Rapport de Développement Efficace (SAE 3.02)

Équipe

G2

- Alexandre Legrand
- Aymane Benafquir
- Louis Beck
- Kylian Robin

Classification

Implémentation du k-NN :

- Nous avons implémenté l'algorithme k-NN dans la classe model/MethodeKnn.java notamment avec la methode knn qui prend en paramètres les attributs permettant la classification, les données de base pour pouvoir calculer la distance, le k choisi, le point à comparer et la distance choisie. Pour cela, nous avons créé une Map qui récupère les données et leur distance. L'utilisation d'une Map nous permet de ne pas avoir à faire des boucles de if qui vérifie chaque distance avec toutes les données.
- Pour le calcul des distances, nous avons créé les classes `DistanceManhattan`, `DistanceManhattanNormalisee`, `DistanceEuclidienne` et `DistanceEuclidienneNormalisee` qui implémentent l'interface Distance. Chaque classe implémente la methode `distance` qui calcule la distance entre 2 points avec les calculs adaptés.
- Pour la normalisation, nous avons simplement créé les classes des distances normalisées qui ont une méthode de calcul permettant de passer les points sur lesquels calculer la distance en paramètres et normalise leurs attributs tout en faisant le calcul de la distance. Pour cela, nous prenons un à un les attributs parmis ceux qui servent à effectuer la classification et nous les normalisons. Puis nous faisons le calcul avec les valeurs normalisées.
- Les méthodes servant à classifier se trouvent dans le controller (`ApplicationController` pour la classification aléatoire et `ClassifyController` pour la classification avec l'algorithme k-NN). Ces méthodes appellent les méthodes présentes dans le `DataModel`. La méthode `randomClassifyPoint` permet de classer un point aléatoirement en récuperant toutes les valeurs possibles pour la catégorie choisie et en sélectionne une au hasard pour l'affecter aux données ajoutées par l'utilisateur. La méthode `classifyPoint` permet de classer un point en utilisant l'algorithme k-NN, elle appelle la methode `knn` qui permet de récuperer les k plus proches voisins et de classer le point en fonction des catégories de ces voisins. En effet, une fois les k plus proches voisins trouvés nous allons créer une Map avec la catégorie de chaque voisin et son poids(poids calculé en fonction de sa distance, plus il est proche, plus il est élevé). Ensuite, nous allons récuperer la catégorie qui a le plus grand poids et affecter sa catégorie au point à classifier. Nous aurions pu utiliser le pattern strategy plutôt qu'une map afin d'utiliser la classe adaptée.
- Afin d'évaluer la robustesse du k-NN, nous avons utilisé les méthodes `robustesseKnn` et `testKnn`. La méthode `robustesseKnn` permet de récuperer (en paramètre) les données de base déjà classifiées, un k, les attributs permettant la classification, la categorie et l'algorithme de distance choisis. Cette méthode va séparer les données en plusieurs jeux de données (nous avons choisi d'en faire 3), va appeler la méthode `testKnn` pour chaque jeu de données et va renvoyer le nombre de données bien classifiées. La méthode `testKnn` prend en paramètre un des jeu de données (une List) et les autres jeux rassemblés (dans un Set), un k, les attributs permettant la classification, la categorie et l'algorithme de distance choisis. Pour chaque donnée du jeu fournit, elle cherche les voisins grâce à la méthode `knn` et vérifie si ils ont ou pas la bonne catégorie. Ensuite elle ajoute 1 au nombre de bonnes classifications si la catégorie est bonne et 0 sinon. Enfin elle renvoie le nombre de bonnes classifications.

Validation croisée :

Nous utilisons la méthode `robustesseKnn` avec plusieurs k, avec toutes les distances afin d'obtenir des résultats pour chaque distances et k. Ensuite nous récuperons le nombre de points bien classifiés et nous calculons le pourcentage de points bien classifiés. Pour ce faire, nous faisons le calcul suivant : `nbCorrect* 100 / getDataList().size()`. - Avec - nbCorrect, le nombre de points bien classifiés - getDataList().size(), le nombre de points totaux.

