UNIVERSITÉ NATIONALE DU VIETNAM À HANOÏ
INSTITUT FRANCOPHONE INTERNATIONAL

Option : Systèmes Intelligents et Multimédia (SIM)

Promotion : XXII

APPRENTISSAGE AUTOMATIQUE

TP1 : KNN ARBRE DE DÉCISION

Jean Claude SERUTI ZAGABE
Azaria ALLY SAIDI
Hugues MADIMBA KANDA
MASTER II

Encadrant :

Dr. Thanh-Nghi Do

Année académique 2018-2019

Table des matières

1 Introduction

1.1 Contexte

Apprentissage supervisé (supervised learning en anglais) est une technique où l'on cherche à produire automatiquement des règles à partir d'une base de données d'apprentissage contenant des exemples (en général des cas déjà traités et validés). Il en ressort de cet apprentissage différents types de méthodes parmi lesquelles nous avons : KNN et ARBRE DE DÉCISIONS qui feront l'objet de notre TP1.

1.2 Objectifs

Les objectifs générals de ce projet sont :

- Compréhension et implémentation de différentes méthodes d'apprentissage supervisé ;
- KNN et Arbre de décision ;

1.3 Subdivision du travail

Les différentes réponses au questions posées seront organisées comme suit :

- Knn (k plus proches voisins)
- Arbre de décision

2 KNN (k plus proches voisins)

2.1 Distance Manuelle L1

Soit la base d'individus :

X1	X2	Classe
0.376000	0.488000	0
0.312000	0.544000	0
0.298000	0.624000	0
0.394000	0.600000	0
0.506000	0.512000	0
0.488000	0.334000	1
0.478000	0.398000	1
0.606000	0.366000	1
0.428000	0.294000	1
0.542000	0.252000	1

FIGURE *1 – Base d'individu*

Nous allons procéder manuellement en calculant la métrique Euclidienne entre chaque nouvel individu et tous les individus de la base d'apprentissage. Avec q = 1, d distance de Manhattan d(u,v)=|u1-v1|+|u2-v2|+...+|un -vn|.

Ci-dessous le calcul de distance :

BASE D"INDIVIDUS						
X1	X2	Classe	**Distance1**	**Distance2**	**Distance3**	**Distance4**
0,3940	0,6000	0	**0,3920**	**0,2940**	**0,2120**	**0,0860**
0,5060	0,5120	0	**0,1920**	**0,0940**	**0,1120**	**0,1140**
0,3760	0,4880	0	**0,2980**	**0,2000**	**0,1180**	**0,1560**
0,3120	0,5440	0	**0,4180**	**0,3200**	**0,2380**	**0,1640**
0,4780	0,3980	1	**0,1060**	**0,1520**	**0,0740**	**0,2000**
0,2980	0,6240	0	**0,5120**	**0,4140**	**0,3320**	**0,2060**
0,4880	0,3340	1	**0,0920**	**0,2060**	**0,1480**	**0,2740**
0,4280	0,2940	1	**0,1920**	**0,3060**	**0,1840**	**0,2980**
0,6060	0,3660	1	**0,0580**	**0,1520**	**0,2340**	**0,3600**
0,5420	0,2520	1	**0,1200**	**0,2340**	**0,2840**	**0,4100**

FIGURE *2 – Distance de Manhattan*

Nous utilisons maintenant 1NN et 3NN pour classifier les individus suivants en fonction de la figure 2 :

CLASSIFICATIONS DES INDIVIDUS SUIVANT					
1NN			3NN		
X1	X2	**Classe**	X1	X2	**Classe**
0,5500	0,3640	**1**	0,5500	0,3640	**1**
0,5580	0,4700	**0**	0,5580	0,4700	**1**
0,4560	0,4500	**1**	0,4560	0,4500	**0**
0,4500	0,5700	**0**	0,4500	0,5700	**0**

FIGURE *3 – Classification des individus*

2.2 Implémentation d'algorithme k plus proches voisins en Python

Notre programme permet de classifier les individus d'une base de test a l'aide de l'algorithme des k plus proches voisins. Pour ce faire, il reçoit en entrée une base d'apprentissage, une base de test et un entier positif k indiquant le nombre de voisins a considérer. Après exécution de l'algorithme KNN, il restitue en sortie une matrice de confusion et un taux global de bon classement qui renseigne sur la performance du classement effectue.

La commande a exécuter pour lancer notre programme est de la forme : python try.py « Nom du fichier de la base d'apprentissage » « Nom du fichier de base du test » suivi du paramètre k. Lorsque cette commande est exécutée, le programme lit l'ensemble des données d'apprentissage et des données de test puis les stocke dans des tableaux. Par la suite, la similarité de chaque individus de l'ensemble de test est évaluée par le calcul de la distance entre ce dernier et tous les individus de l'ensemble d'entraînement. Les distances obtenues sont triées par ordre croissant et la classe majoritaire des k premiers individus est assignée au nouvel arrivant.

