d'images dans le jeu de données et à la résolution de chaque image.

Chaque image étant un carré de 28x28 pixels, la dimension reflète cette structure en deux dimensions.

Chargement des Données :

1. **Importance du Chargement Correct des Données** :
    - **Fondation pour l'entraînement** : Le chargement correct des données est crucial car il constitue la base de tout le processus d'apprentissage automatique. Si les données ne sont pas correctement chargées, toutes les étapes ultérieures, y compris l'entraînement du modèle, seront impactées négativement.
    - **Qualité des données** : Une bonne opération de chargement assure que la qualité des données est maintenue, évitant ainsi des problèmes tels que les données corrompues, manquantes ou mal formatées, qui peuvent conduire à des résultats d'apprentissage erronés.
2. **Visualisation des Données et Compréhension** :
    - **Méthodes de visualisation** : Pour visualiser des exemples d'images des ensembles d'entraînement et de test, on peut utiliser des bibliothèques comme Matplotlib en Python. Un simple script permettrait d'afficher un échantillon d'images et leurs étiquettes correspondantes.
    - **Importance** : Comprendre les données avant de construire un modèle est essentiel. La visualisation aide à identifier les tendances, les anomalies, la variabilité des données, et aussi à confirmer que les données sont correctement chargées et formatées.
3. **Prétraitement des Données MNIST** :
    - **Étapes courantes** :
        - **Normalisation** : Transformer les valeurs de pixels pour qu'elles soient comprises entre 0 et 1. Cela aide à accélérer l'entraînement en normalisant l'échelle des données d'entrée.
        - **Redimensionnement** : Adapter les données pour qu'elles correspondent à l'architecture d'entrée du modèle (par exemple, remodeler les images en vecteurs si nécessaire).
        - **Encodage One-hot** : Transformer les étiquettes en un format adapté pour la classification (ex. : transformation des étiquettes de chiffres en vecteurs one-hot pour la classification multi-classes).
    - **Importance** : Le prétraitement est crucial car il conditionne les données à être mieux adaptées aux exigences et à la sensibilité des algorithmes de machine learning. Cela

augmente l'efficacité de l'entraînement et améliore potentiellement la précision du modèle.

Concernant la fonction **load_data**, elle provient généralement de bibliothèques spécialisées telles que TensorFlow ou Keras, dans le cadre de leur module de jeux de données. Cette fonction peut accepter divers paramètres tels que le chemin de la source des données, la configuration du format de données (par exemple, si les images doivent être retournées en tant que vecteurs ou matrices 2D), ou des paramètres spécifiques au jeu de données comme la fraction des données à utiliser pour l'entraînement vs. le test. L'existence et la nature exacte de ces paramètres supplémentaires dépendent de l'implémentation spécifique de la fonction dans la bibliothèque utilisée.

Extraction des Caractéristiques :

1. **Normalisation des Pixels dans la Plage [0, 1]** :
    - **Objectif** : La normalisation des valeurs des pixels dans la plage [0, 1] est réalisée pour standardiser les entrées du modèle. Cette approche réduit la disparité des échelles de caractéristiques, améliorant ainsi l'efficacité de l'entraînement.
    - **Avantages** : Cette normalisation permet d'obtenir une convergence plus rapide lors de l'entraînement, car elle réduit les variations initiales entre les poids du modèle. De plus, cela aide à éviter des problèmes tels que le gradient explosif, particulièrement dans les réseaux de neurones profonds.
2. **Conversion en Type 'float32'** :
    - **Raison** : Les données sont converties en 'float32' pour permettre des calculs plus précis, en particulier pour les opérations de gradient pendant l'entraînement. Les types de données à virgule flottante offrent une plus grande précision par rapport aux entiers.
    - **Différence avec le Type d'Origine** : Le type de données d'origine est souvent un entier (par exemple, 0-255 pour les pixels en niveaux de gris). La conversion en 'float32' permet des opérations sur des valeurs fractionnaires, ce qui est essentiel pour la normalisation et divers calculs pendant l'entraînement du modèle.
3. **Division par 255.0** :
    - **Signification de 255.0** : La valeur 255.0 représente le maximum de l'intensité d'un pixel en niveaux de gris (dans une échelle allant de 0 à 255). En divisant par cette valeur, on normalise chaque pixel pour qu'il soit compris entre 0 et 1.
    - **Raison de la Division** : La division par 255.0 est une manière simple et efficace de normaliser les données, garantissant que tous les pixels auront des valeurs comprises dans une plage standard, facilitant ainsi le traitement par le modèle.
4. **Impact de la Normalisation sur l'Entraînement** :
    - **Stabilité et Convergence** : La normalisation des pixels contribue à améliorer la stabilité et la convergence de l'entraînement du modèle. En normalisant les caractéristiques d'entrée, on aide à garantir que les gradients ne deviennent pas trop

