RAPPORT APPRENTISSAGE AUTOMATIQUE : TP1 et TP2

Rédigé par :

- **MEDOU Daniel Magloire, P21**
- **ELIODOR Ednalson Guy Mirlin, P21**

Etudiant en Master 2-IFI, option SIM

Sous la supervision de :

Dr. THANH-NGHI Do

Année académique 2016-2017

TABLE DES MATIERES

TABLE DE FIGURES

TABLE DES TABLEAUX

INTRODUCTION

L'apprentissage automatique (Machine Learning en anglais) ou encore Fouille de Données (Data Mining) comprend différentes méthodes de recherche d'informations dans un jeu de données à des fins prévisionnelles et/ou et décisionnelles. L'apprentissage supervisé étant une technique qui nous permettra à produire automatiquement des règles à partir d'une base de données d'apprentissage. Pour ce faire, nous avons plusieurs méthodes d'apprentissage supervisé mise à notre disposition à savoir **Boosting, Machine à Secteurs de Support, Réseau de neurones, Méthode de K plus proches voisins, Arbre de décision, Classification naïve bayésienne** et bien d'autres existant. Toutes ces méthodes ne sont pas appliquées à un jeu de données d'apprentissage comme bon nous semble car plusieurs critères entrent en jeu pour appliquer une quelconque à un jeu de données et selon l'objectif ou le but de ce dernier.

TP1 : K plus proches voisins (knn) et Arbres de Décision

I. K plus proches voisins

1. Application manuelle de KNN

L'application manuelle du KNN sera effectuée par le jeu de données des individus ci-dessous.

Tableau 1: Base d'individus X1 et X2

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

Dans cette partie manuelle, il est question de classifier les nouvels individus X1 et X2 donnés en deux classes différentes 1NN et 3NN.

Le principe qui sera appliqué pour la résolution de cette classification est celui de la **distance de Manhattan**.

$d(U,V) = |U_1-V_1| + |U_2-V_2| + \ldots + |U_n-V_n|$

Dans notre cas d'espèce, $\mathbf{X_1}$ et $\mathbf{X_2}$ $\mathbf{\in}$ **[0,1]** alors nous n'aurons plus à faire une quelconque normalisation de $\mathbf{X_2}$. Ceci étant, après le calcul des **distances de Manhattan (d1, d2, d3, d4)**, nous choisissons la plus petite distance calculée et celle-ci sera la nouvelle classe du nouvel individu.

NB. Un fichier Excel des calculs effectués sera joins à ce rapport pour une éventuelle vérification.

Apres ces calculs, nous obtenons les résultats suivants :

- Classe 1NN

Tableau 2: Classification de nouvels individus, cas 1NN

X1	X2	Classe 1NN
0.550000	0.364000	1
0.558000	0.470000	0
0.456000	0.450000	1
0.450000	0.570000	0

- Classe 3NN

Ici, nous choisissons trois (3) de petites valeurs de chaque distance de Manhattan (d1, d2, d3, d4) calculée, nous retrouvons leurs classes et la celle majoritaire est attribuée à notre nouvel individuel.

Tableau 3: Classification de nouvels individus, cas 3NN

X1	X2	Classe 1NN
0.550000	0.364000	1
0.558000	0.470000	1
0.456000	0.450000	0
0.450000	0.570000	0

2. Implémentation de KNN en C+

Le programme implémenté permet entrée une base d'apprentissage, une base de test et un nombre k et en sortie nous avons une matrice de confusion et un taux global de bon classement associé à cette base.

Taper la commande suivante pour lancer et exécuter le programme:
./Tp1Binome_ELIODOR_Medou_Apprentissage fichier_base_apprentissage fichier_base_test k.

En effet, lorsqu'une commande est exécutée, le programme lit l'ensemble des données d'apprentissage et les stocke dans des tableaux. Puis la similarité de chaque individus de l'ensemble de test est évaluée par le calcul d'une distance entre lui 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.

Par exemple la commande: ***./Tp1Binome_ELIODOR_Medou_Apprentissage data/iris/iris.trn data/iris/iris.tst 2*** exécute l'algorithme d'apprentissage du 2NN sur la base d'apprentissage fp/fp.trn, puis fait le test sur l'ensemble de données fp/fp.tst. En sortie, nous obtenons la matrice de confusion ainsi que le taux de bon classement tel que présenté ci-dessous (cf. fig 1.1).

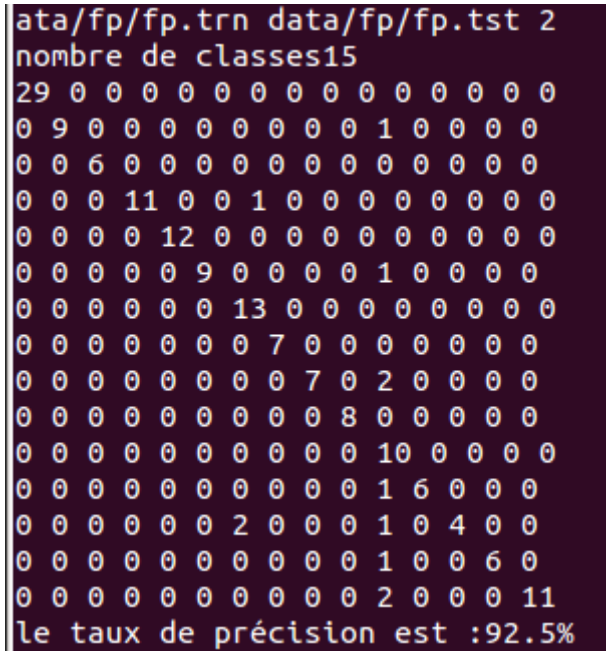


Figure 1 KNN sur l'ensemble de données fp

Dans le préent rapport, nous allons donner quelques tests et leurs résultats. La synthèse de tous les autres sera faite apres ces derniers.