La figure 4 montre la procédure pratique et le résultat obtenu

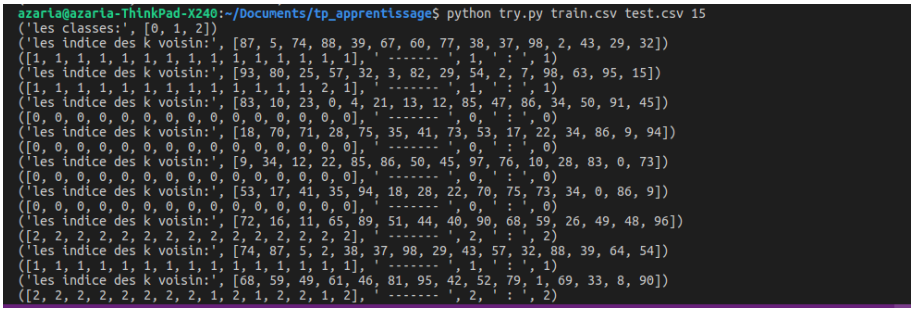

FIGURE *4 – 15 plus proches voisins sur l'ensemble de données Iris*

Après exécution du programme sur base d'apprentissage iris puis du test sur les données iris.

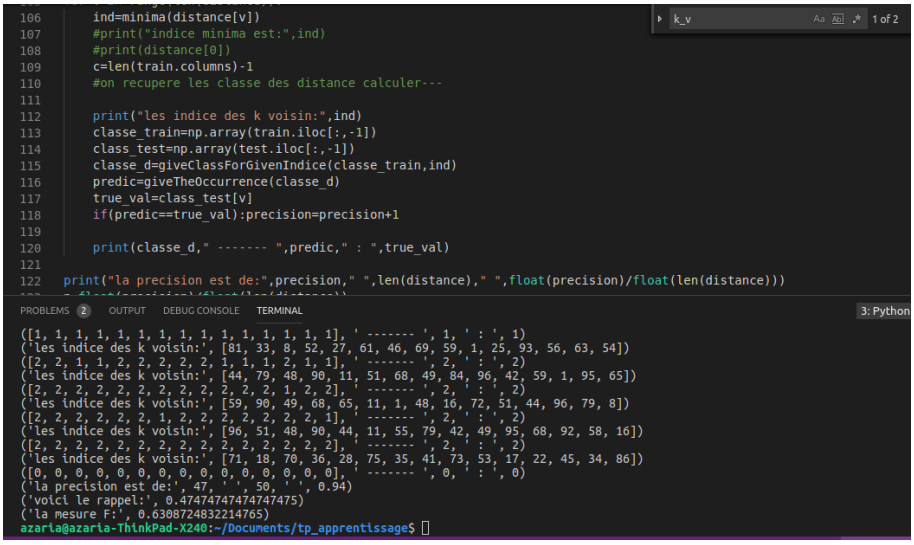

FIGURE *5 – 15 plus proches voisins sur l'ensemble de données Iris*

En sortie, nous obtenons le taux de bon classement avec une précision de 94%.
La figure 6 affiche en sortie en sortie un taux global de bon classement qui renseigne sur la performance du classement effectue avec une précision de 100% avec k= 5.

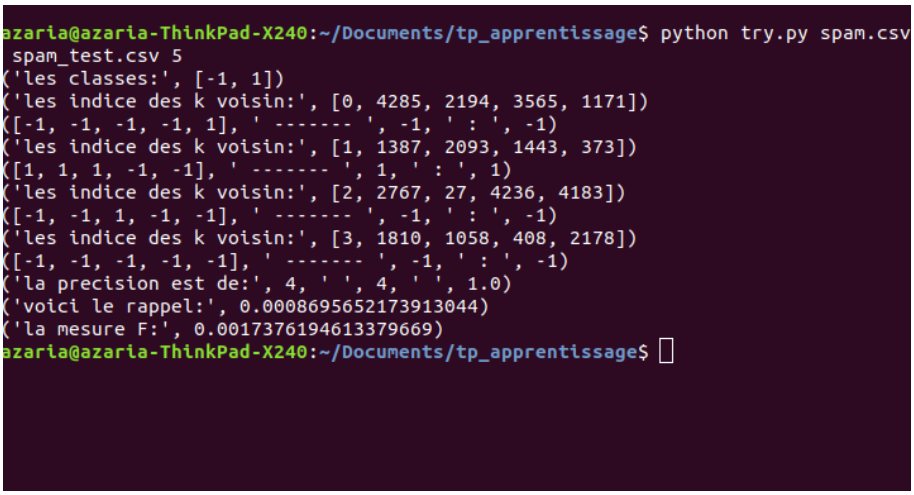

FIGURE *6 – 5 plus proches voisins sur l'ensemble de données Spam*

On exécution encore l'algorithme KNN, il restitue en sortie un taux global de bon classement qui renseigne sur la performance du classement effectue. La figure 7 illustré avec un jeu des données Spam montre le résultat obtenu avec un 15 proches voisins dont une précision de 100%.

Aucune influence sur les paramètres pour impacter la mesure de précision.