Choix du meilleur k :

Iris

- Manattan

CSV	k	distance	pourcentage de réussite		
Iris	3	Manhattan	98,65771812		
	5	Manhattan	97,98657718	meilleur pourcentage :	99,32885906
	7	Manhattan	98,65771812	k associé :	9 et 11
	9	Manhattan	99,32885906		
	11	Manhattan	99,32885906		
	13	Manhattan	97,31543624		
	15	Manhattan	97,31543624		

- Euclidienne

Iris	3	Euclidienne	98,65771812		
	5	Euclidienne	98,65771812		
	7	Euclidienne	98,65771812	meilleur pourcentage :	98,65771812
	9	Euclidienne	98,65771812	k associé :	3 à 13
	11	Euclidienne	98,65771812		
	13	Euclidienne	98,65771812		
	15	Euclidienne	97,98657718		

- Manattan Normalisée

Iris	3	ManhattanNormalisee	99,32885906		
	5	ManhattanNormalisee	97,98657718		
	7	ManhattanNormalisee	97,31543624	meilleur pourcentage :	99,32885906
	9	ManhattanNormalisee	98,65771812	k associé :	3
	11	ManhattanNormalisee	98,65771812		
	13	ManhattanNormalisee	97,31543624		
	15	ManhattanNormalisee	97,31543624		

- Euclidienne Normalisée

Iris	3	EuclidienneNormalisee	99,32885906		
	5	EuclidienneNormalisee	98,65771812		
	7	EuclidienneNormalisee	97,98657718	meilleur pourcentage :	99,32885906
	9	EuclidienneNormalisee	96,6442953	k associé :	3
	11	EuclidienneNormalisee	97,31543624		
	13	EuclidienneNormalisee	96,6442953		
	15	EuclidienneNormalisee	97,31543624		

Pokémon

Type 1 :

- Manattan

CSV	k	distance	pourcentage de réussite		
Pokemon	3	Manhattan	39,64497041		
TYPE1	5	Manhattan	24,65483235	meilleur pourcentage :	39,64497041
	7	Manhattan	15,58185404	k associé :	3
	9	Manhattan	10,45364892		
	11	Manhattan	9,467455621		
	13	Manhattan	6,311637081		
	15	Manhattan	3,747534517		

- Euclidienne

Pokemon	3	Euclidienne	39,44773176		
TYPE1	5	Euclidienne	24,85207101		
	7	Euclidienne	14,59566075	meilleur pourcentage :	39,44773176
	9	Euclidienne	10,0591716	k associé :	3
	11	Euclidienne	9,861932939		
	13	Euclidienne	6,903353057		
	15	Euclidienne	4,733727811		

- Manattan Normalisée

Pokemon	3	ManhattanNormalisee	32,74161736		
TYPE1	5	ManhattanNormalisee	16,37080868		
	7	ManhattanNormalisee	7,297830375	meilleur pourcentage :	32,74161736
	9	ManhattanNormalisee	4,142011834	k associé :	3
	11	ManhattanNormalisee	2,169625247		
	13	ManhattanNormalisee	1,972386588		
	15	ManhattanNormalisee	0,986193294		

- Euclidienne Normalisée

Pokemon	3	EuclidienneNormalisee	32,14990138		
TYPE1	5	EuclidienneNormalisee	14,00394477		
	7	EuclidienneNormalisee	6,903353057	meilleur pourcentage :	32,14990138
	9	EuclidienneNormalisee	4,536489152	k associé :	3
	11	EuclidienneNormalisee	1,775147929		
	13	EuclidienneNormalisee	1,380670611		
	15	EuclidienneNormalisee	0		

Type 2 :

- Manattan

CSV	k	distance	pourcentage de réussite			
Pokemon	3	Manhattan	57,98816568			
TYPE2	5	Manhattan	46,94280079		meilleur pourcentage :	57,98816568
	7	Manhattan	44,18145957		k associé :	3
	9	Manhattan	42,80078895			
	11	Manhattan	40,82840237			
	13	Manhattan	38,46153846			
	15	Manhattan	36,68639053			

