|  |
| --- |
| S3.02 –  Développement d’application  Rendu - Classification |
| BUT Info S3 – IUT de Lille Nathan Hallez  Alexandre Herssens  Maxence Stiévenard  Rémi Vautier |



# Sommaire

[Équipe J2 - *Belamcanda* 3](#_Toc120305607)

[Nathan Hallez 3](#_Toc120305608)

[Alexandre Herssens 3](#_Toc120305609)

[Maxence Stiévenard 3](#_Toc120305610)

[Rémi Vautier 3](#_Toc120305611)

[Analyse des données 4](#_Toc120305612)

[Typage des données 4](#_Toc120305613)

[Énumérations créées 4](#_Toc120305614)

[Calcul des distances 4](#_Toc120305615)

[Normalisateurs 4](#_Toc120305616)

[Comparateurs 4](#_Toc120305617)

[Implémentation de k-NN 5](#_Toc120305618)

[Robustesse de vos modèles 6](#_Toc120305619)

## Équipe J2 - *Belamcanda*

### Nathan Hallez

LOREM ISPUM hihihi

### Alexandre Herssens

LOREM ISPUM hihihi

### Maxence Stiévenard

LOREM ISPUM hihihi

### Rémi Vautier

LOREM ISPUM hihihi

## Analyse des données

Dans cette partie, vous devez relater tout ce qui concerne le chargement et la préparation des données pour vos deux problèmes: présentation des types de données, plages de valeurs et/ou d’énumération, détails sur les distances utilisées et la normalisation éventuellement faite.

### Typage des données

Après avoir consulté toutes les données qui nous ont été fournies dans un tableur, les champs numériques et textuels que nous avons considérés comme basiques, se sont vu attribuer les types **int** pour les entiers, **double** pour les décimaux ainsi que le type **String** pour les chaînes de caractères.

Pour les champs textuels où le type était moins évident, nous nous sommes aidés d’un **filtre**.

S’il n’y avait qu’un nombre limité de valeurs se répétant, nous avons choisi l’option de **l’énumération**, et pour les champs contenant seulement des valeurs uniques, nous avons attribué le type **String**.

#### Énumérations créées

Comme expliqué ci-dessus, nous avons créé des énumérations afin de charger et gérer les attributs des données à classifier à l’aide de notre programme.

Commençons par les **Iris** où nous avons eu besoin d’une unique énumération pour définir la variété : **IrisVariety.** Cette énumération définit les 3 variétés identifiées dans le jeu de données des iris : *SETOSA, VERSICOLOR* et *VIRGINICA.*

Nous nous sommes ensuite intéressés aux données du **Titanic** pour lesquelles nous avons défini deux énumérations, la première concernant le genre des passagers (**Gender**) et qui prend soit les valeurs *MALE* ou *FEMALE*.

La seconde énumération, nommée **Embarked** définissant le quai d’embarquement des passagers : *S*, *Q*, *C* ou *NULL.*

### Calcul des distances

Afin de calculer la distance entre deux points, ici des iris ou des passagers du Titanic, nous avons le choix entre deux types de distance : la distance **Euclidienne** et la distance de **Manhattan** qui somment à leur manière les différences obtenues grâce à aux comparateur définis pour chacune des colonnes.

Nous avons ainsi délégué la différence pour un attribut entre deux points aux **colonnes** : lorsqu’un jeu de données est défini, nous préconfigurons les colonnes avec un **normalisateur** et **comparateur** qui effectue la différence.

#### Normalisateurs

Ici aprler des normalisateurs

#### Comparateurs

Aprler des pseudo comparateurs

## Implémentation de k-NN

Dans cette partie vous expliquerez votre implémentation de k-NN, et vous focaliserez sur toute optimisation effective que vous avez mise en œuvre dans votre projet.

## Robustesse de vos modèles

Dans cette partie, vous devez évaluer la robustesse de vos modèles et détaillant votre protocole pour l’évaluer. Vous aurez au minimum un scénario d’exécution de votre code qui charge vos données et sort dans la console toutes les valeurs qui apparaissent dans cette section.