grands ou trop petits, évitant ainsi les problèmes de disparité dans les mises à jour des poids durant l'apprentissage.

5. **Utilisation de 255 vs 255.0** :
    - **Différence d'Utilisation** : Utiliser 255 (entier) pourrait conduire à une division entière, résultant en des valeurs normalisées incorrectes (soit 0 soit 1), tandis que 255.0 (flottant) assure une division flottante, produisant des valeurs normalisées précises entre 0 et 1.
    - **Conséquences sur la Normalisation** : L'utilisation d'un entier au lieu d'un flottant dans l'opération de normalisation pourrait affecter négativement la précision des données normalisées, et par conséquent impacter l'efficacité et la précision du modèle d'apprentissage automatique.

Division des données :

1. **Forme Originale de x_test et Impact du Remodelage** :
    - **Forme Originale** : Avant le remodelage, **x_test** est généralement sous la forme d'un tableau 4D (pour les images RGB) ou 3D (pour les images en niveaux de gris). Par exemple, pour MNIST, la forme serait [nombre d'images, hauteur, largeur] ou [nombre d'images, hauteur, largeur, canaux].
    - **Impact du Remodelage** : Le remodelage modifie la structure des données pour les rendre compatibles avec les exigences d'entrée du modèle. Pour un modèle de réseau de neurones entièrement connecté, les images 2D sont généralement aplaties en vecteurs 1D.
2. **Rôle du Paramètre "-1" dans le Remodelage** :
    - **Interprétation** : Le paramètre "-1" utilisé dans l'opération de **reshape** permet à la fonction de calculer automatiquement la taille de cette dimension, en se basant sur la taille totale des données et les dimensions spécifiées. C'est une manière pratique de spécifier une dimension "flexible" dont la taille est déduite des autres dimensions.
3. **Forme Résultante de x_test_flat et Importance pour le Modèle** :
    - **Description de la Forme Résultante** : Après le remodelage, **x_test_flat** a généralement une forme de [nombre d'images, caractéristiques], où "caractéristiques" correspond au nombre total de pixels dans chaque image (par exemple, pour des images 28x28, cela serait 784).
    - **Importance** : Cette forme est importante pour les modèles, notamment les réseaux de neurones entièrement connectés, car ils nécessitent des vecteurs d'entrée unidimensionnels. Le remodelage permet de transformer des images 2D ou 3D en vecteurs 1D qui peuvent être traités par le réseau.
4. **Correspondance avec les Exigences du Modèle** :
    - **Exigences du Modèle** : Les modèles de machine learning, en particulier les réseaux neuronaux, ont souvent des spécifications précises pour la forme des données

d'entrée. Par exemple, un réseau de neurones entièrement connecté attend des vecteurs unidimensionnels en entrée.

- **Adéquation avec la Conception du Réseau Neuronal** : En remodelant **x_test** en **x_test_flat**, on adapte les données pour qu'elles correspondent à ces exigences. Cette étape garantit que chaque image est présentée au modèle sous la forme d'un vecteur de caractéristiques, permettant au réseau de traiter efficacement chaque image individuellement.