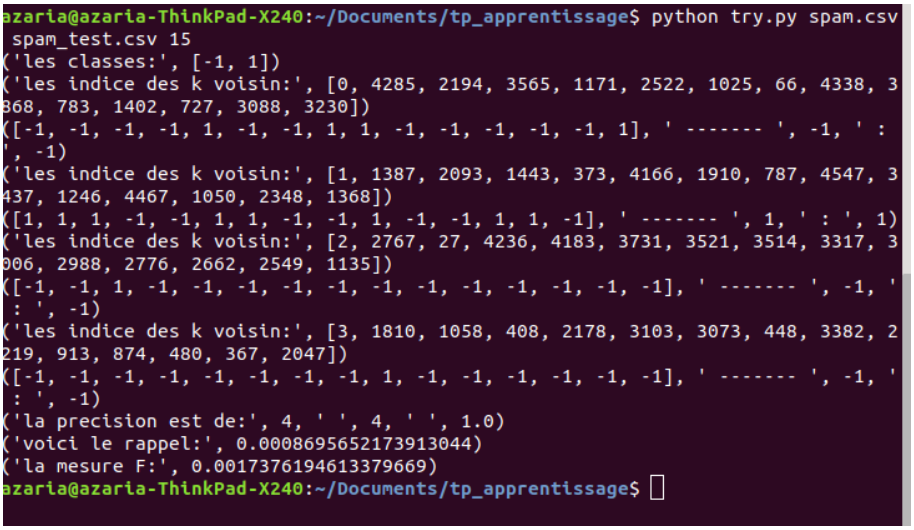

FIGURE *7 – 5 plus proches voisins sur l'ensemble de données Spam*

On expérimente aussi le jeu de données Optics (.trn pour l'apprentissage, .tst pour le test) avec le même programme pour classifier les ensembles de données comme le montre les différentes figures ci-desous.

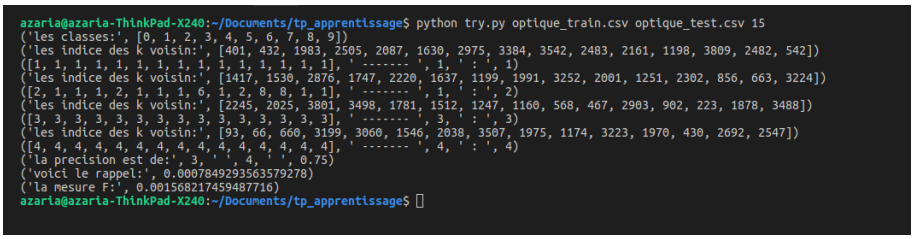

FIGURE *8 – 15 plus proches voisins sur 4 individus*

Avec k= 15 et 4 individus on a une précision de 75%.

La figure 9, avec k=15 et 20 individus on a une précision de 90%.

