Implémentation d'une Interface Graphique pour WEKA
Rapport Technique

CHERIBET CHERIF Chouaib

31 décembre 2024

1 Technologies et Outils Utilisés

1.1 Technologies Principales

- **C# (.NET Framework)** : Choisi pour sa robustesse et son écosystème riche en outils de développement d'interfaces graphiques.
- **WEKA (Waikato Environment for Knowledge Analysis)** : Bibliothèque de référence en apprentissage automatique, fournie sous forme de fichier JAR, offrant une large gamme d'algorithmes de classification et de clustering.
- **Java Runtime Environment (JRE)** : Environnement requis pour l'exécution de WEKA.

1.2 Architecture d'Intégration

L'intégration avec WEKA est réalisée via l'exécution de processus système :

```
public class WekaWrapper
{
    private Process CreateWekaProcess(string arguments)
    {
        ProcessStartInfo startInfo = new ProcessStartInfo
        {
            FileName = "java",
            Arguments = $"-jar weka.jar {arguments}",
            RedirectStandardOutput = true,
            RedirectStandardError = true,
            UseShellExecute = false,
            CreateNoWindow = true
        };

        return Process.Start(startInfo);
```

```
    }
}
```

1.3 Outils de Développement

— **Visual Studio** : Environnement de développement intégré (IDE) utilisé pour le développement C#.
— **Interface en Ligne de Commande (CLI)** : Utilisée pour interagir avec WEKA via des commandes Java.

2 Présentation de l'Application

2.1 Splash Screen

Splash Screen conçu spécialement pour ce projet.

FIGURE 1 – Splash Screen + Logo

2.2 Interface Principale

L'interface principale de l'application comprend plusieurs zones fonctionnelles :

— **Bouton de Chargement des Données** : Permet l'importation de fichiers ARFF.
— **Stack Panel** : Permet de naviguer entre les différentes pages.
— **Éditeur de Texte** : Utilisé pour modifier le contenu des fichiers ARFF.
— **Bouton Go** : Permet d'accéder à la page de sélection des algorithmes.

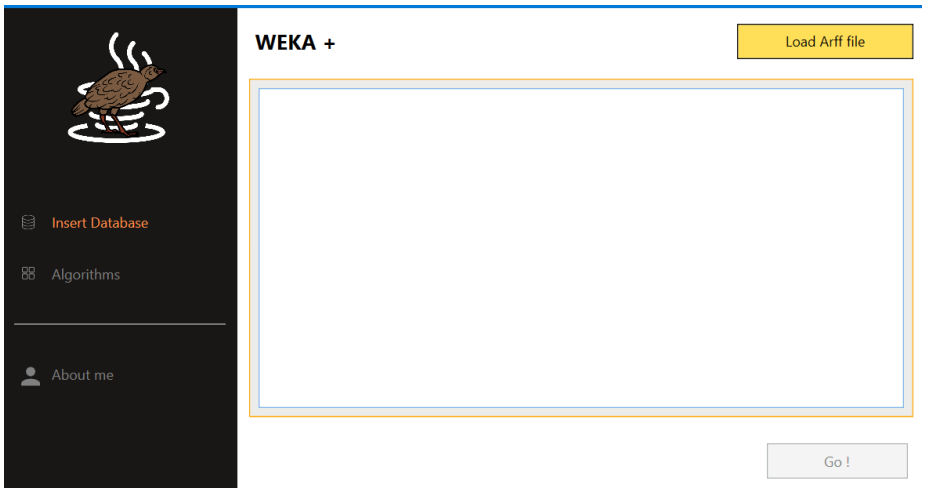

FIGURE 2 – Fenêtre principale de l'interface graphique

2.3 Exécution des Algorithmes

Cette page est conçue pour sélectionner les algorithmes et saisir les paramètres.

