

Laboratoire 1 | Algorithme KNN

420-C52 | Données, mégadonnées et intelligence artificielle I

Table des matières

Introduction	3
Objectifs généraux	3
Objectifs spécifiques	3
Considérations algorithmiques	3
Algorithme KNN	3
Réduction de complexité	4
Filtrage	5
Conditionnement	5
Manipulation d'image	5
Calcul de distance 3D	5
Considérations opérationnelles	5
Interface utilisateur graphique	6
Les données	7
Classification à réaliser	7
Ensembles de données	7
Caractéristiques techniques importantes des images	8
Accès aux données	9
Widget de consultation des données	10
Rapport et bouton À propos... ..	12
Quelques outils informatiques.....	14
Contraintes techniques	14
Évaluation	14
Annexe : Description des ensembles de données	15
Annexe : Description des classes d'images.....	17
Annexe : Captures d'écran d'une proposition de l'interface usager	19

Introduction

Ce laboratoire consiste à réaliser une application apte à classifier automatiquement des images.

Outre les considérations techniques et algorithmiques, on désire que l'application soit à forte teneur éducative au sens où la visualisation du problème est au centre des opérations.

À cette étape de vos études collégiales, ce projet se veut une intégration de plusieurs notions et une excellente opportunité de les mettre en pratique. Notamment, les aspects suivants sont abordés :

- travail d'équipe
- conception, planification et réalisation
- langage **Python**
- bibliothèques **Qt**, **NumPy**, **psycopg2** et **Matplotlib**
- accès à une base de données relationnelle (**PostgreSQL** dans ce cas-ci)

Une attention particulière doit être apportée aux notions d'abstraction, de modularité et de réutilisabilité de votre projet. Dans cet esprit, vous devez pouvoir faire la distinction entre les éléments génériques et spécifiques de cette application pour cibler un développement adapté et pertinent.

Objectifs généraux

Les objectifs principaux de ce laboratoire sont :

- implémenter l'algorithme KNN de façon générique
- utiliser l'algorithme KNN pour résoudre un problème de classification d'images
- réaliser une application complète supportant ces fonctionnalités
- accéder aux données provenant d'une base de données.

Objectifs spécifiques

Plus spécifiquement, ce laboratoire vise à :

- déterminer les métriques qui permettront de réduire la complexité des données à classifier (les images) – vous devez utiliser exactement un espace de solution à 3 dimensions;
- représenter, par une visualisation 3D efficace, l'espace de solution
- implémentation efficace et générique de l'algorithme KNN,
- implémentation efficace et générique des algorithmes d'analyse d'image,
- connexion et accès aux données d'une base de données relationnelle,
- réalisation d'une interface utilisateur assemblant les diverses parties de votre application.

Considérations algorithmiques

Algorithme KNN

L'algorithme **KNN** consiste à classifier un objet en utilisant d'autres objets comparables dont la classe a été préalablement établie (étiquetage).

On peut réduire la présentation de cet algorithme en 5 étapes logiques distinctes :

1. on dispose dans un espace à n dimensions les données d'apprentissage
2. on dispose dans le même espace l'objet à classer
3. on calcul la distance entre l'objet à classer et chacune des données d'apprentissage
4. on retient les k voisins les plus près
5. on retient la classe la plus fréquente parmi ces k voisins

On comprend que le fait de faire plusieurs classifications demande à chaque fois la réalisation des étapes 2 à 5. En contrepartie, l'étape 1 n'est réalisée qu'une seule fois pour un ensemble de données d'apprentissage spécifique.

Pour les étudiants plus chevronnés, il sera possible de pousser un peu en déterminant une solution viable pour les cas limites qui arrivent inévitablement :

- à l'étape 4 :
 - problème : il est possible que la distance la plus près soit trop grande pour correspondre à une classification viable
 - solution : on ajoute ainsi un critère de distance maximum pour accepter la classification
- à l'étape 5 :
 - problème : il existe plusieurs classes qui partagent la même fréquence
 - solution : déterminer la classe qui présente la plus petite distance ou la distance moyenne la plus petite

Réduction de complexité

Vous avez à votre disposition plusieurs images possédant les mêmes caractéristiques techniques. Entre autres, elles sont toutes de dimension identique : 150 x 150 pixels, totalisant 22 500 pixels par image. Utiliser tous les pixels de chaque image amène un problème de haute dimension et l'algorithme **KNN** ne pourrait bien performer (peut être dans une certaine mesure avec des considérations qui dépassent le cadre du cours).

La réduction de complexité devient alors une tâche essentielle permettant une simplification du problème et, par le fait même, elle rend possible une classification efficace par le **KNN**.

Sachant qu'on désire mettre l'emphasis sur la visualisation du processus, on vous demande de déterminer en équipe trois métriques qui permettront une description efficace et pertinente des formes présentes sur les images. Ainsi, votre espace de solution sera 3D et facile à comprendre lors de sa représentation.