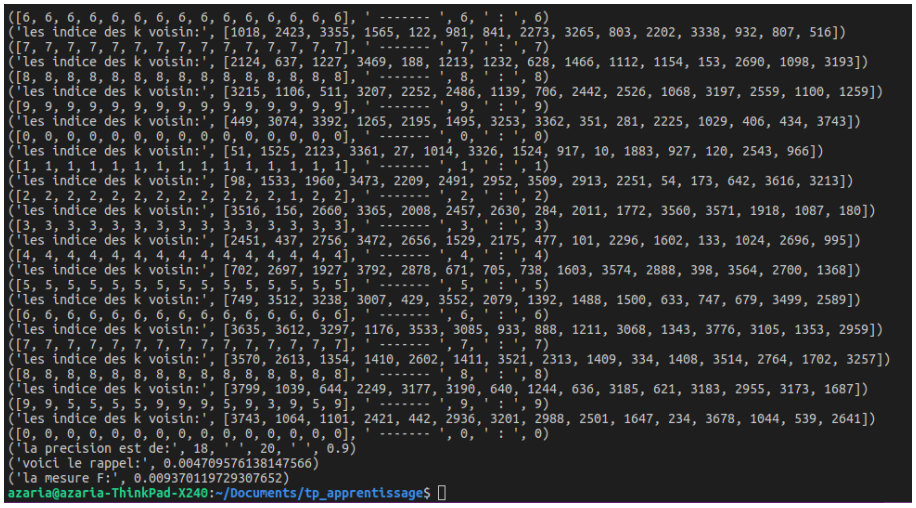

FIGURE *9 – 15 plus proches voisins sur 20 individus*

La figure 10, avec k=15 et 50 individus on a une précision de 96%.

```
([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], ' ------- ', 0, ' : ', 0)
('les indice des k voisin:', [3163, 590, 2525, 1094, 3090, 1205, 2239, 3215, 1207, 510, 3339, 1223, 3799, 3181, 1070])
([9, 9, 9, 9, 3, 5, 9, 9, 9, 5, 3, 9, 9, 3, 9], ' ------- ', 9, ' : ', 9)
('les indice des k voisin:', [1365, 3593, 2602, 1686, 334, 2632, 2777, 996, 1364, 2313, 1702, 323, 3525, 3520, 3805])
([8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8], ' ------- ', 8, ' : ', 8)
('les indice des k voisin:', [2958, 181, 1017, 3816, 3511, 2348, 2955, 3087, 3484, 2790, 303, 2965, 1544, 3754, 2534])
([9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9], ' ------- ', 9, ' : ', 9)
('les indice des k voisin:', [1354, 2528, 2690, 2613, 2514, 3521, 2523, 775, 1408, 3570, 417, 3412, 3245, 1357, 3240])
([8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8], ' ------- ', 8, ' : ', 8)
('les indice des k voisin:', [1336, 2756, 1529, 3536, 1303, 133, 2452, 1058, 1803, 437, 2451, 477, 2800, 2593, 1681])
([4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4], ' ------- ', 4, ' : ', 4)
('les indice des k voisin:', [207, 2588, 732, 60, 917, 51, 927, 3070, 27, 966, 2137, 1014, 240, 256, 3620])
([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], ' ------- ', 1, ' : ', 1)
('les indice des k voisin:', [408, 484, 2615, 3121, 1930, 415, 2789, 201, 2780, 1054, 3401, 440, 2471, 1972, 1952])
([7, 7, 9, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 9, 7], ' ------- ', 7, ' : ', 7)
('les indice des k voisin:', [2924, 1598, 516, 3367, 1567, 914, 932, 3355, 981, 2561, 1987, 3265, 1391, 119, 3338])
([7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7], ' ------- ', 7, ' : ', 7)
('les indice des k voisin:', [3581, 1454, 2457, 2025, 3488, 2519, 3444, 3498, 1536, 1189, 1512, 1706, 1262, 1242, 2660])
([3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3], ' ------- ', 3, ' : ', 3)
('les indice des k voisin:', [1329, 1318, 3705, 2754, 1835, 3733, 1328, 1111, 2586, 1691, 2009, 705, 2703, 896, 335])
([5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 1, 5, 9], ' ------- ', 5, ' : ', 5)
('les indice des k voisin:', [989, 2387, 3371, 542, 927, 732, 207, 2308, 2137, 1690, 1339, 1012, 60, 997, 3730])
([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], ' ------- ', 1, ' : ', 1)
('les indice des k voisin:', [2054, 222, 1623, 2647, 252, 2047, 1667, 1191, 2984, 1483, 1028, 3579, 94, 2988, 824])
([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], ' ------- ', 0, ' : ', 0)
('les indice des k voisin:', [3340, 125, 1848, 2036, 1545, 936, 1836, 1583, 102, 2978, 1611, 2047, 1996, 1075, 921])
([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], ' ------- ', 0, ' : ', 0)
('les indice des k voisin:', [1417, 1289, 3038, 1251, 2220, 2569, 2583, 310, 2039, 344, 1005, 975, 3042, 3492, 1419])
([2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2], ' ------- ', 2, ' : ', 2)
('la precision est de:', 48, ' ', 50, ' ', 0.96)
('voici le rappel:', 0.012558869701726845)
('la mesure F:', 0.02479338842975207)
azaria@azaria-ThinkPad-X240:~/Documents/tp_apprentissage$
```

FIGURE *10 – 15 plus proches voisins sur 50 individus*

La figure 11, avec k=8 et 50 individus on a une précision de 96%. On constate que la précision n'a pas changé par rapport à la figure 10 tout en utilisant le nombre de paramètre diffèrent.

```
azaria@azaria-ThinkPad-X240:~/Documents/tp_apprentissage$ python try.py optique_train.csv optique_test.csv 8
('les classes:', [0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
('les indice des k voisin:', [401, 432, 1983, 2505, 2087, 1630, 2975, 3384])
([1, 1, 1, 1, 1, 1, 1, 1], ' ------- ', 1, ' : ', 1)
('les indice des k voisin:', [1417, 1530, 2876, 1747, 2220, 1637, 1199, 1991])
([2, 1, 1, 1, 2, 1, 1, 1], ' ------- ', 1, ' : ', 2)
('les indice des k voisin:', [2245, 2025, 3801, 3498, 1781, 1512, 1247, 1160])
([3, 3, 3, 3, 3, 3, 3, 3], ' ------- ', 3, ' : ', 3)
('les indice des k voisin:', [93, 66, 660, 3199, 3060, 1546, 2038, 3507])
([4, 4, 4, 4, 4, 4, 4, 4], ' ------- ', 4, ' : ', 4)
('les indice des k voisin:', [353, 3414, 1070, 3215, 1063, 3415, 2966, 2525])
([9, 9, 9, 9, 3, 9, 9, 9], ' ------- ', 9, ' : ', 5)
('les indice des k voisin:', [746, 1672, 3320, 2587, 2570, 3407, 1739, 2114])
([6, 6, 6, 6, 6, 6, 6, 6], ' ------- ', 6, ' : ', 6)
('les indice des k voisin:', [1018, 2423, 3355, 1565, 122, 981, 841, 2273])
([7, 7, 7, 7, 7, 7, 7, 7], ' ------- ', 7, ' : ', 7)
('les indice des k voisin:', [2124, 637, 1227, 3469, 188, 1213, 1232, 628])
([8, 8, 8, 8, 8, 8, 8, 8], ' ------- ', 8, ' : ', 8)
('les indice des k voisin:', [3215, 1106, 511, 3207, 2252, 2486, 1139, 706])
([9, 9, 9, 9, 9, 9, 9, 9], ' ------- ', 9, ' : ', 9)
('les indice des k voisin:', [449, 3074, 3392, 1265, 2195, 1495, 3253, 3362])
([0, 0, 0, 0, 0, 0, 0, 0], ' ------- ', 0, ' : ', 0)
('les indice des k voisin:', [51, 1525, 2123, 3361, 27, 1014, 3326, 1524])
([1, 1, 1, 1, 1, 1, 1, 1], ' ------- ', 1, ' : ', 1)
('les indice des k voisin:', [98, 1533, 1960, 3473, 2209, 2491, 2952, 3509])
([2, 2, 2, 2, 2, 2, 2, 2], ' ------- ', 2, ' : ', 2)
('les indice des k voisin:', [3516, 156, 2660, 3365, 2008, 2457, 2630, 284])
([3, 3, 3, 3, 3, 3, 3, 3], ' ------- ', 3, ' : ', 3)
('les indice des k voisin:', [2451, 437, 2756, 3472, 2656, 1529, 2175, 477])
([4, 4, 4, 4, 4, 4, 4, 4], ' ------- ', 4, ' : ', 4)
('les indice des k voisin:', [702, 2697, 1927, 3792, 2878, 671, 705, 738])
([5, 5, 5, 5, 5, 5, 5, 5], ' ------- ', 5, ' : ', 5)
('les indice des k voisin:', [749, 3512, 3238, 3007, 429, 3552, 2079, 1392])
```

