Mini PROJET 7

Exemples d'apprentissages supervisés *(Machine learning)* grâce à un algorithme de classification de type kNN

Dans ce projet il faudra choisir un problème et tenter d'y répondre à l'aide de l'algorithme des kNN. Du plus simple au plus complexe :

→ *Déterminer si un individu devra suivre un traitement ou non contre le diabète.*

→ *Déterminer si un individu embarqué à bord du Titanic sera mort ou vivant à l'issue du naufrage.*

→ *Reconstruire une photo abîmée.*

TRAVAIL DEMANDE :

Dans chacun des cas, il faudra fournir :

Au minimum

- Un programme python répondant à la question
- Un document texte donnant la description précise du *dataset* (attributs, formats des valeurs, ...etc), la *distance* utilisée et les résultats obtenus pour différentes valeurs de *k*.

En plus

- Différents résultats avec différentes *distances*.
- Une représentation graphique des *data-train*.
- Une interface graphique de saisie pour d'autres *data-test*.

Le projet sera à rendre en ligne dans un dossier zippé le **lundi 09/05.**
Le travail se fera par groupe de 2 maximum.

CRITERES EVALUATION :

		Critères observés
ANA **/6**	*Décrire et Spécifier les caractéristiques d'un processus et les données d'un problème (algo kNN)*	• Préparation et/ou vérification des données pour qu'elles soient exploitables (type, validité, …) • Fonction de calcul de distance • Fonction de recherche des voisins
REA **/8**	*Imaginer et concevoir une solution, décomposer en blocs, se ramener à des sous-problèmes simples et indépendants.*	• Découpage en fonctions pertinentes • Documentation de fonctions (docstrings et commentaires) • Calcul de distance implémenté • Possibilité de changer la valeur de k
COM **/8**	*Communiquer à l'écrit dans un langage précis et rigoureux*	• Description complète des données utilisées • Présentation de la (les) distance(s) utilisée(s) • Présentation des résultats obtenus pour différentes valeurs de k • Vocabulaire rigoureux et précis

1. Problème N°1 : Diabète

On dispose d'un fichier **diabete.csv** présentant 8 colonnes de diverses mesures (insuline, âge, ...). Le séparateur de données est la virgule.

La 9ème colonne présente des 0 et des 1. Il s'agit des deux classes possibles pour ce problème :

- 1 signifie que la personne suit un traitement pour des problèmes liés au diabète,
- 0 qu'elle n'en suit pas.

Un nouveau patient est caractérisé par les mesures suivantes (il s'agit, dans l'ordre, des données correspondant aux 8 premières colonnes) :

```
nouveauPatient = [1, 89, 67, 24, 80, 23, 0.6, 65]
```

→ ***Ce nouveau patient devra-t-il suivre un traitement ?***

2. Problème N°2 : Titanic

Le jeu de données **titanic.csv** contient des informations sur une partie des passagers (891 passagers) du Titanic. Pour un petit rappel historique, on peut consulter la page Wikipédia consacrée à ce paquebot ici : wiki

→ ***Le passager ci-dessous aurait-il survécu au naufrage du Titanic ?***

PCLASS	NAME	SEX	AGE	SIBSP	PARCH	TICKET	FARE	EMBARKED
2	Norris Chuck	male	33	1	4	244377	21.075	C

AIDE :

- Toutes les colonnes ne vont pas forcement être pertinentes, par exemple, lors du naufrage, le nom du passager n'a pas eu une quelconque importance sur le fait qu'il ait ou non survécu.

- Pour certains passagers, il manque des données. Par exemple, l'âge de certains passagers n'est pas renseigné. On pourra supprimer du fichier les passagers ayant des données incomplètes.

- L'utilisation de l'algorithme des k plus proches voisins nous oblige à proscrire les données non numériques. Par exemple, la colonne « Sex » ne peut pas être utilisée telle quelle, l'algorithme n'est pas capable de traiter les « male » et « female ». Il faut trouver une alternative pour remplacer les chaînes de caracteres « male » et « female ».

- Pour les calculs de distances, il faut faire en sorte que les amplitudes des valeurs des différentes colonnes soient dans le même ordre de grandeur de façon à ce que les distances calculées puissent être comparables entre elles. (par exemple, l'amplitude des colonnes « Pclass » et « Age » ne sont pas les mêmes).
  Les modifications à faire pour chaque colonnes concernées sont :
  1. On repère la valeur minimale **v_min** et la valeur maximale **v_max**
  2. On va diviser chacune des valeurs de la colonne par la diffrence **v_max – v_min**

Par exemple : Si une colonne contient les valeurs [5, 4, 1, 11, 7], on a v_min = 1 et v_max = 11. Alors on divise toutes les valeurs par 8 ce qui donne [0.5, 0.4, 0.1, 1.1, 0.7]

3. Problème N°3 : Retouche photo

Cette photo **bob.png** a subi une dégradation : un gros carré rouge dans l'oeil !
On dispose du fichier **image.py** pour étudier rapidement la photo.

On rappelle que :

- Chaque pixel est repéré dans l'image par ses coordonnées : le pixel (0,0) est le coin supérieur gauche et le pixel (largeur-1, hauteur-1) est le coin inférieur droit.
- Les colonnes sont numérotées dans l'ordre croissant de gauche à droite, les lignes sont numérotées dans l'ordre croissant de haut en bas.
- Chaque pixel a une couleur de la forme (composante rouge, composante vert, composante bleu). Chaque composante est un entier entre 0 et 255.

La zone des pixels dégradés est située :

- lignes entre 30 et 100,
- colonnes entre 250 et 320.

Les pixels rouge qui dégradent la photo ont pour couleur (255, 0, 0). Il n'y aucun autre point de la zone de la photo ayant cette couleur.

→ Remplacer la partie dégradée de l'image et produire une autre image « corrigée ».

AIDE :

- On peut créer et utiliser une fonction `distance` qui prend en paramètres les couples de coordonnées de deux pixels et renvoie la distance euclidienne entre ces deux pixels.

- Pour reconstituer la photo, on souhaite remplacer la couleur (255, 0, 0) d'un pixel dégradé par la moyenne des couleurs de ses k voisins les plus proches.
  On pourra donc s'aider d'une fonction `moyenne` qui renvoie la moyenne des couleurs des k pixels non dégradés plus proches.

- Pour tester et visualiser si le résultat est satisfaisant

```
img.save("image_reparee.png")
```