La détermination de ces métriques est la partie la plus sensible de ce laboratoire. Pour chaque métrique, vous devez garder en tête ces caractéristiques :

- l'utilisation d'un nombre est plus simple
- autant que possible, la métrique doit favoriser une dispersion des données d'apprentissage pour chaque classe existantes
- chaque métrique doit être indépendante les unes des autres
- idéalement, la métrique est simple à calculer
- idéalement, la métrique est invariante aux transformations géométriques appliquées aux images (translation, rotation, homothétie)
- idéalement, les métriques sont normalisées dans des plages connues.

Par exemple, l'aire et le périmètre sont des nombres indépendants qui permettent généralement une bonne dispersion des classes et qui sont relativement simples à calculer. D'un autre côté, ils ne sont pas invariants à l'homothétie.

Filtrage

Les données d'entrées ont été générées artificiellement sans bruit. Par le fait même, aucun filtrage n'est nécessaire.

Conditionnement

Le conditionnement consiste à préparer les métriques pour qu'elles soient utilisées par l'algorithme KNN. Le but étant de vous assurer que chaque dimension soit comparable les unes avec les autres par leur domaine de valeurs.

Par exemple, supposons ces deux métriques :

- la première métrique produit des valeurs dans l'intervalle [1000, 5000]
- la deuxième métrique génère des valeurs dans l'intervalle [0, 1]

Dans ce cas, il est certain qu'un calcul de distance donnera un très fort biais sur la première métrique rendant la deuxième absolument insignifiante simplement à cause d'un facteur d'échelle.

C'est à vous de mettre de l'avant une stratégie qui garantit que le calcul de distance soit pertinent.

Manipulation d'image

Vous devez utiliser la librairie **NumPy** pour faire les calculs sur les images. Vous trouverez en annexe quelques éléments de code utilitaire à cet effet.

Calcul de distance 3D

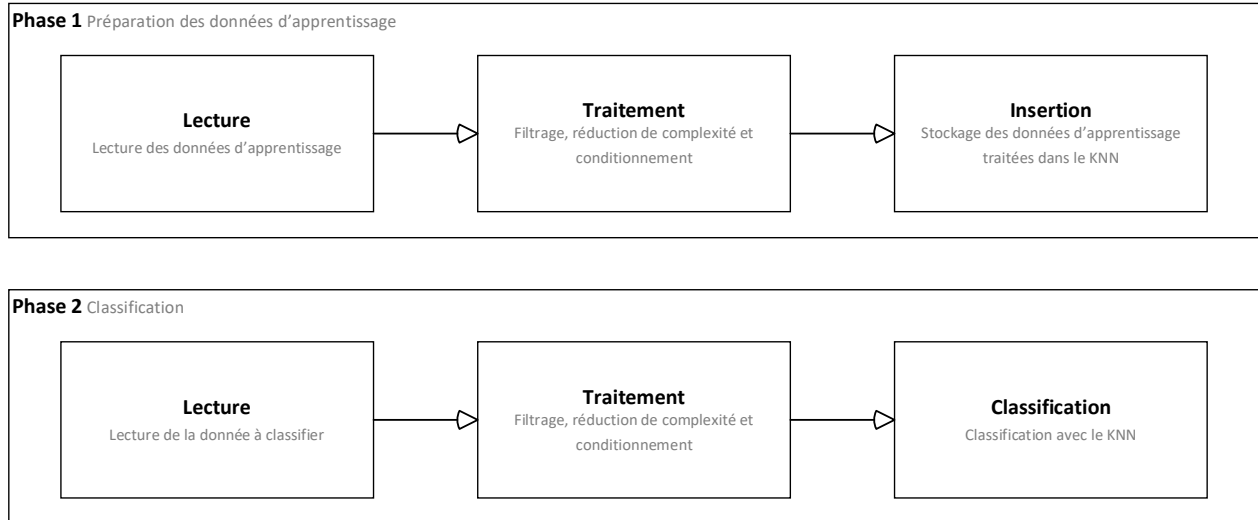
Pour ce projet, on utilisera la formule de Pythagore pour calculer la distance dans un espace à trois dimensions. Considérant que vous avez les deux points p et q , le calcul de distance est alors :

$$d = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2 + (p_z - q_z)^2}$$

Considérations opérationnelles

Même si l'algorithme **KNN** est simple, son utilisation en production nécessite la mise en place d'outils permettant un usage utile et adéquat. On propose ici une approche en deux phases :

