Titre : *Développement d'une application console en C# pour la classification automatique des grains de blé.*

Cours	Programmation orientée objet II (INF11207)
Session	Hiver 2026
Contact	yacine_yaddaden@uqar.ca

Table des matières

1. Objectif du travail

L'objectif de ce travail pratique est de concevoir et de développer une **application console en C#**, orientée objet, permettant de **classifier automatiquement des grains de blé** à partir du jeu de données *Seeds*. Ce projet vise à mettre en pratique les concepts avancés de la programmation orientée objet, tels que les **interfaces**, les **classes abstraites**, l'**héritage** et la **modélisation UML**.

Les étudiant(e)s devront implémenter un **classifieur *k*-NN**, gérer des **fichiers CSV et JSON**, intégrer des **algorithmes de recherche et de tri**, et développer une **interface console interactive** avec *Spectre.Console*. Ce travail permet également d'évaluer les performances du système à l'aide de métriques standards.

2. Technologies à utiliser

Pour mener à bien ce projet, les technologies suivantes seront utilisées :

☑ **Environnement de développement et outils :**
- → Microsoft Visual Studio 2026 Community (programmation).
- → Git pour la gestion de version.
- → GitHub pour héberger le dépôt du projet.

☑ **Langage de programmation :**
- → C# avec des notions avancées de programmation orientée objet.

☑ **Bibliothèques :**
- → **CsvHelper** pour la gestion des fichiers CSV.
- → **Json.NET** pour la gestion des fichiers JSON.
- → **Spectre.Console** pour de belle interface console en C#.

☑ **Logiciel :**
- → **NClass** pour la création du diagramme de classes UML.

Il est recommandé de faire de la *gestion de version* avec l'outil **Git**. Il est recommandé également de créer un dépôt sur GitHub.

3. Description détaillée

Afin de réaliser le projet, il est nécessaire de tenir compte des éléments suivants :

3.1. Analyse du problème et cahier des charges

Dans cette première étape, les étudiant(e)s doivent analyser la problématique générale et comprendre le fonctionnement attendu de l'application. Le but est de classifier automatiquement des grains de blé à partir du jeu de données *Seeds*, qui comporte trois classes (**Kama**, **Rosa** et **Canadian**) et sept caractéristiques numériques. Les données sont fournies sous forme de deux fichiers CSV : un fichier d'apprentissage (*train.csv*) et un fichier de test (*test.csv*). L'application devra produire les prédictions sur les données de test, calculer l'exactitude (taux de reconnaissance) et la matrice de confusion, et sauvegarder l'ensemble des paramètres et résultats dans un fichier JSON global.

3.2. Modélisation UML du domaine métier

Cette étape consiste à modéliser le domaine applicatif agricole indépendamment de l'algorithme de classification. Les étudiant(e)s doivent concevoir un diagramme de classes UML représentant la gestion d'une ferme et de ses acteurs, tels que la ferme, le fermier, les clients ou les lots de grains. Le diagramme doit inclure de l'héritage, au moins une interface et au moins une classe abstraite, ainsi que des relations UML clairement identifiées (associations, compositions et multiplicités). L'objectif est de bien distinguer la logique métier de la logique algorithmique.

3.3. Modélisation UML pour la classification *k*-NN

Dans cette étape, les étudiant(e)s doivent concevoir le diagramme UML du module de classification *k*-NN. L'architecture proposée doit être orientée objet, extensible et basée sur des interfaces et le polymorphisme. Elle doit permettre de configurer dynamiquement le nombre de voisins *k* et d'utiliser plusieurs distances (au minimum Euclidienne et Manhattan). La recherche des plus proches voisins devra être réalisée à l'aide d'une **approche naïve (brute force)**, consistant à calculer la distance entre l'échantillon à classifier et l'ensemble des données d'apprentissage, puis à trier ces distances afin de sélectionner les *k* plus proches voisins. Les responsabilités entre la classification, le calcul des distances et les structures de données doivent être clairement séparées et justifiées.

3.4. Définition de la structure du fichier JSON global

Cette étape vise à définir la structure d'un fichier JSON unique permettant de stocker l'état global de l'application. Ce fichier doit contenir les paramètres d'exécution (valeur de *k*, distance utilisée, date), les informations sur les jeux de données (taille des ensembles d'apprentissage et de test) ainsi que les résultats de l'évaluation (exactitude et matrice de confusion). La structure proposée doit être claire, cohérente et extensible afin de permettre des évolutions futures.