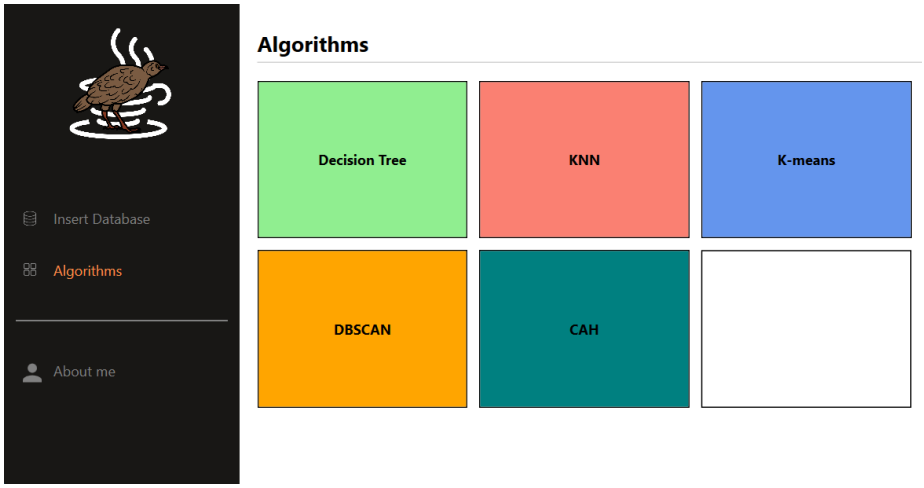

FIGURE 3 – Sélection des algorithmes

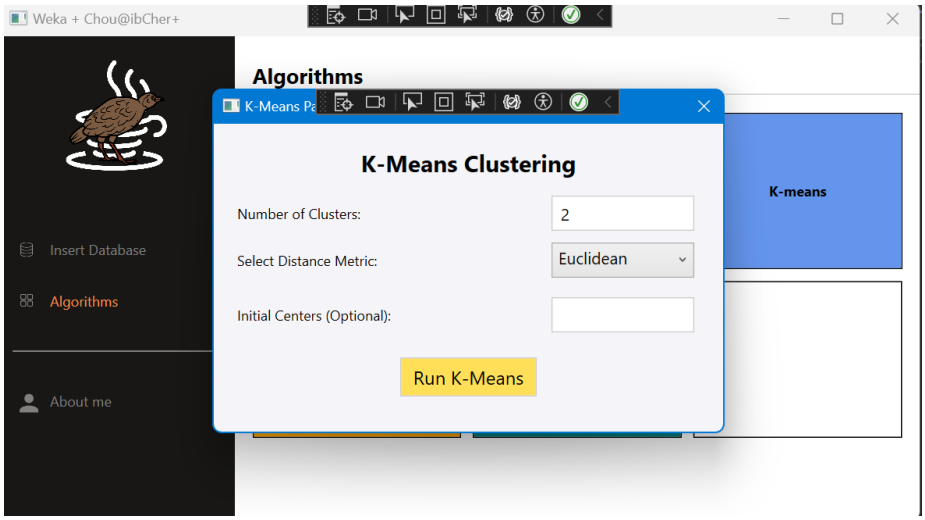

FIGURE 4 – Fenêtre pour saisir les paramètres

La communication avec WEKA se fait via des commandes système :

```
public string ExecuteWekaCommand(string algorithm, string dataFile)
{
    string arguments = $"weka.classifiers.{algorithm} -t {dataFile}";
    using (Process wekaProcess = CreateWekaProcess(arguments))
    {
        string output = wekaProcess.StandardOutput.ReadToEnd();
        wekaProcess.WaitForExit();
        return output;
    }
}
```

2.4 Visualisation des Résultats

Pour les algorithmes J48 et CAH, la visualisation est codée en Java Swing (TreeVisualizer and HierarchyVisualizer). Le fichier ARFF est passé en tant qu'argument, ainsi que d'autres arguments relatifs aux algorithmes. Un fichier JAR est ensuite créé pour chacun d'eux. Ces fichiers JAR sont appelés depuis le code C, avec l'ARFF passé en tant qu'argument.

L'interface de résultats présente :

— Les résultats bruts de l'exécution WEKA.

— Un parsing et une mise en forme des résultats.