1. Préparation des données d'apprentissage
2. Classification



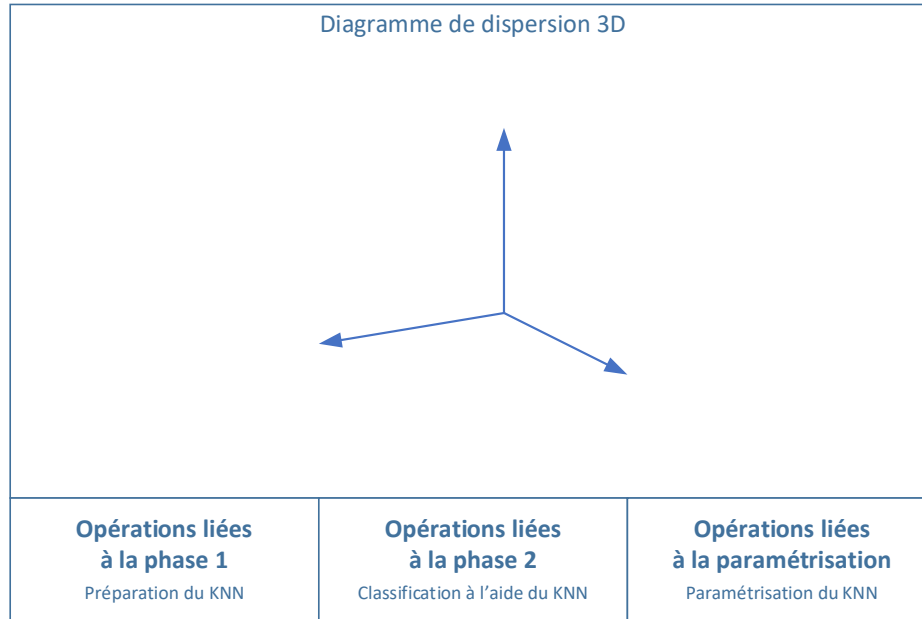
À cet égard, l'interface graphique doit être orientée de façon à simplifier ces deux phases opérationnelles.

Interface utilisateur graphique

Vous devez produire une interface graphique divisée en quatre sections :

- Visualisation de l'espace de solution :
 - Un diagramme de dispersion 3D
- Opérations liées à la phase 1 :
 - minimalement,
 - un *widget* permettant de sélectionner un ensemble de données
 - quelques informations sur l'ensemble de données choisi
 - à la sélection de l'ensemble de données, les données sont traitées et insérées dans le KNN.
- Opérations liées à la phase 2 :
 - minimalement,
 - un *widget* permettant de sélectionner une image à classifier
 - un bouton permettant de lancer le processus de classification
 - un court texte donnant les informations sur les résultats de l'opération de classification
 - Cette section ne devrait pas être disponible si la première phase n'est pas réalisée.
- Opérations liées à la paramétrisation :
 - minimalement :
 - un *widget* permettant de déterminer le paramètre k (de 1 à n)
 n étant le quart du nombre d'images dans l'ensemble d'entraînement
 - optionnellement, un *widget* permettant de déterminer la distance maximum acceptable (c'est à vous de déterminer cette plage)

Lorsque des opérations liées aux phases 1 et 2 sont réalisées, il est nécessaire que le diagramme de dispersion se mette à jour afin de représenter les données.



Présentation suggérée

Vous n'avez aucune contrainte sur la présentation et la disposition de votre interface utilisateur. L'important est de respecter les contraintes données.

Vous trouverez en annexe quelques captures d'écran représentant un exemple de ce qui est à faire.

Les données

Vous avez à votre disposition un large éventail d'images et d'ensembles de données vous permettant de tester votre logiciel pour différents scénarii. Plus précisément vous avez :

- 45 ensembles de données
- 23 834 images
- 34 classes
- 8 combinaisons de transformations géométriques

Classification à réaliser

Vous n'avez pas à réussir les classifications sur chacun des ensembles de données. On vous propose plutôt d'envisager ces ensembles de données comme des défis progressifs de difficultés.

C'est la qualité de vos descripteurs de forme qui permettront de produire de meilleurs résultats pour les différents ensembles de test.

Ensembles de données

Chaque ensemble de données représente un ensemble destiné à tester votre projet. Ils possèdent une série d'images séparées en deux sous-ensembles :

- l'ensemble d'apprentissages,
- l'ensemble test.

De plus, chaque ensemble de données possède un nombre variable de :

- classes d'images
- transformations géométriques appliquées sur les formes.

Ainsi, plus il y a de variations, plus grand est le défi de classification.

Il existe 13 structures d'ensembles de données pour lesquels 12 ont 4 tailles formant ainsi 45 ensembles de données différents.

Vous trouverez en annexe, une description détaillée des ensembles de données et des images.