- Euclidienne

Pokemon	3	Euclidienne	57,3964497			
TYPE2	5	Euclidienne	46,15384615			
	7	Euclidienne	44,37869822		meilleur pourcentage :	57,3964497
	9	Euclidienne	42,6035503		k associé :	3
	11	Euclidienne	39,64497041			
	13	Euclidienne	37,67258383			
	15	Euclidienne	36,48915187			

- Manattan Normalisée

Pokemon	3	ManhattanNormalisee	56,80473373			
TYPE2	5	ManhattanNormalisee	45,9566075			
	7	ManhattanNormalisee	41,22287968		meilleur pourcentage :	56,80473373
	9	ManhattanNormalisee	40,03944773		k associé :	3
	11	ManhattanNormalisee	37,27810651			
	13	ManhattanNormalisee	36,09467456			
	15	ManhattanNormalisee	34,9112426			

- Euclidienne Normalisée

Pokemon	3	EuclidienneNormalisee	54,0433925			
TYPE2	5	EuclidienneNormalisee	44,18145957			
	7	EuclidienneNormalisee	40,43392505		meilleur pourcentage :	54,0433925
	9	EuclidienneNormalisee	38,85601578		k associé :	3
	11	EuclidienneNormalisee	38,2642998			
	13	EuclidienneNormalisee	38,85601578			
	15	EuclidienneNormalisee	36,09467456			

isLegendary :

- Manattan

CSV	k	distance	pourcentage de réussite			
Pokemon	3	Manhattan	99,40828402			
isLegendary	5	Manhattan	98,81656805		meilleur pourcentage :	99,40828402
	7	Manhattan	98,42209073		k associé :	3
	9	Manhattan	98,42209073			
	11	Manhattan	98,42209073			
	13	Manhattan	98,42209073			
	15	Manhattan	98,42209073			

- Euclidienne

Pokemon	3	Euclidienne	99,60552268			
isLegendary	5	Euclidienne	98,81656805			
	7	Euclidienne	98,42209073		meilleur pourcentage :	99,60552268
	9	Euclidienne	98,42209073		k associé :	3
	11	Euclidienne	98,42209073			
	13	Euclidienne	98,42209073			
	15	Euclidienne	98,42209073			

- Manattan Normalisée

Pokemon	3	ManhattanNormalisee	99,60552268			
isLegendary	5	ManhattanNormalisee	99,01380671			
	7	ManhattanNormalisee	98,61932939		meilleur pourcentage :	99,60552268
	9	ManhattanNormalisee	98,61932939		k associé :	3
	11	ManhattanNormalisee	98,61932939			
	13	ManhattanNormalisee	98,61932939			
	15	ManhattanNormalisee	98,61932939			

- Euclidienne Normalisée

Pokemon	3	EuclidienneNormalisee	99,40828402			
isLegendary	5	EuclidienneNormalisee	99,01380671			
	7	EuclidienneNormalisee	98,81656805		meilleur pourcentage :	99,40828402
	9	EuclidienneNormalisee	98,81656805		k associé :	3
	11	EuclidienneNormalisee	98,81656805			
	13	EuclidienneNormalisee	98,81656805			
	15	EuclidienneNormalisee	98,81656805			

Conclusion sur le meilleur choix à faire :

Pour les Iris,

- le meilleur pourcentage de réussite est de 99,33% avec :
  - la distance euclidienne normalisée et un k de 3
  - la distance de manhattan normalisée et un k de 3
  - la distance de manhattan et un k de 9 ou 11

Pour les Pokemon,

Pour le type 1,

- le meilleur pourcentage de réussite est de 39,64% avec :
  - la distance de manhattan et un k de 3

Pour le type 2,

- le meilleur pourcentage de réussite est de 57,99% avec :
  - la distance de manhattan et un k de 3

Pour isLegendary,

- le meilleur pourcentage de réussite est de 99,61% avec :
  - la distance de manhattan normalisée et un k de 3
  - la distance de euclidienne et un k de 3

Conclusion :

Le k le plus efficace semble être k=3. Pour la distance, cela varie en fonction de la catégorie. La distance de manhattan semble plus efficace sur des catégories assez vastes (type 1 et 2) et la distance de manhattan normalisée semble être plus efficace sur des catégories peu vastes (isLegendary pour Pokemon et variety pour Iris).