Rapport CLASSIFICATION k-NN pour la SAÉ S3.02

Groupe G6

Membres : - Maxime Bimont
- Bastien Warnier
- Pierre Foulon
- Arthur Debacq

Analyse des données

Afin d'implémenter l'algorithme de classification nous avons dû définir les données à utiliser. Ces données sont chargées à l'aide de la classe ChargementDonneesUtil qui contient deux méthodes par Objet (Pokemon, Iris, Titanic). La première permet de charger le fichier CSV avec le chemin d'accès à celui-ci, ensuite on appelle la seconde méthode du même nom qui utilise la librairie OpenCSV qui permet de lire les données stockées dans un fichier sous le format CSV.
Nous avons essayé de créer une méthode qui vérifie le nom des colonnes et qui permet donc de charger n'importe quel CSV et le rediriger vers la bonne méthode, malheureusement nous n'avons pas réussi à nous en servir. Nous faisons donc appel directement à la méthode correspondante au bon Objet lors du chargement du CSV.

Pokémon :

En ce qui concerne les Pokémons, les caractéristiques utilisées sont la durée d'éclosion, le taux de capture, la quantité d'expérience nécessaire à l'évolution et la vitesse du pokémon afin de déterminer si le pokémon est légendaire ou non. Toutes ces données sont des valeurs numériques.
Pour chacune de ces valeurs, chaque pokémon est étudié afin de comparer leurs caractéristiques et déterminer la valeur minimale et maximale dans la liste. L'amplitude trouvée en effectuant la soustraction de la valeur maximale avec la valeur minimale dans chaque caractéristique, nous permet de normaliser pour éviter l'hétérogénéité des valeurs. Cette normalisation se calcule de la façon suivante, en admettant deux pokémons différents p_1 et p_2 :

$$\frac{p_1.donnée - p_2.donnée}{amplitude\ de\ la\ donnée}$$

Par exemple, la normalisation de la vitesse sera calculée de la manière suivante :

$$\frac{p_1.speed - p_2.speed}{amplSpeed}$$

Iris :

Du côté des Iris, les données sont majoritairement de type numérique (double) car on y retrouve les caractéristique physique de l'iris tel que la longueur du sépale, sa largeur, la longueur de la pétale ou encore sa largeur à l'exception de la variété ou cette donnée sera sous forme d'une chaîne de caractère et est donc un type énuméré constitué de Versicolor, Setosa et Virginica. Afin de calculer les distances entre deux Iris nous calculons au préalable les amplitudes de chaque données, variété exclu, en effectuant la soustraction du max avec le min de cette manière nous pouvons normaliser nos données afin d'éviter la disparité entre les valeurs et avoir une échelle commune entre les données, considérons notre premier iris i^1 et le deuxième i^2 la normalisation de nos données se fera alors sous cette forme :

$$\frac{i_1.donnée - i_2.donnée}{amplitude\ de\ la\ donnée}$$

Par exemple le calcul de normalisation de la donnée longueur du sépale ça sera donc :
(i2.sepal_length - i1.sepal_length) / amplitude_sepal_length

$$\frac{i_1.sepal\_length - i_2.sepal\_length}{amplsepal\_length}$$

Titanic :

Pour les passagers du Titanic, les données utilisées sont numériques de type double. Les caractéristiques qui servent au calcul de distance sont pClass qui correspond à la classe de siège sur le bateau, cette valeur est comprise entre 1 et 3, où 1 = 1ère classe, etc… La seconde caractéristique (age) utilisée est l'âge du passager, sibSp indique le nombre de frères et soeurs et/ou le nombre de conjoint(e), parch montre le nombre de parents/enfants à bord et fare correspond au prix du ticket qui est calculé par rapport au nombre de personne c'est à dire les frères et soeurs et/ou conjoint(e), ainsi que la classe à bord pour chacun des passagers sur une réservation.
La normalisation de chacune des valeurs entre 0 et 1, en admettant qu'il existe deux passagers du titanic t_1 et t_2, la normalisation entre ces deux passagers se traduit par :

$$\frac{t_1.donnée - t_2.donnée}{amplitude\ de\ la\ donnée}$$

La normalisation entre les deux passagers par rapport à leur âge se fait sous cette forme:

$$\frac{t_1.age - t_2.age}{amplAge}$$

Ainsi selon la méthode de calcul de distance que nous désirons nous ferons la somme des données normalisé soit au carré et racine de la somme pour la distance Euclidienne

Pour la distance Manhattan on calcule la somme des valeurs absolues.
Nous avons implémenté les deux distances mais nous utilisons la distance Euclidienne arrondie à 2 chiffres après la virgule.