Caractéristiques techniques importantes des images

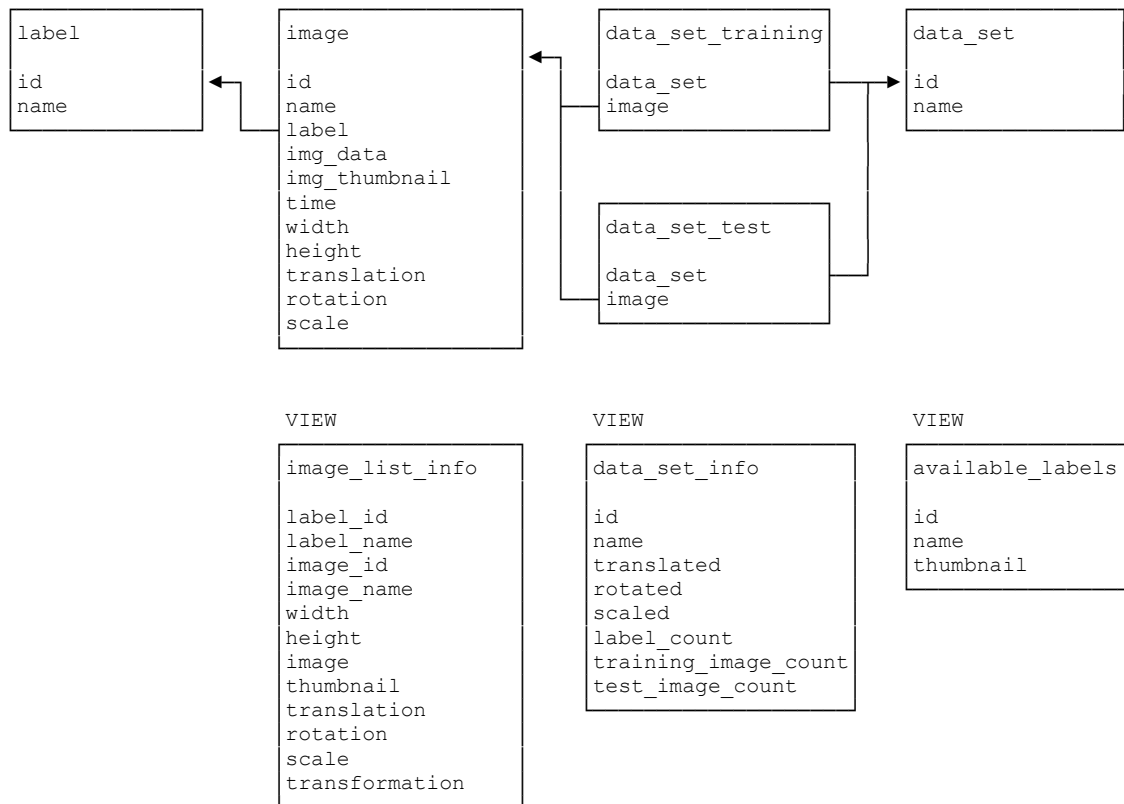
Afin de simplifier la nature de ce projet, les images ont été générées artificiellement et présentent toutes les mêmes caractéristiques importantes :

- Les images sont binaires :
 - le fond de l'image est blanc, on considère que l'arrière-plan (le *non-signal*) est 255
 - le signal (les formes qui nous intéressent) est noir (typiquement 0)
- Les images sont opaques (aucun pixel n'est transparent).
- Il n'y a qu'une seule forme dans chaque image.
- La forme ne présente aucun trou.
- la forme est définie par un voisinage à 8 pixels
- La forme ne touche pas le contour de l'image. Ceci implique qu'il existe au moins 1 pixel blanc tout autour de l'image.
- Plusieurs métadonnées sont incluses à même les images :
 - **Tag** = la classe sous forme de chaîne de caractères (c'est-à-dire l'étiquette donnée)
 - **Author** = l'auteur de l'image (une chaîne de caractère correspondant exactement à « KlustR »)
 - vous pouvez vous servir de ces informations pour déterminer automatique la classe de l'image, mais aussi si l'image est conforme à ce projet.
- Le format de sauvegarde est PNG.
- Le format d'encodage des images est de 32 bits organisés ARGB (correspondant au format de Qt `QImage.Format_ARGB32`).
- Pour chaque image, la forme peut avoir subi aucune ou plusieurs transformations affines: translation, rotation, homothétie (*scaling*). Il existe 8 configurations de transformation possibles :

Identifiant	Translation	Rotation	Homothétie
000	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
100	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
010	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
001	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
110	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
101	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
011	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
111	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Accès aux données

Les données sont disponibles sur une base de données relationnelle PostgreSQL. La base de données se trouve dans le schéma est **klustr**. L'architecture des tables est la suivante :



Le fichier **klustr.dump** correspond à la base de données que vous devez installer sur votre poste. Vous devez exécuter la commande suivante pour restaurer la base de données :

```
pg_restore --host=localhost --port=5432 --username=postgres --dbname=postgres klustr.dump
```

L'application **pg_restore** permet l'installation d'un « *dump file* » dans **PostgreSQL**. Les arguments préfixés par un double tirets correspondent aux options habituelles liées aux paramètres de connexion.