FIGURE *11 – 8 plus proches voisins sur 50 individus*

```
([0, 0, 0, 0, 0, 0, 0, 0], ' ------- ', 0, ' : ', 0)
('les indice des k voisin:', [3163, 590, 2525, 1094, 3090, 1205, 2239, 3215])
([9, 9, 9, 9, 3, 5, 9, 9], ' ------- ', 9, ' : ', 9)
('les indice des k voisin:', [1365, 3593, 2602, 1686, 334, 2632, 2777, 996])
([8, 8, 8, 8, 8, 8, 8, 8], ' ------- ', 8, ' : ', 8)
('les indice des k voisin:', [2958, 181, 1017, 3816, 3511, 2348, 2955, 3087])
([9, 9, 9, 9, 9, 9, 9, 9], ' ------- ', 9, ' : ', 9)
('les indice des k voisin:', [1354, 2528, 2690, 2613, 2514, 3521, 2523, 775])
([8, 8, 8, 8, 8, 8, 8, 8], ' ------- ', 8, ' : ', 8)
('les indice des k voisin:', [1336, 2756, 1529, 3536, 1303, 133, 2452, 1058])
([4, 4, 4, 4, 4, 4, 4, 4], ' ------- ', 4, ' : ', 4)
('les indice des k voisin:', [207, 2588, 732, 60, 917, 51, 927, 3070])
([1, 1, 1, 1, 1, 1, 1, 1], ' ------- ', 1, ' : ', 1)
('les indice des k voisin:', [408, 484, 2615, 3121, 1930, 415, 2789, 201])
([7, 7, 9, 7, 7, 7, 7, 7], ' ------- ', 7, ' : ', 7)
('les indice des k voisin:', [2924, 1598, 516, 3367, 1567, 914, 932, 3355])
([7, 7, 7, 7, 7, 7, 7, 7], ' ------- ', 7, ' : ', 7)
('les indice des k voisin:', [3581, 1454, 2457, 2025, 3488, 2519, 3444, 3498])
([3, 3, 3, 3, 3, 3, 3, 3], ' ------- ', 3, ' : ', 3)
('les indice des k voisin:', [1329, 1318, 3705, 2754, 1835, 3733, 1328, 1111])
([5, 5, 5, 5, 5, 5, 5, 5], ' ------- ', 5, ' : ', 5)
('les indice des k voisin:', [989, 2387, 3371, 542, 927, 732, 207, 2308])
([1, 1, 1, 1, 1, 1, 1, 1], ' ------- ', 1, ' : ', 1)
('les indice des k voisin:', [2054, 222, 1623, 2647, 252, 2047, 1667, 1191])
([0, 0, 0, 0, 0, 0, 0, 0], ' ------- ', 0, ' : ', 0)
('les indice des k voisin:', [3340, 125, 1848, 2036, 1545, 936, 1836, 1583])
([0, 0, 0, 0, 0, 0, 0, 0], ' ------- ', 0, ' : ', 0)
('les indice des k voisin:', [1417, 1289, 3038, 1251, 2220, 2569, 2583, 310])
([2, 2, 2, 2, 2, 2, 2, 2], ' ------- ', 2, ' : ', 2)
('la precision est de:', 48, ' ', 50, ' ', 0.96)
('voici le rappel:', 0.012558869701726845)
('la mesure F:', 0.02479338842975207)
azaria@azaria-ThinkPad-X240:~/Documents/tp_apprentissage$
```

FIGURE *12 – 8 plus proches voisins sur 50 individus(2)*

Vu les différents résultat obtenus et différents commentaires nous pourrons résumer que la précision, qui est l'une de mesure de la performance varie en fonction des données d'individus de test que nous utilisons.

