|  |
| --- |
| Université du Québec à Rimouski |
| **Travail Pratique I** |
| 8INF957 - Programmation objet avancée  Professeur – Yacine Yaddaden, Ph. D.  **Application pour Médecin et diagnostique par IA** |

|  |  |
| --- | --- |
| |  | | --- | | **⭢ Mamadou Mouslim Diallo**  **⭢ Benjamin Lapointe-Pinel**  **⭢ FateMeh Bashardoustjoubjarkouli** | |

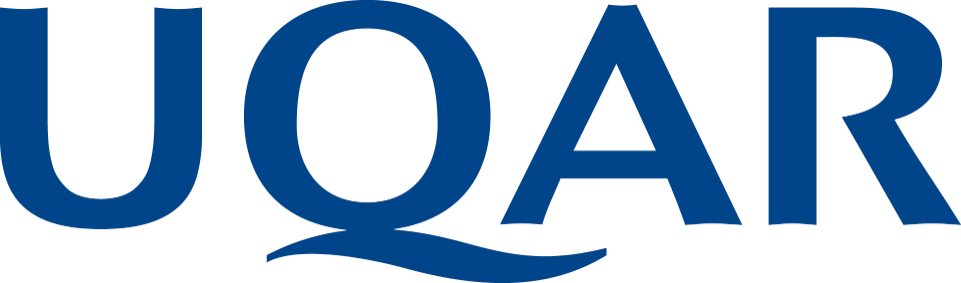


Table des matières

[Introduction 2](#_Toc96175531)

[Modélisation UML 2](#_Toc96175532)

[Aperçu de l’application (GUI) 2](#_Toc96175533)

[Algorithme du -NN et Résultats 2](#_Toc96175534)

[Problèmes et Difficultés rencontrées 2](#_Toc96175535)

[Conclusion 2](#_Toc96175536)

[Références 2](#_Toc96175537)

# Introduction

Ce travail a été réalisé dans le cadre des TP dans le cours de Programmation Orientée Objet Avancée. Le travail consiste à concevoir une application de bureau destinée aux médecins qui capable de faire des diagnostique par l’intelligence artificielle à l’aide de l’algorithme KNN. Pour la réalisation de notre projet conformément à l’énoncé, nous avons utilisé plusieurs méthodes apprises en cours notamment la technique de programmation orientée objet dans la mise en place de notre code, la conception des interfaces graphiques avec le Framework WPF (Windows Presentation Fundation) combiné à l’algorithmique des plus proches voisins KNN(K-Nearest Neighbors).

De ce fait, les technologies utilisées sont Visual Studio 2022 Community qui facilite grandement la performance de notre code en l’analysant et le simplifiant ainsi que la librairie externe CSVHelper que nous avons installée dans les Tools avec le gestionnaire Nuget pour l’upload des fichiers. En plus nous avons opté pour le design pattern **MVVM** pour organiser les éléments de notre logiciel avec chacun son rôle. C’est fondamental pour structurer notre programme.

Une image contenant texte, capture d’écran, moniteur, noir

Description générée automatiquement

Figure - Structure de notre programme avec MVVM

Si vous voulez apprendre à coder, le meilleur moyen c’est d’écrire le plus possible de code. C’est ce que nous tenter de faire ici. VS Community est un outil puissant qui intègre plusieurs langages POO notamment en Csharp où des milliers d’applications roulent.

Le langage de programmation C# vous permet de créer de nombreux types d’applications, par exemple :

* Applications métiers pour capturer, analyser et traiter les données
* Applications web dynamiques accessibles à partir d’un navigateur web
* Jeux 2D et 3D
* Applications financières et scientifiques
* Applications cloud
* Applications mobiles

Pour revenir à notre projet

L’objectif dans un premier temps est pouvoir ajouter son profil et puis se connecter en tant que médecin et de pouvoir ensuite les modifier.

—

Nous avons réalisé au second volet le survol entre les onglets *Informations*, *Diagnostique* et *Configuration* *IA.*

L’onglet *Diagnostique* contient un bouton oùnous avons la possibilité d’ajouter un patient sans pouvoir modifier ses informations.

Ce sont ses informations (valeurs, texte, combobox) qui seront ajoutées pour faire le diagnostic. L’objectif est d’appliquer le KNN pour avoir le résultat.

Sur l’onglet *Configuration IA,* nous avons travaillé sur les paramètres KNN ainsi que les boutons d’importations des fichiers train.csv et test.csv pour la classification et l’évaluation. Nous utilisons une technique d’apprentissage supervisé. Pour ceux qui ont plongé la main dedans, l’intelligence artificielle ou le modèle Learning a pour but de faire des prédictions tout simplement.

Nous avons implémenté KNN combiné avec un algorithme de tri appelé tri Shell pour trouver le taux de reconnaissance. Cela a été possible en appliquant une évaluation de performance de l’algorithme en utilisant les données de références et d’évaluation.