Pour que la ligne de commande précédente fonctionne, il faut que les accès aux fichiers **pg_restore** et **klustr.dump** soient possibles. Par exemple, au cégep vous devez utiliser cette approche si vous avez déposé le fichier **klustr.dump** dans le dossier **c:/travail/** :

```
c:/travail/PostgreSQL/14/bin/pg_restore
--host=localhost --port=5432 --username=postgres --dbname=postgres C:/travail/klustr.dump
```

Après restauration, les informations de connexion sont (que vous pouvez utiliser dans votre application ou avec **pgAdmin** par exemple) :

host	localhost	user	postgres
port	5432	password	AAAAaa123
database	postgres		

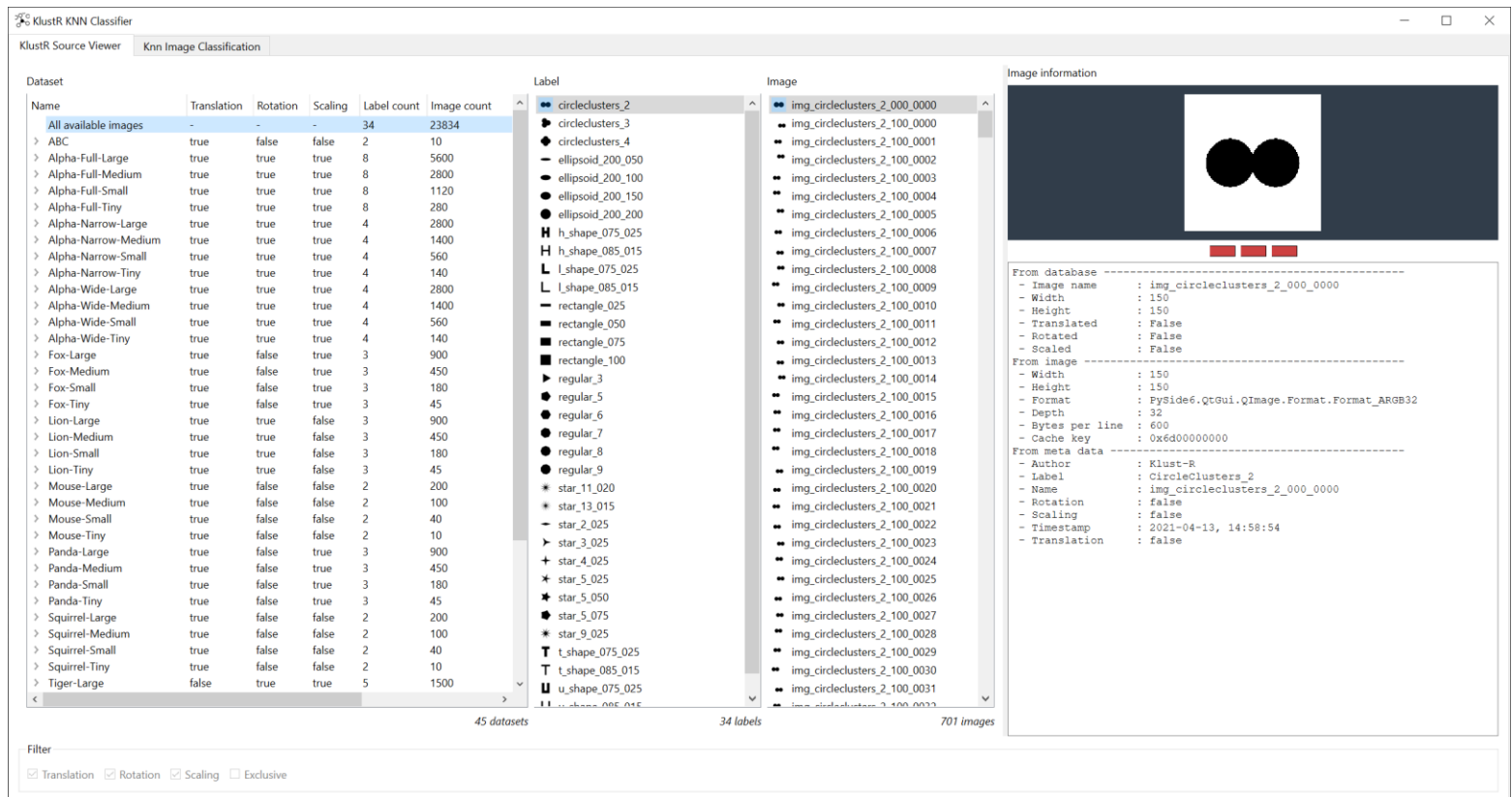
À même la base de données, vous avez à votre disposition les fonctions suivantes qui facilitent l'accès aux données :

- `klustr.available_datasets()`
in : rien
out : retourne les informations sur les ensembles de données
- `klustr.select_label_from_data_set(dataset_name)`
in : le **nom** de l'ensemble de données
out : retourne les informations sur les classes d'images incluses dans l'ensemble de données
- `klustr.select_image_from_data_set(dataset_name, training_image)`
in : le **nom** de l'ensemble de données
si les données sont celles d'apprentissage (TRUE) ou de test (FALSE)
out : retourne les informations sur les images d'un ensemble de données

Widget de consultation des données

Vous avez à votre disposition un *widget* permettant de consulter les données facilement. Cet outil est à la fois un exemple de code pour plusieurs aspects du projet, mais aussi un outil indispensable pour mieux comprendre la structure des données mise à votre disposition.

La classe `KlustrDataSourceViewWidget` disponible dans le fichier `klustr_widget.py` est un *widget* dans lequel se trouvent, par assemblage, plusieurs constituants formant une vue facile à utiliser. La capture d'écran suivante représente ce *widget*.