Donc lorsque on utilise peu de donnée de test cela influence aussi le résultat final en fonction de la mesure de performance, d'où il faudrait utiliser plus les données de test pour avoir une mesure de performance très intéressante.

2.3 Démonstration du théorème

Pour un suffisamment grand ensemble d'apprentissage de taille m, le taux d'erreur de 1NN est inférieure à deux fois le taux d'erreur minimal obtenu par la classification bayésienne.

Démonstration du théorème : Pour un suffisamment grand ensemble d'apprentissage de taille m, le taux d'erreur de 1NN est inferieure a deux fois le taux d'erreur minimal obtenu par la classification bayésienne. Soit l'erreur E* de Bayes :

$$E = \int_x p(x)[1 - maxp(i/x)]$$

Le fait essentiel est que si le nombre n d'exemples de formation est assez grand, les distributions de probabilité d'étiquette pour n'importe quel point et son voisin le plus proche seront essentiellement les mêmes. Dans ce cas, pour tout point x, le taux d'erreur attendu du classificateur 1NN est :

$$\sum_{i=1}^{m} p(i|x)[1 - p(i|x)]$$

Pour prouver le théorème, nous devons montrer que :

$$\sum_{i=1}^{m} p(i|x)[1 - p(i|x)] \leq 2[1 - \max_i p(i|x)]$$

En posant du coté droite max= r et laissons que ce maximum soit atteint avec i = j Ensuite, le côté gauche devient :

$$r(1-r) + \sum_{i \neq j} p(i|x)[1 - p(i|x)]$$

Et le côté droit nous donnera : 2 (1 - r). La somme ci-dessus est maximisée. La valeur du côté gauche devient alors :

$$\begin{aligned} A &= r(1-r) + (m-1)\frac{1-r}{m-1}\frac{m-1-(1-r)}{m-1} \\ &= r(1-r) + (1-r)\frac{m+r-2}{m-1}. \end{aligned}$$

Nous remarquons que

$$r \leq 1 et m - 2 + r < m - 1 \Longrightarrow A < 2(1-r)$$

C'est ce que nous voulons prouver.

3 Arbre de décision

3.1 Construction de l'Abre de décision

Pour le besoin de cette partie nous utilisons une base contenant des données météorologiques, elle est constituée de 5 variables et 14 instances. L'objectif de cette partie est de construire un arbre de décision pour prédire la pratique ou non du golf selon les données météorologiques.

Outlook	Temperature	Humidity	Windy	Class
sunny	85	85	false	Don't Play
sunny	80	90	true	Don't Play
overcast	83	78	false	Play
rain	70	96	false	Play
rain	68	80	false	Play
rain	65	70	true	Don't Play
overcast	64	65	true	Play
sunny	72	95	false	Don't Play
sunny	69	70	false	Play
rain	75	80	false	Play
sunny	75	70	true	Play
overcast	72	90	true	Play
overcast	81	75	false	Play
rain	71	80	true	Don't Play

TABLE *1 – Ensemble d'apprentissage*

Pour résoudre ce problème nous procédons de la manière suivante :

* Détermination de la variable racine en calculant l'entropie des différentes variables et en choisissant la plus petite information.

  Pour rappel, "Outlook" = "Sunny" = Info ([2,3]) = entropy(2/5,3/5) = - 2/ 5log(2 / 5) - 3/ 5log(3/5) = 0.971 bits

Variable Outlook			
"Outlook"="**Sunny**"			
Play	Don't Play	Effectif	Info([2,3])
2	3	5	0,97095059
"Outlook"="**overcast**"			
Play	Don't Play	Effectif	Info([4,0])
4	0	4	0
"Outlook"="**Rain**"			
Play	Don't Play	Effectif	Info([3,2])
3	2	5	0,97095059
Info(Outlook) avec 14 indivudus			
Info([2,3],[4,0],[3,2])=			**0,69353614**
Gain("Outlook)			0,24674982

Variable Windy			
"Windy"="**True**"			
Play	Don't Play	Effectif	Info([3,3])
3	3	6	1
"Windy"="**False**"			
Play	Don't Play	Effectif	Info([6,2])
6	2	8	0,81127812
Info(Wendy) avec 14 indivudus			
Info([3,3],[6,2])=			**0,89215893**
Gain("Wendy")			0,04812703

FIGURE *13 – Entropie(Outlook) et Entropie(Wendy)*

Entropy(Temperature) : Spécifiquement pour cette partie nous procédons à la segmentation des valeurs de cette variable quantitative. Le principe de segmentation se fait par chaque changement de positif au négatif et inversement.