# Modélisation UML

Dans la modélisation UML de notre projet, nous avons recensé les classes et entités qui entrent en jeu. Nous avons :

1. La classe Médecin
2. La classe Patient
3. La classe Diagnostic qui implémente IDiagnostic
4. La classe KNN
5. La classe IDiagnostic (ajouter les attributs)

# Aperçu de l’application (GUI)

Notre application destinée aux médecins dispose de plusieurs interfaces graphiques interconnectées conformément à notre énoncé. Nous avons ajouté au fur et à mesure d’autres interfaces selon le besoin. Voyons la description :

***Interface de Connexion***

Cette interface a été créée dans Views et organiser grâce aux **StackPanel** et **Grid.** Nous avons créé la première interface graphique WPF où nous avons la possibilité d’ajouter le profil du médecin afin qu’il puisse se connecter en tant que médecin et de pouvoir ensuite modifier ses informations.

Cette interface est binder avec la fenêtre *Créer un compte* et l’interface *Accueil* regroupant les 3 onglets.

Une image contenant table

Description générée automatiquement

Figure 2 - Interface de Connexion

Sur nos deux boutons, nous avons fait un binding à deux variables (Connect & CreateAccount) que nous avons créé dans le ViewModel.

Le premier aspect consiste à créer un compte (le profil du médecin) afin de remplir ses informations. Puis, vous aurez la possibilité d’ajouter le médecin puis vous connecter.

Une image contenant texte

Description générée automatiquementVoici un extrait du code.

Figure - extrait code pour la connexion

Une fois connecté, vous allez sur l’interface Accueil.

***L’interface Accueil.***

Elle représente l’interface de « Bienvenue » où nous avons trois 3 onglets : *informations, diagnostique, configuration IA. Voici la capture.*

*Une image contenant texte

Description générée automatiquement*

Figure - Accueil

Sur l’onglet ***Informations*** nous avons les informations du médecin où nous avons la possibilité de modifier ou annuler.

Sur l’onglet ***Diagnostique*** où nous avons des boutons pour trouver les informations sur le patient, ajouter un patient sans pouvoir modifier ses informations, définir le type de douleur thoracique, Thalassémie dans des Combobox. Ce sont ses informations (valeurs, texte, combobox) qui seront ajoutées pour faire le diagnostic.

Définir aussi le nombre de dépressions ST induites et le nombre de gros vaisseaux. Voir la capture.

Une image contenant texte

Description générée automatiquement

Figure - Onglet Diagnostique

Sur l’onglet ***Configuration IA*** où nous avons deux boutons pour ajouter nos deux fichiers ***train.csv*** et ***test.csv.*** Il suffit de cliquer sur Parcourir et les charger depuis votre ordinateur. Aussi les paramètres K et le type de distance soient *Manhattan* ou *Euclidienne*.

Une image contenant texte

Description générée automatiquement

Figure - Onglet Configuration IA

À faire : sur l’onglet Configuration IA, une fois cliquer sur évaluer, qu’est ce qui doit se passer ? Ça donne le résultat du taux de reconnaissance en pourcentage.

À faire : faire fonctionner le bouton *Informations, ajouter patient, Diagnostiquer* dans l’onglet **Diagnostique (Diagnostic classe et IDiagnostic)**.

*Interface Créer Patient*

# Algorithme du -NN et Résultats

L’implémentation de l’algorithme KNN connu sous le nom de K-Nearest Neighbors – les plus proches voisins – a connu plusieurs étapes.

KNN est un algorithme qui utilise une technique simple et a pour objectif est de faire des prédictions. Les données d'entrainement sont conservées, puis on localise les K enregistrements dedans les plus similaires, puis à partir de ses voisins on fait une prédiction.

Comment calculer la distance euclidienne pour un échantillon donné ?

Comment calculer le Manhattan ?

**Implémenter KNN en 3 étapes**

**Étape 1 : Calculer la distance euclidienne entre deux lignes**

**Étape 2 : Localiser les voisins les plus proches**

**Étape 3 - faire des prédictions**

Nous utiliserons les voisins similaires collectés à partir de notre dataset d'apprentissage pour faire la prédiction.

# Problèmes et Difficultés rencontrées

Comme tout autre projet, dans la réalisation de notre projet (TP1), nous nous sommes confrontés à plusieurs problèmes, c’est le cas de

* L’interconnexion des interfaces avec le binding
* Le survol d’une fenêtre à une autre

Malgré les difficultés, les occupations, nous avons pu nous dépasser pour aboutir à ce résultat. La documentation sur internet en a largement contribué.

# Conclusion

Nous remercions, M. Yacine, pour ce TP1 qui nous fait largement découvrir le contour de ce langage ainsi l’outil WPF pour la conception des interfaces graphiques qui demandent beaucoup de soins et de concentration.

*À faire une vidéo de présentation*

# Références

**CsvHelper:** https://joshclose.github.io/CsvHelp