L'algorithme KNN (k Nearest Neighbors – apprentissage supervisé)

1) Introduction

Les algorithmes de Machine Learning supervisés sont utilisés pour résoudre des problèmes de **classification** ou de **régression.**

- Un problème de **régression** a pour sortie un nombre réel (un nombre décimal avec une virgule). Par exemple, nous pourrions estimer le poids d'une personne en fonction de sa taille.
- Un problème de classification a une valeur discrète en sortie. Par exemple, les 2 intitulés "aime les films d'horreur" et "N'aime pas les films d'horreur" sont des données discrètes. Il n'y a pas de juste milieu.

2) Les k plus proches voisins (k Nearest Neighbors – KNN)

L'algorithme kNN suppose que des objets similaires existent à proximité. En d'autres termes, des éléments similaires sont proches les uns des autres.

L'algorithme des k plus proches voisins est l'un des algorithmes utilisés dans le domaine de l'intelligence artificielle. C'est un algorithme d'apprentissage automatique supervisé qui attribue une catégorie à un élément en fonction de la classe majoritaire de ses plus proches voisins dans l'échantillon d'entraînement. Son principe peut être résumé par cette phrase : Dis-moi qui sont tes amis et je te dirai qui tu es.

3) Principes

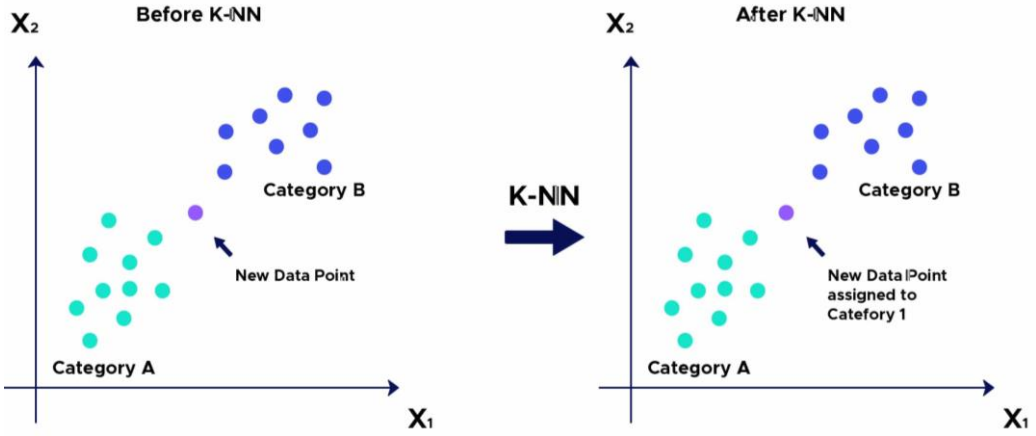
- L'algorithme des K plus proches voisins est un algorithme *d'apprentissage supervisé*.
- L'objectif de l'algorithme est de donner un *label* à des données.
- On dispose de données *labélisées* qui servent à l'apprentissage et à la mesure de qualité des prédictions.
- Une fois l'algorithme entrainé et testé on peut l'employer afin qu'il prédise le label d'une nouvelle donnée

4) L'algorithme KNN

1. Charger les données
2. Initialiser **k** au nombre de plus proches voisins choisi.
3. Pour chaque exemple dans les données :

3.1 Calculer la distance entre notre requête et l'observation itérative actuelle de la boucle depuis les données.
3.2 Ajouter la distance et l'indice de l'observation concernée à une collection ordonnée de données

4. Trier cette collection ordonnée contenant distances et indices de la plus petite distance à la plus grande (dans ordre croissant).
5. Sélectionner les k premières entrées de la collection de données triées (équivalent aux k plus proches voisins)
6. Obtenir les étiquettes des k entrées sélectionnées
7. Si régression, retourner la moyenne des k étiquettes.
8. Si classification, retourner le mode (valeur la plus fréquente/commune) des k étiquettes



The KNN algorithm uses various distance functions to compute the proximity amongst the data points.
There is no need to code these distances yourself, generally, Machine Learning libraries like Scikit Learn, perform these calculations internally. You just have to indicate the distance measure you want to use.

- **Euclidean distance:** calculates the square root of the sum of the square differences between the coordinates of two points. The formula says:

$$D_e(x,y) = \sqrt{\sum_{j=1}^{n}(x_j - y_j)^2}$$

Where: (xj, yj) are the coordinates of the points and De(x,y) is the distance between (x1, y1) and (x2, y2).

- **Manhattan distance**: calculates the sum of the absolute values of the differences between the coordinates of two points. The formula says:

$$D_m(x,y) = \sum_{i=1}^{k} \mid x_i - y_i \mid$$

- **Hamming distance**: the distance between two given points is the maximum difference between their coordinates on one dimension. The formula says:

$$D_h(x,y) = \sum_{i=1}^{k} \mid x_i - y_i \mid$$

If X=Y → D = 0
If X# Y → D = 1

Note that there are other distances depending on the use case of the algorithm, but the Euclidean distance remains the most used.

5) Choisir la bonne valeur pour k

Pour sélectionner la valeur de **k** qui convient à vos données, nous exécutons plusieurs fois l'algorithme KNN avec différentes valeurs de **k**. Puis nous choisissons le **k** qui réduit le nombre d'erreurs rencontrées tout en maintenant la capacité de l'algorithme à effectuer des prédictions avec précision lorsqu'il reçoit des données nouvelles (non vues auparavant).

