1. ClassifierKnn.cs — Le cœur du modèle KNN

Cette classe implémente ton algorithme de classification k-Nearest Neighbors.

- ✓ Attributs
- IDistance distance → stratégie de calcul (Euclidienne ou Manhattan)
- EnsembleDonnes donneesApprentissage→ les données d'entraînement
- int k→ nombre de voisins à considérer
- ✓ Constructeur

public ClassifierKnn(int k, IDistance distance)

Initialise le classifieur avec un k et une distance.

- ✓ Entrainer()

public void Entrainer(EnsembleDonnees data)

Stocke les données d'apprentissage.

Le KNN ne fait aucun calcul ici, il se contente de stocker les exemples.

- ✓ Predire()

public string Predire(double[] caracteristiques)

Étapes :

- Calculer la distance entre le point et tous les échantillons
- Trier les distances
- Prendre les k plus proches voisins
- Faire un vote majoritaire
- ✓ CalculerToutesDistances()

Compare le point à classer avec chaque échantillon du dataset.

- ✓ TrierVoisinsParDistance()

Implémente un tri rapide (QuickSort) récursif.

- ✓ VoteMajoritaire()

Identifie les classes parmi les k voisins et renvoie celle qui est la plus fréquente.

2. Data.cs — Chargement des fichiers CSV

Cette classe lit les fichiers CSV contenant les grains.

- ✓ conversion_liste()
- Lit toutes les lignes du fichier
- Identifie les colonnes grâce à l'en-tête
- Convertit chaque ligne en objet
- Retourne une liste de grains

- ✓ Afficher()

Affiche les grains dans la console (utile pour debug).

3. DistanceEuclidienne.cs

Implémente la distance euclidienne :

$$d = \sqrt{\sum (a_i - b_i)^2}$$

Méthode :

public double Calculer(double[] a, double[] b)

4. DistanceManhattan.cs

Implémente la distance Manhattan :

$$d = \sum |a_i - b_i|$$

5. Echantillon.cs

Représente un exemple d'apprentissage.

- double[] Caracteristique→ les 7 valeurs numériques
- string Etiquette→ la classe (Kama, Rosa, Canadian)

6. EnsembleDonnees.cs

Contient une liste d'échantillons.

Méthodes :

- Ajouter()→ ajoute un échantillon
- ObtenirEchantillon()→ retourne la liste
- Taille()→ nombre d'échantillons

7. Grain.cs

Représente un grain brut venant du CSV.

Attributs :

- TypeDeGrain
- Area
- Perimeter
- Compactness
- LongueurNoyau
- LargeurNoyau
- AsymetryCoefficient
- GrooveLength

Il s'agit de la structure brute avant sa conversion en échantillon.

8. TypeDeGrain.cs

Enumération des classes :

Kama, Rosa, Canadian

9. Voisin.cs

Représente un voisin dans le KNN :

Echantillon pour l'exemple

Distance pour distance calculée

10. IClassifier.cs

Interface du classifieur :

- Entrainer()
- Predire()

11. IDistance.cs

Interface pour les distances :

- double Calculer(double[] a, double[] b)

Permet d'utiliser plusieurs distances interchangeables.

12. Program.cs — Le menu principal

C’est le menu de notre application.

Le menu propose :

- Importer données
- Choisir distance
- Entraîner modèle
- Tester modèle
- Quitter
- ✓ Importer données

Importe le fichier CSV d'entraînement et génère les échantillons.

- ✓ Choisir distance

Permet de sélectionner Euclidienne ou Manhattan.

- ✓ Entraîner modèle

Appelle knn.Entrainer().

- ✓ Tester modèle

- Charge test.csv
- Prédit chaque grain
- Remplit la matrice de confusion
- Calcule l’exactitude
- Affiche un tableau Spectre.Console
- Sauvegarde un fichier JSON

- ✓ ClasseToIndex()

Convertit les classes en indices 0,1,2 pour la matrice.

13. ResultatsEvaluationPerformances.cs

Structure de données utilisées pour la sérialisation des résultats d'évaluation au format JSON :

Exactitude

- MatriceConfusion (int[][])
- Distance
- K
- DateEvaluation

- ✓ 14. Ferme, FermeGrain, LotDeGrain, Personne, IClient, IFermier

Dans ton KNN, ces classes ne sont pas mises en œuvre, mais elles figurent dans le schéma métier (gestio n de ferme).

Elles représentent :

- une ferme
- des lots de grains
- des clients
- des fermiers

NB : J'ai retiré ces catégories car certaines données étaient déjà implémentées.

Sources :

Documentation knn

https://visualstudiomagazine.com/articles/2024/10/01/implementing-k-nn-classification-using-c.aspx?Page=1

https://visualstudiomagazine.com/articles/2024/10/01/implementing-k-nn-classification-using-c.aspx?Page=2

Présente, algorithme et vote knn

https://learn.microsoft.com/fr-fr/archive/msdn-magazine/2017/december/test-run-understanding-k-nn-classification-using-csharp

https://medium.com/@kdcodechronicles/understanding-k-nearest-neighbors-algorithm-knn-c-example-d4fce614ea46

Matrice de confusion

https://learn.microsoft.com/en-us/dotnet/api/microsoft.ml.data.confusionmatrix?view=ml-dotnet-preview

Json : serialise, ecrire

https://learn.microsoft.com/en-us/dotnet/standard/serialization/system-text-json/how-to

https://rupen-anjaria.medium.com/working-with-json-in-c-using-system-text-json-9b61f95b551e

https://www.newtonsoft.com/json

https://www.newtonsoft.com/json/help/html/SerializingJSON.htm

spectre.console

https://spectreconsole.net/console

tableau

https://www.luisllamas.es/en/csharp-spectre-console/