Implémentation de k-NN

Dans le but de l'implémentation de la méthode k-NN, nous avons créé une classe MethodeKnn dédiée afin d'y réaliser les méthodes de calculs des n plus proches voisins et les méthodes d'affectation d'une classe à l'instance et ça pour chaque type.

Ainsi la méthode de calcul des n plus proches voisins prend en paramètre le nombre de voisins que l'on désire ainsi qu'un objet Pokémon, Iris ou Titanic et renvoie une liste d'objet Pokémon, Iris ou Titanic avec comme taille le nombre de n voisins les plus proches voulus.

Pour nous aider à implémenter cette méthode nous avons créé une classe comparator pour chaque type objet, (DistanceComparatorPokemon, DistanceComparatorIris et DistanceComparatorTitanic) ces classes comparator vont implémenter chacune une méthode compare qui va nous servir à comparer la distance entre deux objet par rapport à un troisième objet Pokémon/Iris/Titanic, suite à ça nous utilisons sur notre liste de Pokémon/Iris/Titanic la méthode "sort" qui triera celle-ci dans l'ordre que le DistanceComparator retournera. Une fois la liste triée, les Objets les plus proches seront ceux avec une distance la plus proche de celle de notre instance auquelle on désire affecter une classe ainsi il ne nous reste plus qu'à prendre le nombre de voisins en appelant la méthode "subList" qui retournera une portion de la liste jusqu'à l'indice voulu, ici le nombre de voisins voulus.

De ces méthodes des N plus proches voisins, nous allons les invoquer dans les méthodes d'affectations de classe nommé respectivement isLegendary pour la classe Pokémon renvoyant un boolean, whoSurvive pour la classe Titanic renvoyant aussi un boolean et whatVariety pour la classe Iris qui renvoie une chaîne de caractères contrairement aux deux autres. Le principe est de parcourir la liste des N plus proches voisins en utilisant un forEach et de compter le nombre de voisins avec une classe différente et ensuite grâce au vote majoritaire affecter la classe la plus présente parmis le total des voisins. Prenons comme exemple notre méthode d'affectation pour la class Pokémon, isLegendary, celle ci va parcourir notre liste des plus proches voisins et va compter le nombre de pokémons légendaires voisins par rapport au nombre de pokémons commun voisins, si celui-ci est supérieur ou égale au nombre de pokémon commun alors le pokémon sera considéré comme de rareté légendaire sinon il sera commun. Le même principe s'applique aux classes TItanic et Iris.

Robustesse de vos modèles

Le but d'évaluer la robustesse est de déterminer à quel point l'apprentissage avec la méthode k-NN est efficace et robuste face à des données incomplètes. Pour cela nous avons 2 méthodes pour chaque objet Pokémon/Iris/Titanic. La première retourne le nombre de k voisins plus proche le plus efficace pour la classification. On commence par charger les données et définir deux variables précisionMax et kMax pour déterminer d'une part la justesse de la valeur qui en ressort et d'autre part le nombre de voisin avec lequel nous obtenons cette justesse.

Chaque entité de notre csv est testée et obtient un pourcentage de réussite, ce pourcentage est calculé grâce à la seconde méthode, si le pourcentage qui en ressort est supérieur à la précision obtenu avec un autre k différent, on garde ce nouveau nombre de k plus proche voisin et ce sera le nombre le plus optimal et le plus efficace pour la classification.

Pour les pokémons on obtient avec 5NN un pourcentage d'efficacité de 92%. N'ayant pas de données à tester pour les autres objets nous n'avons pas pu tester la robustesse de la méthode k-NN de manière optimale. Nous avons donc fait le choix de tester la robustesse sur le même fichier CSV. On obtient alors pour les Iris avec 375 plus proches voisins un pourcentage de 62% de robustesse et pour les passagers du Titanic, avec 90-NN un pourcentage de fiabilité de 97%