Variable Temperature								
Temperature	Division	Play	Don't Play	Somme	Nombre	Entropy	Info. Temp.	Info. Apres
Temperature < 64,5	1	1	0	1	14	0	0	0,892576847
Temperature > 64,5		8	5	13		0,961237	0,892576847	
Temperature < 66,5	2	1	1	2	14	1	0,142857143	0,929967858
Temperature > 66,5		8	4	12		0,918296	0,787110715	
Temperature <70,5	3	4	1	5	14	0,721928	0,257831462	0,894951787
Temperature >70,5		5	4	9		0,991076	0,637120324	
Temperature < 77,5	4	7	3	10	14	0,881291	0,629493499	0,915207785
Temperature > 77,5		2	2	4		1	0,285714286	
Temperature < 80,5	5	7	4	11	14	0,94566	0,743018811	0,939796489
Temperature > 80,5		2	1	3		0,918296	0,196777679	
Temperature < 84	6	9	4	13	14	0,890492	0,826885094	0,826885094
Temperature > 84		0	1	1		0	0	
Information generale							0,940285959	0,826885094

Variable Humidity								
Humidity	Division	Play	Don't Play	Somme	Nombre	Entropy	Info. Humi.	Info. Avant
Humidity < 67,5	1	1	0	1		0	0	
Humidity > 67,5		8	5	13	14	0,961237	0,892576847	0,892576847
Humidity < 82,5	2	7	2	9		0,764205	0,491274326	
Humidity > 82,5		2	3	5	14	0,970951	0,346768069	0,838042395
Humidity < 87,5	3	7	3	10		0,881291	0,629493499	
Humidity > 87,5		2	2	4	14	1	0,285714286	0,915207785
Humidity <95,5	4	8	5	13		0,961237	0,892576847	
Humidity >95,5		1	0	1	14	0	0	0,892576847

FIGURE *14 – Entropie(Température) et Entropie(Humidity)*

En fonction de la figure 13 et figure 14, on retient la racine comme variable Outlook suite à sa petite information, **Info (outlook) = 0.693 bits**

La racine a été trouvé on passe maintenant a l'étape suivante de Partionner **"Outlook"= "Sunny"** en faisant reprendra avec l'itération pour faire le choix du prochain noeu soit Temperature, Humidity ou Hendy. Donc variables de la branche Sunny.

Outlook	Temperature	Humidiy	Windy	Class
sunny	69	70	false	Play
sunny	75	70	true	Play
sunny	85	85	false	Don't Play
sunny	80	90	true	Don't Play
sunny	72	95	false	Don't Play

H(Windy)			
"Windy"="True"			
Play	Don't Play	Effectif	Info([1,1])
1	1	2	1
"Windy"="False"			
Play	Don't Play	Effectif	Info([1,2])
1	2	3	0,91829583
Info(Windy) avec 5 indivudus			
Info([1,1],[1,2])=			0,95331191

FIGURE *15 – Entropie(Windy)*

Ci dessous le calcul d'information de la Variable Temperature et Humidity

On remarque sur cette figure 16 par ce partitionnement, l'information la plus petite est celle de la variable Humidity.

H(Temperature)								
Temperature	Division	Play	Don't Play	Somme	Nombre	Entropy	Info. Temp	Info. Apres
Temp< 70,5	1	1	0	1	4	0	0	0,550978
Temp > 70,5		1	2	3		0,9182958	0,5509775	
Temp < 73,5	2	1	1	2	4	1	0,5	1
Temp >73,5		1	1	2		1	0,5	
Temp <77,5	3	2	1	3	4	0,9182958	0,5509775	0,550978
Temp >77,5		0	1	1		0	0	
Information generale							0,940285959	0,550978

H(Humidity)								
Humidity	Division	Play	Don't Play	Somme	Nombre	Entropy	Info. Humi.	Info. Avant
Humidity < 77,5	1	2	0	2	5	0	0	0
Humidity > 77,5		0	3	3		0	0	

N.B: Le choix se portera sur la variable Humidity

Choix min(info) => Humidity

FIGURE *16 – Entropie(Température) et Entropie(Humidity)*

Partitionner "Outlook" ="Overcast"

Partitionner Outlook =Overcast

Outlook	Temperature	Humidity	Windy	Class
overcast	83	78	false	Play
overcast	64	65	true	Play
overcast	72	90	true	Play
overcast	81	75	false	Play

Choix min(info) => Play

Partitionner Outlook = Rain

Outlook	Temperature	Humidity	Windy	Class
rain	70	96	false	Play
rain	68	80	false	Play
rain	65	70	true	Don't Play
rain	75	80	false	Play
rain	71	80	true	Don't Play

FIGURE *17 – Entropie(Outlook=Overcast et Outlook=Rain))*

En fonction de cette figure 17 on peut voir clairement que à partir de la branche overcast la décision est play, qui est directement la feuille.

On continue l'iteration dans la branche de Rain pour choisir le prochain noeu soit Temperature, humidity ou wendy.