3.5. Chargement et validation des données CSV

Dans cette étape, les étudiant(e)s doivent implémenter la lecture et la validation des fichiers *train.csv* et *test.csv*. Chaque ligne du fichier doit être vérifiée, convertie en données numériques valides et transformée en un objet métier représentant un grain de blé. Les erreurs de lecture ou de format doivent être correctement gérées et aucune donnée ne doit être codée en dur dans l'application.

3.6. Implémentation du classifieur *k*-NN

Les étudiant(e)s doivent implémenter l'algorithme de classification *k*-NN. Le classifieur doit permettre de configurer dynamiquement le nombre de voisins *k* ainsi que la distance utilisée. La décision finale doit être basée sur un vote majoritaire et les cas particuliers, comme les égalités ou le nombre insuffisant de voisins, doivent être correctement gérés. Le classifieur doit être accessible via une interface afin de respecter les principes de la programmation orientée objet.

3.7. Algorithme de tri

Dans le cadre de la recherche des *k* plus proches voisins, les étudiant(e)s doivent implémenter explicitement un algorithme de tri de référence vu au cours. Cet algorithme sera utilisé pour trier les voisins selon leur distance ou pour maintenir la liste des *k* plus proches voisins. L'utilisation des méthodes de tri intégrées du langage n'est pas autorisée et le choix de l'algorithme devra être justifié dans le rapport.

3.8. Évaluation des performances

Cette étape vise à évaluer la qualité du classifieur sur les données de test. Les étudiant(e)s doivent comparer les prédictions aux classes réelles, calculer automatiquement l'exactitude et la matrice de confusion (3×3), afficher clairement ces résultats dans la console et les sauvegarder dans le fichier JSON global.

3.9. Interface console interactive

Les étudiant(e)s doivent développer une interface console interactive et soignée à l'aide de la bibliothèque **Spectre.Console**. L'interface doit permettre de

configurer les paramètres de classification, de lancer le processus de classification et d'afficher les résultats sous une forme claire et lisible. Une attention particulière sera portée à la validation des entrées utilisateur et à la qualité de la présentation.

4. Structure du projet

Lors de la création d'un nouveau projet dans **Microsoft Visual Studio 2026 Community**, une structure de base est automatiquement générée. Il est recommandé de :

☑ Organiser les fichiers en créant un dossier dédié aux **classes** du projet.
☑ Utiliser **Git** pour la gestion des versions et **GitHub** pour le partage.

5. Modalité d'évaluation

Le travail pratique comptera pour **30%** de la note du cours. Ces points sont répartis de la manière suivante :

Partie	Points
Modélisations UML	5,0%
Fonctionnalités à implémenter	**20,0%**
→ Chargement des données (CSV)	2,0%
→ Classifieur *k*-NN fonctionnel	6,0%
→ Distances configurables	2,0%
→ Algorithme de tri implémenté	2,0%
→ Évaluation des performances	2,0%
→ Interface console fonctionnelle	6,0%
Qualité du code source	1,5%
Qualité du rapport final	2,0%
Vidéo de présentation du projet	1,5%
Total	**30%**

6. Date de remise

→ La date limite de remise est le **28 février 2026 avant 23h00**
→ Fichiers à remettre :
  ✓ Un fichier compressé (.zip) contenant le code source,
  ✓ Le rapport (un modèle **LaTex** est fourni),
  ✓ Une courte vidéo de démonstration faite avec **OBS Studio**.

Tous les fichiers doivent être remis sur Moodle avant la date limite.

7. Assistance

Si vous avez besoin d'assistance, vous pouvez contacter l'une de nos deux personnes auxiliaires d'enseignement et de recherche, selon le campus où vous vous trouvez :

☑ Dorra Lamouchi (Lévis) : Dorra_Lamouchi@uqar.ca
☑ François Gosselin (Rimouski) : Francois_Gosselin@uqar.ca

Si vous avez besoin de plus d'informations ou de précisions concernant le travail à réaliser, vous pouvez contacter le professeur par courriel.

8. Points importants

Note I :

→ Le travail est en équipe de deux (voir la liste des équipes),
→ Le professeur peut poser des questions liées au travail pratique,
→ Le non-respect de l'énoncé entraînera une perte de points,
→ Le plagiat sera sanctionné selon la politique de l'université,
→ Le retard dans la remise du projet entraînera des pénalités,
→ Il est strictement interdit d'utiliser des outils de l'IA.

Note II : *Dans le cas où il y a des aspects qui ne sont pas clairs, n'hésitez pas à m'en faire part afin que je puisse apporter des éclaircissements et éventuellement mettre à jour l'énoncé du travail pratique.*

9. Documentation

Il est recommandé d'utiliser la documentation officielle des différentes bibliothèques indiquées plus haut. Il est également possible de consulter d'autres documentations sur internet.