Le choix de la valeur K à utiliser pour effectuer une prédiction avec KNN varie en fonction de l'ensemble de données. En général, moins on utilise de voisins (un petit nombre K), plus on est

sujet à un sous-ajustement (underfitting). En revanche, plus nous utilisons de voisins (un grand nombre K), plus notre prédiction sera fiable. Cependant, si nous utilisons un nombre K de voisins avec K=N et N étant le nombre d'observations, nous risquons d'avoir un overfitting et donc un modèle qui généralise mal sur des observations qu'il n'a pas encore vues.

Le principal inconvénient de KNN, qui est le ralentissement considérable à mesure que le volume de données augmente, en fait un choix peu pratique dans un contexte où les prédictions doivent être faites rapidement. De plus, certains algorithmes plus rapides peuvent produire des résultats de classification et de régression plus précis.

Exemple

On considère deux espèces, les *crocodiles* et les *alligators*. On suppose qu'on peut les distinguer en mesurant la largeur de leur gueule et la longueur de leur corps.

gueule	longueur	classe
0.17	2.84	alligator
0.24	3.82	alligator
0.24	3.39	alligator
0.2	2.60	alligator
0.25	4.21	crocodile
0.47	4.64	crocodile
0.47	4.48	crocodile
0.49	4.9	crocodile
0.46	4.08	Crocodile
0.19	2.91	?

On ajoute un nouvel animal dont on connait les caractéristiques **mais pas l'espèce** :

Nouveau : gueule = 0.19, longueur = 2.91 (animal)

À quelle espèce appartient-il ?

On complète les données précédentes avec la distance séparant cet animal des précédents.

On emploie ici une **distance Euclidienne usuelle** :

$$AB = \sqrt{ (x_B - x_A)^2 + (y_B - y_A)^2 }$$

gueule	longueur	classe
0.17	2.84	alligator
0.24	3.82	alligator
0.24	3.39	alligator
0.2	2.60	alligator
0.25	4.21	crocodile
0.47	4.64	crocodile
0.47	4.48	crocodile
0.49	4.9	crocodile

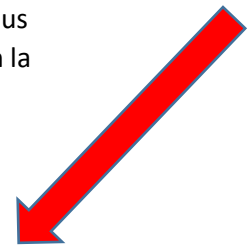
On calcule la distance de chaque observation avec notre animal (nouveau)

Distance	Classe
0.072	alligator
0.911	alligator
0.482	alligator
0.310	alligator
1.301	crocodile
1.752	crocodile
1.594	crocodile

Distance	Classe
2.012	crocodile
1.200	crocodile

Distance	Classe
0.072	alligator
0.310	alligator
0.482	alligator
0.911	alligator
1.200	crocodile
1.301	crocodile
1.594	crocodile
1.752	crocodile
2.012	crocodile

Ensuite on fait le tri de la valeur la plus petite à la plus grande selon la distance



Nouveau : gueule = 0.19, longueur = 2.91 (animal)

Choisissons **k=3**, on ne conserve que les trois animaux les plus proches de notre nouvel animal :

Distance	Classe
0.072	alligator
0.310	alligator
0.482	alligator

Dans ce sous ensemble composé de trois éléments, la classe majoritaire est alligator, et c'est celle qu'on attribue à notre nouvel animal. → On a choisi une classe car il s'agit d'un problème de classification.

Pour un problème de régression : par exemple prédire le poids d'une personne en fonction de la taille, on suit le même principe, seulement à la fin on calcule la moyenne des valeurs, regardons l'exemple suivant :

Taille	Poids
160	80
150	60
10	20
12	25
140	?

On veut connaitre le poids d'une personne dont la taille est 140.

Donc, on calcule la distance pour chaque observation par rapport au point (notre point est la taille 140 et on trie ces distances par ordre croissant, ensuite, on choisit les lignes en fonction de la valeur de K, ici k=2 (par exemple), donc, on aura les lignes en jaune.

Comme résultat, le poids est la moyenne des poids des deux lignes choisi (60+80) /2 =70

Donc 140 a pour poids 70

Distance	Poids
10	60
20	80
128	25
130	20

6) Domaine d'utilisation de KNN.

- Comparaison de personnes possédant des caractéristiques financières similaires pour accord de prêts bancaires.
- Elaboration d'un profil pour proposer aux abonnés des films appropriés.
- Classer un électeur potentiel dans les catégories «votera» ou «ne votera pas» pour tel ou tel candidat.

Cette liste est loin d'être exhaustive.

KNN : Basic idea, test and train data in python

In python, the KNN classification algorithm is implemented in the *KNeighborsClassifier* class in the neighbor's module. Before we can use the model, we need to instantiate the class into an object. This is when we will set any parameters of the model. The most important parameter of KNeighbor sClassifier is the number of neighbors.

Résumé

L'algorithme KNN est un algorithme d'apprentissage automatique supervisé simple qui peut être utilisé pour résoudre des problèmes de classification et de régression. Il est facile à mettre en œuvre et à comprendre, mais présente l'inconvénient majeur de ralentir considérablement à mesure que la taille des données utilisées augmente.

KNN recherche les distances entre une "inconnu" et toutes les données de la base d'apprentissage, sélectionne le nombre spécifié exemples (K) les plus proches de la requête, puis vote pour l'étiquette la plus fréquente (dans le cas de la classification) ou pour la moyenne des étiquettes (en le cas de la régression).

Dans le cas de la classification et de la régression, nous avons vu que choisir le bon K pour nos données se faisait en essayant plusieurs K et en choisissant celui qui fonctionnait le mieux.

Exercice

Appliquez l'algorithme des k (k = 3) plus proches voisins (regressor) sur la dataset iris en utilisant la métrique « euclidean ». Et calculez la précision de celui-ci.