Enseignantes : **A. NAJJAR-** **I. BEN OTHMEN – M. FARHAT** **TP : I. HAMROUNI- S.BIROUZA-** **F. JENHANI**	**T4** **Machine Learning**	**Classe** **3ème GLSI**

Exercice 1

On souhaite effectuer de la classification en utilisant la méthode des **K-plus proches voisins**. Dans ce TP, on se basera sur la base "**IRIS"** du package **sklearn**.**datasets**.

1- Charger les données.
   a. Créer une matrice qui ne contient que les données des attributs (séparer les valeurs des attributs de leurs classes d'appartenances).
   b. Créer un vecteur qui ne contient que les labels des classes.
2- Afficher les noms des variables et en déduire leur nombre.
3- Afficher les classes. En déduire le type de classification auquel on s'intéresse.
4- Normaliser les valeurs des variables pour qu'elles suivent une loi normale de moyenne 0 et de variance 1.
5- Ecrire une fonction qui permet de faire la classification avec l'algorithme des k-plus proches voisins et afficher le taux d'erreur obtenu sur l'ensemble de test. Cette fonction doit prendre en paramètre les données d'apprentissage, les données de test et la valeur de K.
6- Diviser l'ensemble de données en un ensemble d'apprentissage qui contient 60% des observations et un ensemble de test (40%).
7- Effectuer la classification des données en considérant différentes valeurs de k.
8- Utiliser la validation croisée pour déterminer la valeur optimale de k puis refaire la classification en utilisant cette valeur.
9- Afficher la matrice de confusion et interpréter le résultat obtenu.
10- Déterminer les valeurs du rappel et de la précision. Interprétez.

Exercice 2

Dans cet exercice, on va s'intéresser à une base très célèbre, appelé MNIST. Il est constitué d'un ensemble de **70000** images **28x28** pixels en noir et blanc annotées du chiffre correspondant (entre 0 et 9). L'objectif de ce jeu de données était de permettre à un ordinateur d'apprendre à reconnaître des nombres manuscrits automatiquement (pour lire

des chèques par exemple). Cette base utilise des données réelles qui ont déjà été **pré-traitées** pour être plus facilement utilisables par un algorithme. Des exemples d'images de cette base sont donnés sur la Figure 1.

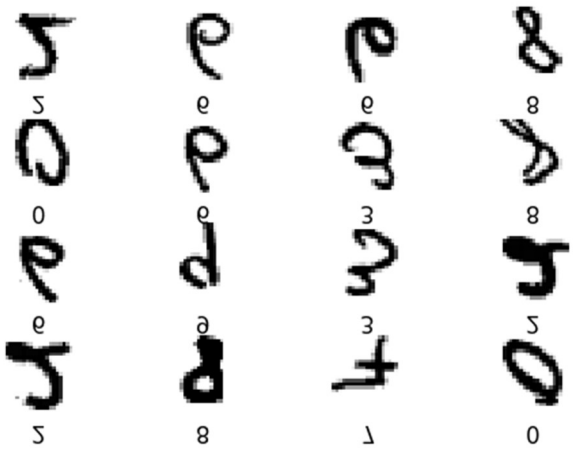

Figure 1

1. Charger la base à partir de openml.org à l'aide de la bibliothèque Sklearn.datasets.

Documentez vous autour de la fonction '***fetch_openml***' à travers ce lien :
https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_openml.html
Le nom de la dataset est : 'mnist_784'

2. L'objet mnist contient deux entrée principales, data et target. On peut les afficher :

```
# Le dataset principal qui contient toutes les images
print (mnist.data.shape)

# Le vecteur d'annotations associé au dataset (nombre entre 0 et 9)
print (mnist.target.shape)
```

- **Data** : contient les images sous forme de tableaux de 28 x 28 = 784 couleurs de pixel en niveau de gris, c'est-à-dire que la couleur de chaque pixel est représentée par un nombre entre 0 et 16 qui représente si celle-ci est proche du noir ou pas (0 = blanc, 16 = noir).
- **Target** : qui contient les annotations (de 1 à 9) correspondant à la valeur "lue" du chiffre.

On dit ici que le nombre d'attributs/Features (ou dimension) en entrée est de 28 x 28 x 1 = 784. Dans le cas où l'on aurait utilisé des images en couleurs et pas en niveaux de gris, on serait passé à 3 composantes couleurs par pixel (rouge, vert, bleu) et donc le nombre d'attributs aurait été : 28 x 28 x 3 = 2352.

La base est relativement petite mais, pour le modèle k-NN, il est déjà trop gros pour obtenir rapidement des résultats.

3. Effectuer un échantillonnage (sampling) et travailler sur seulement 5000 données. Utiliser la méthode : ***dataframe.sample*** (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.sample.html)

4. Séparer le jeu de données en training set (80%) et testing set (20%)

5. Effectuer un apprentissage en utilisant l'algorithme K-NN (K-plus proche voisins) en s'appuyant sur la méthode ***KNeighborsClassifier*** et en choisissant K= 1. Quelle est la distance utilisée dans ce classifieur ?
(https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html).

6. Afficher le taux d'erreur en appliquant le k-NN sur l'ensemble de jeu de données de test pour une valeur de K=1.

7. Ré-exécuter les instructions (4-5-6), avez-vous le même résultat ? pourquoi ?

8. Faites varier le k de 1 jusqu'à 15, appliquer le k-NN et utiliser une variable (de type array) **ERRORS** qui stocke les valeurs d'erreurs pour chaque valeur de K.

9. Afficher la courbe de l'évolution de l'erreur en fonction de K.

10. Sélectionner la valeur de K la plus performante. Et appliquer le classifieur (K-NN) avec la valeur K trouvée.

11. Récupérer les prédictions sur les données de test dans un vecteur **PREDICTED**.

12. Utiliser le code suivant pour afficher les images avec les prédictions associées dans 11.

```
# On redimensionne les données sous forme d'images
images = xtest.values.reshape((-1, 28, 28))
# On sélectionne un échantillon de 12 images au hasard
select = np.random.randint(images.shape[0], size=12)
# On affiche les images avec la prédiction associée
fig,ax = plt.subplots(3,4)
for index, value in enumerate(select):
    plt.subplot(3,4,index+1)
    plt.axis('off')
    plt.imshow(images[value],cmap=plt.cm.gray_r,interpolation="nearest")
    plt.title('Predicted: {}'.format( predicted[value]) )
plt.show()
```

13. Augmenter la taille de l'ensemble de données utilisée dans 3 comme suit (10000, 20000, 40000) et refaire les calculs d'erreur pour la valeur de k trouvée dans 10. Que remarquez-vous ?

    On souhaite répéter la prédiction mais en utilisant la distance de **manhattan**. Pour cela, commencer par définir une fonction ***manhattan*** qui prend comme paramètres deux vecteurs, et renvoie un scalaire. Puis, refaire les prédictions avec cette nouvelle métrique en utilisant la fonction ***KNeighborsClassifier*** et le k trouvé dans 10. Qu'est-ce que vous remarquez ?

/. Bon Travail./