Création et Entraînement du Modèle :

1.Réseaux de Neurons :

1. **l'architecture générale du modèle :**

ce modèle utilise des couches de convolution pour l'extraction de caractéristiques, suivies de couches de pooling pour réduire la dimensionnalité. Les caractéristiques aplaties sont ensuite traitées par des couches denses pour la classification. La couche de dropout est utilisée pour réduire le surapprentissage, et la dernière couche dense avec activation 'softmax' est utilisée pour la classification finale.

1.1 les couches :

Couche d'Entrée (Input) :

- **Fonction** : Définit la forme d'entrée du modèle, qui correspond aux dimensions des images d'entrée (par exemple, 28x28x1 pour les images en niveaux de gris de MNIST).

Première Couche Convolutionnelle (Conv2D avec 32 filtres de taille 3x3) :

- **Rôle** : Applique 32 filtres convolutifs de taille 3x3 sur l'entrée. Chaque filtre extrait des caractéristiques spécifiques (comme les bords, les textures) de l'entrée.
- **Activation 'relu'** : La fonction 'relu' ajoute une non-linéarité, permettant au modèle de capturer des relations complexes dans les données.

Première Couche de Pooling (MaxPooling2D avec une taille de pool de 2x2) :

- **Fonction** : Réduit la dimension spatiale (hauteur et largeur) de la sortie de la couche précédente. Cela diminue la complexité computationnelle et le risque de surapprentissage.

Deuxième Couche Convolutionnelle (Conv2D avec 64 filtres de taille 3x3) :

- **Rôle** : Augmente la profondeur de l'apprentissage des caractéristiques en appliquant 64 filtres. Cela permet de détecter des caractéristiques plus complexes.

Deuxième Couche de Pooling (MaxPooling2D avec une taille de pool de 2x2) :

- **Fonction** : Continue de réduire la dimension spatiale, consolidant ainsi les caractéristiques apprises et réduisant encore la complexité.

Couche d'Aplatissement (Flatten) :

- **Rôle** : Transforme la sortie 2D des couches de pooling en un vecteur 1D. Ceci est nécessaire pour alimenter les données dans les couches denses suivantes.

Couche Dense (Dense avec 128 neurones) :

- **Fonction** : Une couche entièrement connectée qui apprend des relations non linéaires à partir des caractéristiques aplaties.
- **Activation 'relu'** : Permet d'introduire des non-linéarités dans l'apprentissage.

Couche de Dropout (Dropout avec un taux de 0.5) :

- **But** : Aide à prévenir le surapprentissage en "éteignant" aléatoirement 50% des neurones durant l'entraînement, forçant ainsi le modèle à apprendre des caractéristiques robustes.

Couche de Sortie (Dense avec num_classes neurones) :

- **Fonction** : Calcule les scores de classification pour chaque classe.
- **Activation 'softmax'** : Convertit les scores en probabilités, où chaque valeur est la probabilité que l'entrée appartienne à une des classes.

3. **Forme de l'Entrée et Couche Flatten** :
    - La **input_shape** est définie selon les dimensions des images d'entrée. La couche **Flatten** nécessite cette définition pour convertir les matrices de caractéristiques en vecteurs.
4. **Choix de 128 Neurones dans la Couche Dense** :
    - Le choix de 128 neurones offre un bon équilibre entre la capacité d'apprentissage et la complexité computationnelle, permettant au modèle de capturer des relations suffisamment complexes sans être trop coûteux en termes de calcul.
5. **Utilisation de 'relu' dans la Première Couche Dense** :
    - 'relu' aide à résoudre le problème de disparition de gradient et accélère la convergence pendant l'entraînement. Contrairement à d'autres fonctions comme 'sigmoid' ou 'tanh', 'relu' permet d'activer les neurones de manière plus efficace.
6. **Rôle de la Dernière Couche Dense avec 'softmax'** :
    - Cette couche sert à la classification multi-classes. Le 'softmax' convertit les scores en probabilités pour chaque classe, correspondant directement au problème de classification d'images (par exemple, 10 classes pour MNIST).

7. **Performance du Classifieur Naive Bayes Multinomial** :

La comparaison des performances du classifieur Naïf Bayes Multinomial avec d'autres modèles de classification dans scikit-learn, basée sur les résultats obtenus, révèle des différences notables en termes d'efficacité et d'applicabilité.