Cette classe possède toutefois plusieurs dépendances :

- `sys`
- `pyside6`
- donné avec les fichiers du projet :
 - `db_credential`
 - `klustr_dao`
 - `klustr_utils`

L'exemple de code suivant présente un cas minimum de son usage. Dans cet exemple, le *widget* représente l'application principale. Gardez en tête que cette classe hérite directement de `QWidget` et que, dès lors, il peut être utilisé en tant que tel.

```
# Importation des modules
import sys

from db_credential import PostgreSQLCredential
from klustr_dao import PostgreSQLKlustrDAO
from klustr_widget import KlustrDataSourceViewWidget

from PySide6 import QtWidgets
from __feature__ import snake_case, true_property

# Application principale de Qt
app = QtWidgets.QApplication(sys.argv)

# Information de connexion à la base de données
credential = PostgreSQLCredential(
    host='localhost',
    port=5432,
    database='postgres',
    user='postgres',
    password='ASDasd123')

# DAO utilisé
klustr_dao = PostgreSQLKlustrDAO(credential)

# Instanciation et affichage du widget de visualisation des données du projet KlustR
source_data_widget = KlustrDataSourceViewWidget(klustr_dao)
source_data_widget.show()

# Démarrage de l'engin Qt
sys.exit(app.exec_())
```

Rapport et bouton À propos...

Vous avez un mini rapport à produire pour ce projet. Ce dernier consiste à donner les informations principales du projet et une description concise, mais précise de ce que vous avez réalisé.

Ce rapport doit aussi prendre la forme du texte afficher par une boîte de dialogue de type **À propos...** (ou *About*). Cet élément sous-entend que vous devez disposer d'un bouton qui affiche cette boîte de dialogue.

Le texte à produire est strict et correspond à ce modèle où tous les mots balisés par un double soulignement sont à remplacer.

Ce logiciel est le projet no 1 du cours C52.

Il a été réalisé par :

- __étudiant_1__
- __étudiant_n__

Il consiste à faire __quelque_chose__ avec les concepts suivants :

- __concept_1__
- __concept_n__

Nos 3 descripteurs de forme sont :

- __descripteur_1__
 - en __unité__ pour le domaine __domaine__
 - correspondant à __courte_description__
- __descripteur_2__
 - en __unité__ pour le domaine __domaine__
 - correspondant à __courte_description__
- __descripteur_3__
 - en __unité__ pour le domaine __domaine__
 - correspondant à __courte_description__

Plus précisément, ce laboratoire permet de mettre en pratique les notions de :

- __notion_1__
- __notion_n__

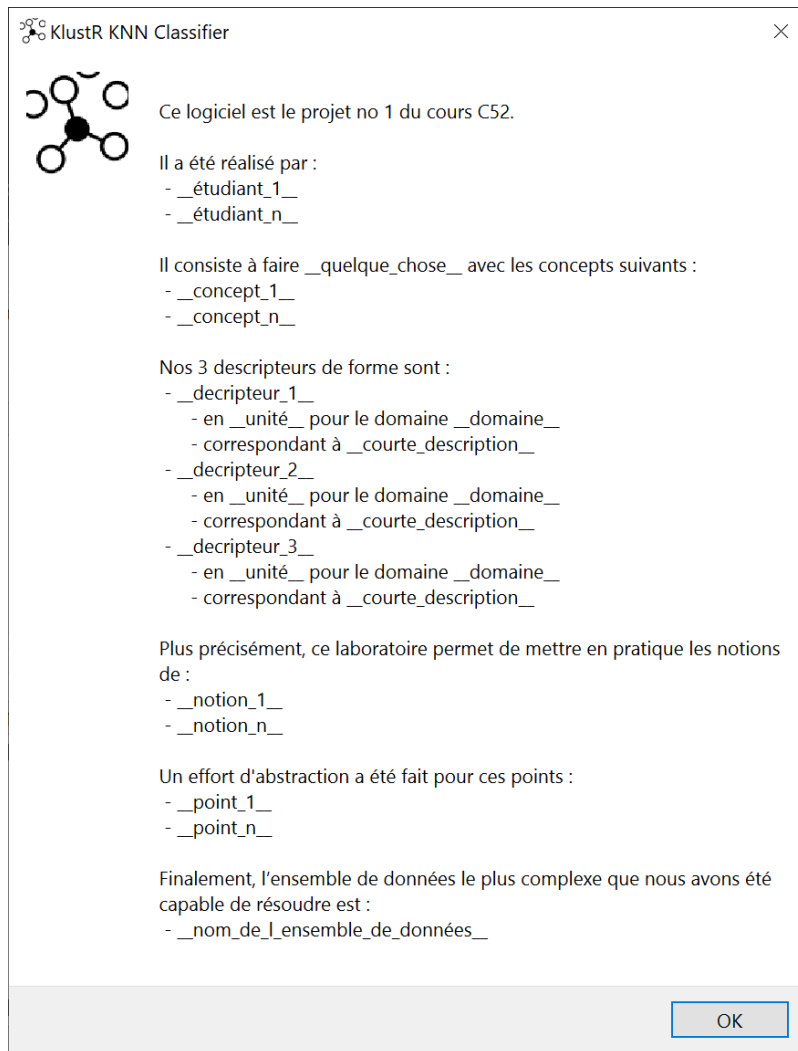
Un effort d'abstraction a été fait pour ces points :

- __point_1__
- __point_n__

Finalement, l'ensemble de données le plus complexe que nous avons été capable de résoudre est :

- __nom_de_l_ensemble_de_données__

La capture d'écran suivante présente un exemple de cette boîte de dialogue.