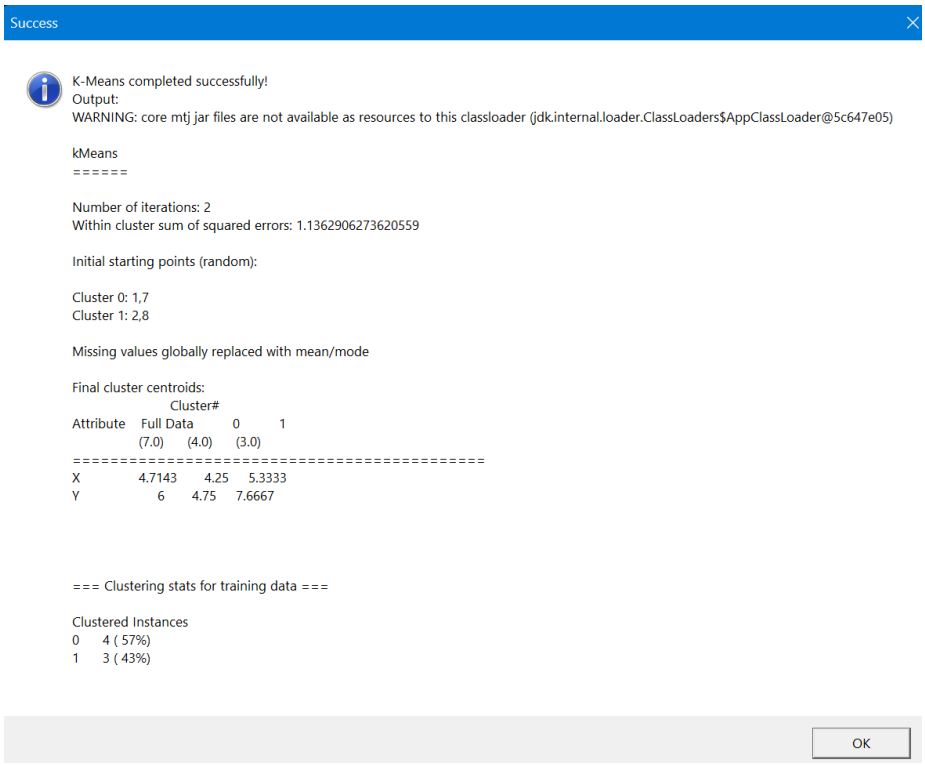

FIGURE 5 – Exemple de sortie de l'exécution de k-means

3 Propositions d'Améliorations

3.1 Améliorations Fonctionnelles

- **Utilisation d'IKVM** : Permet de convertir des bibliothèques Java en bibliothèques .NET utilisables directement dans un projet C. Cela éliminerait la nécessité d'exécuter des processus Java séparés.
- **Cache des Résultats** : Mise en cache des résultats pour éviter des calculs redondants.

3.2 Améliorations Techniques

- **Gestion des Erreurs** :
  - Améliorer la capture et l'affichage des erreurs Java.
  - Implémenter des mécanismes de retry pour les commandes échouées.
- **Interface Utilisateur** :
  - Ajouter des indicateurs de progression pendant l'exécution.
  - Permettre l'annulation des processus en cours.

3.3 Nouvelles Fonctionnalités

— **Configuration Avancée** : Interface de configuration du chemin Java et gestion des arguments JVM.
— **Traitement par Lots** : Exécution séquentielle de plusieurs algorithmes et export des résultats.

4 Utilisation de WPF et MVVM

Pour améliorer l'interface utilisateur, **Windows Presentation Foundation (WPF)** a été utilisé en suivant le modèle **Model-View-ViewModel (MVVM)**. Cela permet une conception d'interface utilisateur moderne et réactive.

5 Travail avec Java Swing

En parallèle, j'ai également développé une interface graphique en utilisant **Java Swing**. Cette expérience m'a permis de comparer les deux technologies et de mieux comprendre leurs avantages respectifs. De plus, j'ai aidé mes collègues en leur fournissant un projet de départ avec toutes les bibliothèques nécessaires, facilitant ainsi leur propre développement.

6 Conclusion

Cette implémentation d'une interface graphique pour WEKA utilisant l'exécution de processus système offre une solution pratique pour l'utilisation des algorithmes WEKA dans un environnement .NET. Les améliorations proposées permettraient d'optimiser l'expérience utilisateur et d'étendre les fonctionnalités de l'application. L'utilisation de WPF et MVVM a permis de créer une interface utilisateur moderne et réactive, tandis que l'expérience avec Java Swing a enrichi notre compréhension des différentes technologies disponibles pour le développement d'interfaces graphiques.