Performances Comparatives:

Naïf Bayes Multinomial (NBM): Avec une précision de 83,57%, le NBM se positionne comme le moins performant parmi les modèles testés. Bien que ce soit une méthode rapide et simple, sa performance est limitée, particulièrement dans les tâches complexes comme la classification d'images.

K-Plus Proches Voisins (KNN): Le KNN a démontré une précision considérablement plus élevée de 97,17%. Ce modèle, bien qu'étant plus complexe que le NBM, offre une meilleure capacité de généralisation sur les données d'images.

Machines à Vecteurs de Support (SVM): Avec une précision de 94,04%, le SVM surpasse également le NBM. Le SVM est plus adapté pour gérer les caractéristiques non linéaires et complexes des données d'image.

Réseaux de Neurones Convolutionnels (CNN): Le CNN, avec une précision de 99,41%, est le plus performant. Il illustre l'efficacité des techniques d'apprentissage profond dans la classification d'images, surpassant largement le NBM.

Avantages et Inconvénients:

Avantages du NBM:

Simplicité: Facile à implémenter et à comprendre.

Vitesse: Rapide à entraîner, idéal pour des ensembles de données de taille modérée.

Efficacité sur des données textuelles: Particulièrement performant sur des données catégorielles ou textuelles.

Inconvénients du NBM:

Performances limitées sur les images: Moins efficace pour la classification d'images en raison de l'hypothèse de l'indépendance des caractéristiques, qui est rarement le cas dans les données d'image.

Sensibilité à la représentation des données: La performance dépend fortement de la façon dont les données sont présentées au modèle.

En conclusion, bien que le classifieur Naïf Bayes Multinomial soit un outil utile dans certaines circonstances, notamment pour sa simplicité et sa rapidité, il est surpassé en termes de précision et de capacité de généralisation par d'autres modèles disponibles dans scikit-learn, en particulier lorsqu'il s'agit de la classification d'images complexes.

8. **Choix d'une Architecture de Réseau de Neurones** :
    - L'architecture présentée dans le code (CNN) est bien adaptée pour la classification d'images, combinant la capacité d'extraction de caractéristiques des couches convolutionnelles avec la classification des couches denses.
9. **Entraînement pour 5 Époques** :
    - L'entraînement sur 5 époques est un choix initial pour évaluer les performances. Le nombre optimal d'époques est généralement déterminé par la surveillance de la performance sur un ensemble de validation pour éviter le surapprentissage.
10. **Signification de 'batch_size'** :

- 'batch_size' de 32 est un choix équilibré qui permet une convergence rapide tout en étant efficace en termes de mémoire. Il détermine le nombre d'échantillons traités avant la mise à jour des poids du modèle.

11. **Utilisation de 'validation_split'** :

- Le **validation_split** permet de séparer une partie des données d'entraînement pour la validation. Cela aide à surveiller et à prévenir le surapprentissage pendant l'entraînement.

12. **Sauvegarde du Modèle Entraîné** :

- Le modèle peut être sauvegardé via des fonctions comme **model.save()** pour une utilisation ultérieure sans nécessiter de réentraînement.

13. **Définition du Nombre de Classes** :

- Le nombre de classes (**num_classes**) doit correspondre au nombre total de catégories distinctes dans l'ensemble de données. Dans not cas **num_classes** est 10

2. KNN :

1. **Adaptation du Modèle KNN aux Données d'Entraînement et Rôle des Voisins** :
    - **Adaptation** : Le modèle KNN ne s'adapte pas activement pendant l'entraînement comme le font d'autres algorithmes d'apprentissage. Au lieu de cela, il mémorise l'ensemble de données d'entraînement.
    - **Rôle des Voisins** : Dans la classification KNN, la décision est prise en fonction des 'k' voisins les plus proches d'un point de données inconnu. Le modèle prédit la classe en se basant sur la classe majoritaire parmi ces voisins.
2. **Aplatissage des Données d'Entraînement pour KNN** :
    - **Raison de l'Aplatissage** : Les données sont aplaties pour convertir les images 2D ou 3D en vecteurs 1D, car KNN compare la distance entre les points de données dans un espace de caractéristiques. Les images doivent être représentées sous forme de vecteurs pour calculer ces distances.