Pour votre information, sachez que la méthode statique `QMessageBox.about` facilite ce genre de présentation.

Quelques outils informatiques

Vous avez à votre disposition quelques fonctions essentielles :

- `klustr_utils.qimage_argb32_from_png_decoding`
Effectue le décodage d'un 'buffer' de données correspondant à une image PNG. Essentiel pour décoder les images provenant de la base de données.
- `klustr_utils.ndarray_from_qimage_argb32`
Effectue la conversion d'une image de format QImage.Format_ARGB32 vers une matrice NumPy. Essentiel pour convertir les images d'entrées en matrice NumPy.

Contraintes techniques

Pour la réalisation de ce projet, vous devez respecter ces contraintes :

- Ce projet doit être réalisé **en équipe** de 4 étudiants. Il est essentiel que les membres de l'équipe travaillent équitablement.
- Ce projet doit être réalisé en **Python** avec les bibliothèques **Qt**, **NumPy**, **psycopg2** et **Matplotlib**. Vous ne devez pas utiliser d'autres librairies externes telles que **OpenCV**.
- Incluant la période de présentation du projet, vous avez 4 semaines pour réaliser ce projet.

Évaluation

Ce travail est évalué par ces critères (en ordre d'importance) :

1. Qualité des 3 métriques utilisées – le concept.
2. Implémentation de l'algorithme **KNN**.
3. Modularité et réutilisabilité du code là où ça compte.
4. Implémentation des algorithmes d'extraction des métriques à partir des images.
5. Implémentation de l'interface usager.
6. Niveau le plus élevé des ensembles de données que vous pouvez résoudre.
7. Qualité du code **Python** et respect de **PEP8**.

À la fin du projet, chaque équipe doit remplir une grille Excel dans laquelle se trouvent compilées des mesures comparatives sur la participation de chacun des membres.

Voir le fichier : **Fiche autoévaluation comparative.xlsx**.

La note finale est établie avec ces paramètres (voir le fichier Excel donné) :

- Proportion de la note d'équipe : 50%
- Redistribution des points excédentaires : 0%

Annexe : Description des ensembles de données

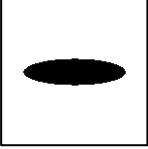
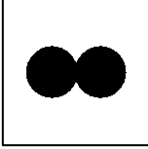
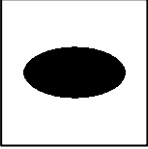
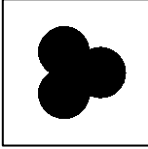
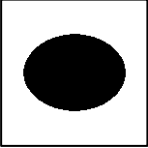
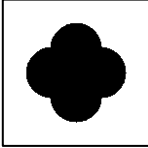
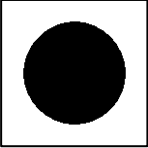
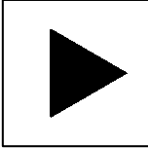
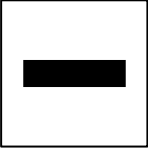
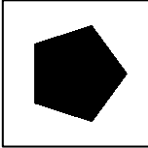
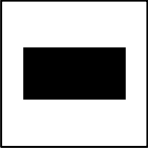
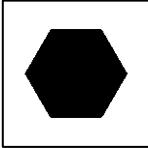
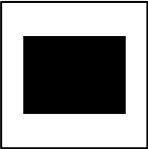
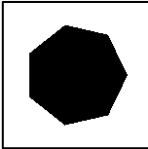
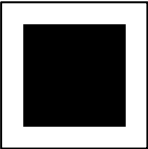
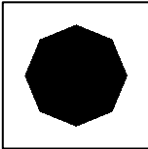
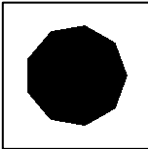
Voici la liste des ensembles de données triées en ordre de difficulté croissante :


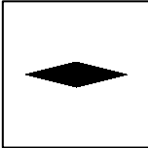

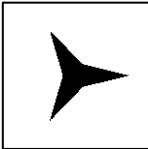
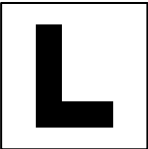
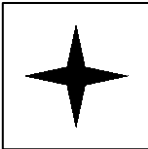
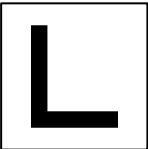