INFO Température

Temperature	Class
65	Don't Play
68	Play
70	Play
71	Don't Play
75	Play

	Div1=66	Div2=70,5	Div3=73
Partie Gauche			
Nbr Occurrence Play	0	2	2
Nbr Occurrence Don't Play	1	1	2
Partie Droite			
Nbr Occurrence Play	3	1	1
Nbr Occurrence Don't Play	1	1	0
Entropie Gauche	0	0,9182958	1
Entropie Droite	0,8112781	1	0
Informations générales	0,649022	0,9509775	0,8

FIGURE *18 – Entropie(Température)Branche Rain*

Calcul d'Entropie (Humidity)

INFO Humidity	
Humidity	**Class**
70	Don't Play
80	Play
80	Play
80	Don't Play
96	Play

		Div1=75	Div2=88
Partie Gauche			
Nbr Occurrence Play		0	2
Nbr Occurrence Don't Play		1	2
Partie Droite			
Nbr Occurrence Play		3	1
Nbr Occurrence Don't Play		1	0
Entropie Gauche		0	1
Entropie Droite		0,8112781	0
Informations générales		0,649022	0,6

FIGURE *19 – Entropie(Humidity) Branche Rain*

Calcul d'Entropie (Windy)

H(Windy)			
"Windy"="True"			
Play	Don't Play	Effectif	Info([0,2])
0	2	0	0
"Windy"="False"			
Play	Don't Play	Effectif	Info([3,0])
3	0	3	0
Info(Windy) avec 5 indivudus			
Info([0,2],[3,0])=			0
Choix min(info) => Windy			

FIGURE *20 – Entropie(Wendy) Branche Rain*

Au regard de la figure 20, l'information minimale est la variable Windy.
A partir de toutes les informations nous pourrions bien construire notre arbre de décisions.

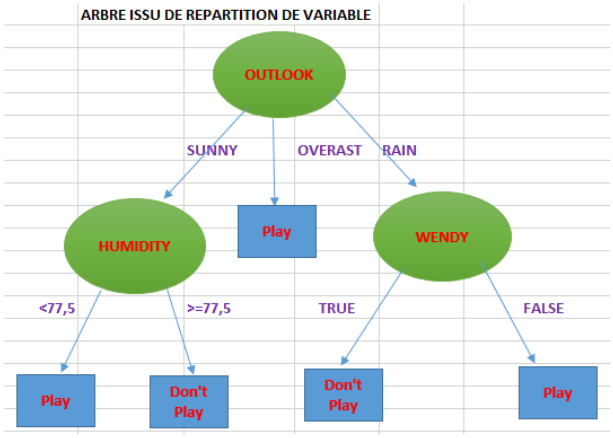

FIGURE *21 – Arbre de décision*

∗ **De cet arbre nous pouvons extraire les règles de décision suivante :**

- Si(outlook=sunny) Et (Humidity<77,5) Alors Play
- Si(outlook=sunny) Et (Humidity>=77,5) Alors Don't play
- Si(outlook=overcast) Alors Play
- Si(outlook=rain) Et (windy=true) Alors Don't play
- Si(outlook=rain) Et (windy=false) Alors play

Ainsi à partir de ces règles nous pouvons classer les individus suivants

Classification des individus

Outlook	Temperature	Humidity	Windy	Class
overcast	63	70	false	Play
rain	73	90	true	Don't Play
sunny	70	73	true	Play

FIGURE *22 – Arbre de décision*

3.2 Expliquer pourquoi l'ensemble d'arbres de décision améliore la prédiction d'un seul

En effet un ensemble d'arbre permet de faire un meilleur apprentissage qu'un seul car chacun des arbres apprend différemment les mêmes données et pour réaliser la classification on choisit la classe majoritaire prédite ce qui assure une meilleur classification que si c'est un seul qui est considéré.

Aussi l'erreur de classification réalisée par un ensemble d'arbre lors d'un apprentissage est inférieure à celle réalisé par un seul.

3.3 Démontrer que le bagging de k plus proches voisins n'améliore pas le comportement d'un seul

Il consiste à sous-échantillonner le training set et de faire générer à l'algorithme voulu un modèle pour chaque sous-échantillon. On obtient ainsi un ensemble de modèles dont il convient de moyenner (lorsqu'il s'agit d'une régression) ou de faire voter (pour une classification) les différentes prédictions.

L'utilisation du Bagging est adaptée aux algorithmes à fortes variance qui sont ainsi stabilisés (réseaux neuronaux, arbres de décision pour la classification ou la régression...), mais il peut également dégrader les qualités pour des algorithmes plus stables (k plus proches voisins avec k grand, régression linéaire). On générera par exemple des dizaines de classifieurs au total qu'il va falloir « bagger ». Aussi le résultat obtenu avec le KNN n'est pas trop dégrader, donc nous pouvons conclure qu'un bagging du KNN détériorera la stabilité du KNN.

4 Conclusion et Perspective

- Ce Travail à permis de nous structurer en matière de machine learning selon les problèmes posés.
- Il n'existe pas de méthode d'apprentissage parfaite, les paramètres d'entrée peuvent influencer la qualité des modèles obtenus.

Références

[1] Les méthodes ensembliste, http ://blog.octo.com/les-methodes-ensemblistes-pour algorithmes-de-machine-learning/ ;

[2] Theorem : For sufficiently large training set size n, the error rate of the 1NN classifier is less than twice the Bayes error rate. (Cover Hart, 1967)