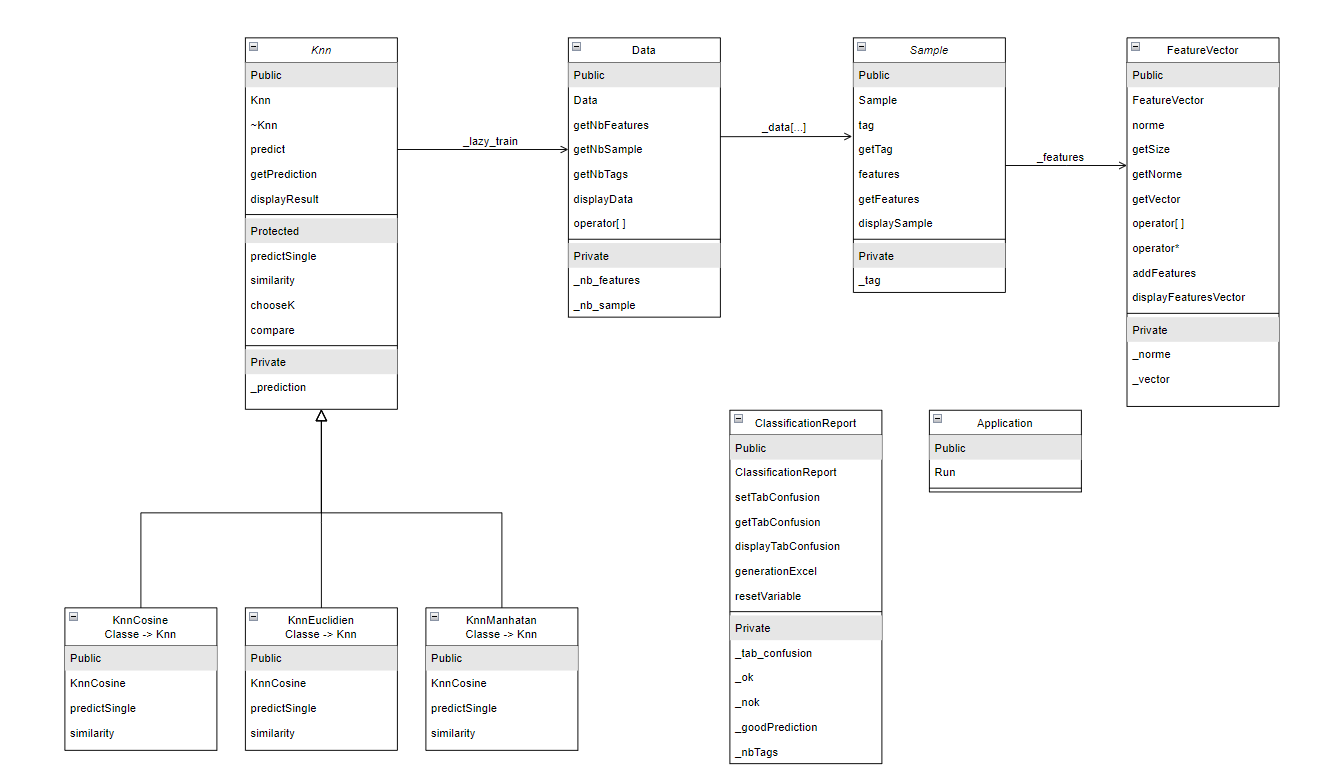
**Présenté par :** Noah JACOB, Lucas GUYOT, Jerôme KAUFFMANN, Antoine Le Tallec

**Année :** 2023-2024

**Projet Classifieur** Algorithme des K plus proches voisins

# Introduction



# Organisation des data

Avant de commencer à coder la logique, il est important de ranger de manière ordonnée les données qui nous sont fourni. Elles se présentent sous la forme d’un fichier .svm, dans les premières lignes on retrouve le nombre de d’échantillon ainsi que le nombres de caractéristiques qui les décrivent. Chaque ligne représente un élément, qui peut être connu si c’est un fichier de test. Alors un tag lui est attribué en début de lignes.

Dans notre code on retrouve l’organisation du fichier. Plusieurs classes sont utilisées pour représenter cette organisation. La classe Data qui englobe toutes les informations du fichier et surtout un tableau qui dans chaque case est stocké un objet de type Sample. Sample représente une ligne du fichier, les objets de cette classe seront comparés entre eux pour déterminer le tag inconnu de certain. Dans la classe Sample on retrouve un objet vecteur de caractéristique codé par la classe FeatureVector. Dans cette classe on retrouve un tableau qui stock chaque caractéristique de l’échantillon.

L’organisation des données reste très proche de celle proposé par le sujet, néanmoins nous avons pris quelques libertés et codé des fonctions supplémentaires ou au contraire passé outre celle proposé. Par exemple la classe Data ne possèdent pas de fonction load pour charger les données. En revanche les données sont chargées lors de la construction de l’objet grâce au chemin vers le fichier et une variable qui dit si oui ou non les données sont utilisées pour l’entrainement.

# Traitement des données

# Interprétation des données

# Détermination du meilleur k

Pour déterminer le meilleur K à utiliser nous avons voulu essayer deux méthodes différentes. La première méthode consiste à donner le pourcentage de confiance de la prédiction donné. La seconde méthode est d’utilisé un autre paquet de données que l’on connaît et de regarder le nombre de bonne prédiction.

Il faut répéter ces deux méthodes avec un multitude de K différents et regarder pour quel K nous obtenons les meilleurs résultats

### Pourcentage de confiance

### Nombre de bonne prédiction

A partir de la classe ClassificationReport nous possédons déjà le pourcentage de bonne prédiction pour une valeur de K et une méthode. L’objectif est donc de récupérer cette information pour plusieurs K différents et afin de pouvoir déterminer le meilleur K nous voulons extraire les résultats dans un fichier CSV pour Excel.

Nous avons donc implémenté une fonction en plus dans la classe ClassificationReport pour générer le fichier. Elle fonctionne et génère un fichier CSV mais manque d’optimisation. Actuellement l’architecture nous fait réorganiser les échantillons pour chaque K alors que ce n’est pas nécessaire. Par manque de temps nous avons décidé de laisser ce problème d’optimisation et de revenir dessus sur notre temps libre.

D’après le graphique la meilleur valeur de

# Application

Dans l’objectif de rendre l’expérience utilisateur meilleur nous avons voulu rajouter une classe Application. Cette classe ne contient qu’une fonction, elle permet d’afficher dans la console des menus et de suivre un déroulement logique pour rentrer les différentes données nécessaires au bon fonctionnement du programme.