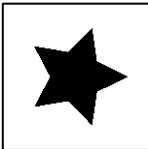
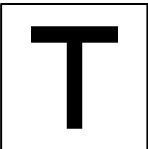
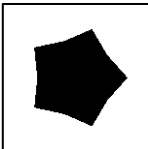
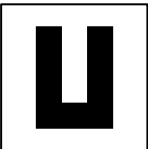
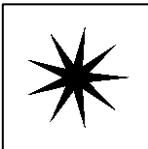
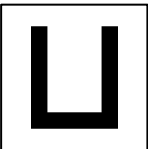
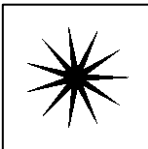
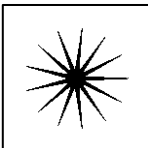
Nom	Information
ABC	Nombre de classes d'image : 2 Translation, rotation, homothétie : 100 Tailles de l'ensemble de données en nombre d'images : 10 Commentaires : présente l'ensemble de départ (très simple) – Cercle et triangle
Mouse	Nombre de classes d'image : 2 Translation, rotation, homothétie : 100 Tailles de l'ensemble de données en nombre d'images : 10, 40, 100, 200 Commentaires : Cercle et Étoile à 2 pointes
Squirrel	Nombre de classes d'image : 2 Translation, rotation, homothétie : 100 Tailles de l'ensemble de données en nombre d'images : 10, 40, 100, 200 Commentaires : Cercle et Étoile à 3 pointes
Panda	Nombre de classes d'image : 3 Translation, rotation, homothétie : 101 Tailles de l'ensemble de données en nombre d'images : 45, 180, 450, 900 Commentaires : Carré, Étoile à 2 pointes et Étoile à 5 pointes
Fox	Nombre de classes d'image : 3 Translation, rotation, homothétie : 101 Tailles de l'ensemble de données en nombre d'images : 45, 180, 450, 900 Commentaires : Noyau de 3 cercles, Triangle et Étoile à 3 pointes
Lion	Nombre de classes d'image : 3 Translation, rotation, homothétie : 110 Tailles de l'ensemble de données en nombre d'images : 45, 180, 450, 900 Commentaires : Polygones à 3, 5 et 7 côtés
Tigre	Nombre de classes d'image : 5 Translation, rotation, homothétie : 011 Tailles de l'ensemble de données en nombre d'images : 75, 300, 750, 1500 Commentaires : Étoile à 2, 3, 4, 5 et 9 pointes
Trex	Nombre de classes d'image : 9 Translation, rotation, homothétie : 111 Tailles de l'ensemble de données en nombre d'images : 315, 1260, 3150, 6300 Commentaires : Noyau de 2-3-4 cercles, Polygones à 3, 5 et 7 côtés, Étoile à 5 pointes étroite, moyenne et épaisse
Alpha-Narrow	Nombre de classes d'image : 4 Translation, rotation, homothétie : 111 Tailles de l'ensemble de données en nombre d'images : 140, 560, 1400, 2800 Commentaires : Lettres étroites H-L-T-U
Alpha-Wide	Nombre de classes d'image : 4 Translation, rotation, homothétie : 111 Tailles de l'ensemble de données en nombre d'images : 140, 560, 1400, 2800 Commentaires : Lettres épaisses H-L-T-U

Alpha-Full	<p>Nombre de classes d'image : 8</p> <p>Translation, rotation, homothétie : 111</p> <p>Tailles de l'ensemble de données en nombre d'images : 280, 1120, 2800, 2600</p> <p>Commentaires : Lettres étroites et épaisses H-L-T-U</p>
Zoo	<p>Nombre de classes d'image : 34</p> <p>Translation, rotation, homothétie : 111</p> <p>Tailles de l'ensemble de données en nombre d'images : 1190, 4760, 11900, 23800</p> <p>Commentaires : toutes les classes d'image disponibles</p>

Annexe : Description des classes d'images

Il existe 34 classes d'image disponibles.

Classe (étiquette) en respectant la casse	Exemple	Classe (étiquette) en respectant la casse	Exemple
Ellipsoid_200_050		CircleClusters_2	
Ellipsoid_200_100		CircleClusters_3	
Ellipsoid_200_150		CircleClusters_4	
Ellipsoid_200_200		Regular_3	
Rectangle_025		Regular_5	
Rectangle_050		Regular_6	
Rectangle_075		Regular_7	
Rectangle_100		Regular_8	
		Regular_9	

H_Shape_075_025		Star_2_025	
H_Shape_085_015		Star_3_025	
L_Shape_075_025		Star_4_025	
L_Shape_085_015		Star_5_025	
T_Shape_075_025		Star_5_050	
T_Shape_085_015		Star_5_075	
U_Shape_075_025		Star_9_025	
U_Shape_085_015		Star_11_020	
		Star_13_015	

Annexe : Captures d'écran d'une proposition de l'interface usager

La capture d'écran suivante est un exemple de ce que vous pouvez faire. Vous n'avez aucune contrainte sur la présentation en autant que tous les éléments demandés soient présents.