- **Impact sur la Performance** : L'aplatissement peut parfois entraîner une perte de l'information spatiale. Toutefois, pour de nombreux ensembles de données, cette représentation reste efficace pour la classification avec KNN.

3. **Influence du Choix de 'k' sur la Précision de KNN** :
    - **Impact sur la Précision** : Un 'k' trop petit peut rendre le modèle sensible au bruit, tandis qu'un 'k' trop grand peut lisser les frontières de classification et réduire la précision.
    - **Compromis** : Il existe un compromis entre précision et complexité. Un 'k' plus élevé augmente la complexité du calcul, mais peut améliorer la précision jusqu'à un certain point.
4. **Détermination Empirique du Meilleur Nombre de Voisins** :
    - **Méthodologie** : Pour déterminer le meilleur 'k', utilisez des méthodes empiriques telles que la validation croisée. Testez plusieurs valeurs de 'k' et évaluez les performances du modèle sur un ensemble de validation.
    - **Critères d'Évaluation** : Choisissez le 'k' qui offre le meilleur équilibre entre la précision des prédictions et la complexité computationnelle, en tenant compte de la spécificité des données.

Impacte d'hybridation des modèles des données sur la classification des images :

1. **Utilisation de np.vstack pour l'Empilement Vertical des Prédictions** :
    - **But** : **np.vstack** est utilisé pour empiler verticalement les prédictions des deux modèles (par exemple, réseau de neurones et KNN). Cela crée une nouvelle structure de données où chaque ligne représente les prédictions des différents modèles pour la même instance de données.
    - **Facilitation de la Combinaison** : Cette structuration facilite la comparaison des prédictions de chaque modèle côte à côte, permettant des opérations ultérieures comme le vote majoritaire ou d'autres formes de combinaison.
2. **Fonctionnement du Vote Majoritaire** :
    - **Mécanisme** : Dans le vote majoritaire, pour chaque instance de données, on considère les prédictions de tous les modèles. La classe qui reçoit le plus de votes (c'est-à-dire est choisie le plus souvent par les différents modèles) est considérée comme la prédiction finale.

- **Prédiction Finale** : La prédiction finale pour chaque instance est la classe qui a obtenu le plus grand nombre de votes parmi tous les modèles.

3. **Différences entre les Prédictions de NN et KNN** :
    - **Nature des Prédictions** : Les réseaux de neurones (NN) et KNN peuvent avoir des manières différentes d'interpréter les données, en raison de leurs mécanismes d'apprentissage distincts. Par exemple, les NN sont bons pour capturer des relations complexes et non linéaires, tandis que KNN se base sur la proximité locale.
    - **Exploitation de la Combinaison** : En combinant ces modèles, on cherche à exploiter les forces de chaque modèle. Par exemple, la capacité des NN à apprendre des représentations profondes et celle de KNN à prendre des décisions basées sur des exemples proches.
4. **Gestion des Prédictions Égales dans la Combinaison par Vote Majoritaire** :
    - **Cas des Prédictions Égales** : Si les prédictions des deux modèles sont identiques pour une instance donnée, cela renforce la confiance dans cette prédiction. Dans un vote majoritaire, cette prédiction reçoit alors tous les votes pour cette instance.
    - **Gestion dans le Vote Majoritaire** : Si les deux modèles s'accordent sur la même classe, cette classe est automatiquement sélectionnée comme prédiction finale pour cette instance, car elle a obtenu la majorité (ou l'unanimité) des votes.

L'hybridation de modèles tels que NN et KNN peut donc offrir une approche plus équilibrée et robuste pour la classification d'images, tirant parti des forces uniques de chaque type de modèle.

2023-2024

TP 3

MERCI

Zahaf Boualem Nadjib
Rabiai Mehdi Ayoub