$ ./Tp1Binome_ELIODOR_Medou_Apprentissage data/iris/iris.trn data/iris/iris.tst 5

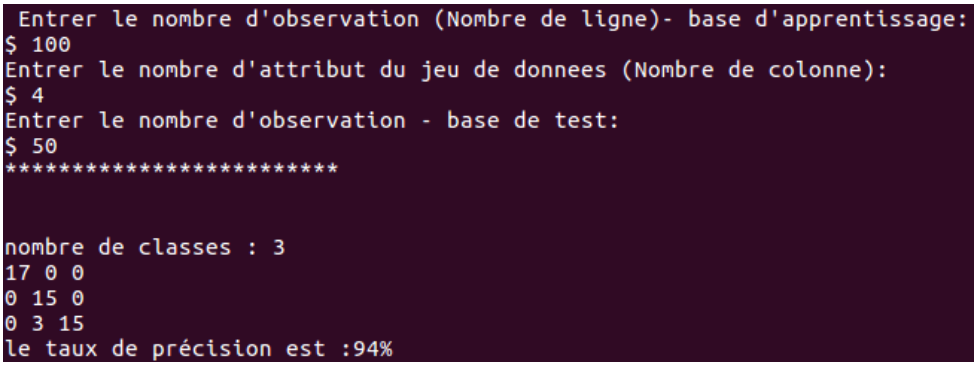


Figure 2 KNN sur l'ensemble de données iris (k=5)

$ ./Tp1Binome_ELIODOR_Medou_Apprentissage data/optics/opt.trn data/optics/opt.tst 4

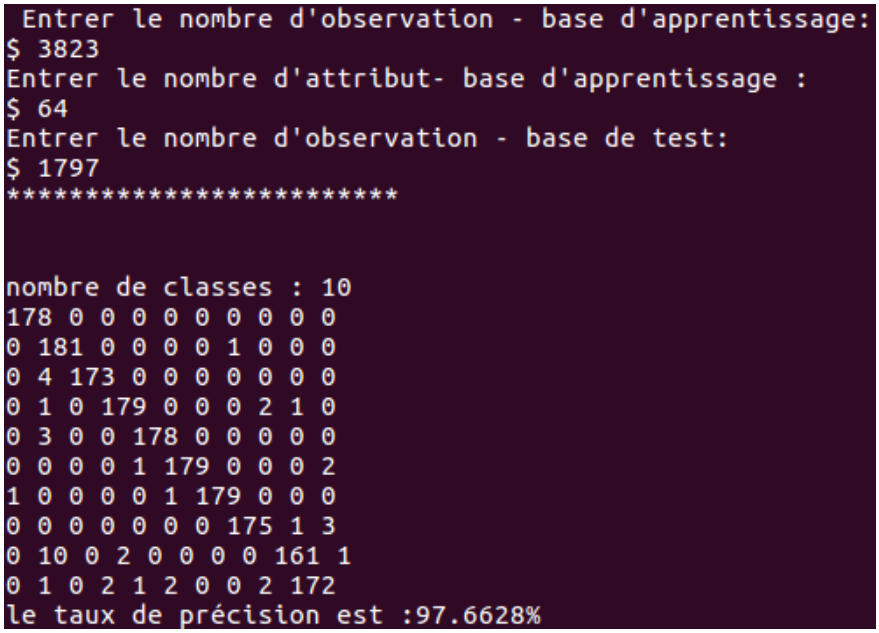


Figure3 KNN sur l'ensemble de données optics (k=4)

$ ./Tp1Binome_ELIODOR_Medou_Apprentissage data/letter/let.trn data/letter/let.tst 5

Les bases utilisées dans le cadre de l'expérimentation sont *iris, letter, optics et fp*. L'évaluation de notre programme se base sur le taux global de bon classement obtenu pour chacune de ces bases. Un premier jeu de test nous permet de constater que le taux global de bon classement est meilleur pour la distance Euclidienne contrairement à la distance de Manhattan par exemple. Aussi la valeur de k influence la qualité du modèle obtenu. Notamment avec k=5, nous obtenons généralement les meilleurs taux pour les bases classées et cela quelque soit la distance choisie. Ci-dessous, un tableau récapitulant les taux de bon classement obtenus en fonction de la base, de la valeur de k. Dans ces tableaux, nous présenterons uniquement les résultats pour les bases iris, letter et optics. Nous procéderons différemment pour la base fp.

Dataset	K	Distance	Taux de bon placement global
Letter	5	Manhattan	94.0594%
iris	5	Manhattan	92,00%
Optics	5	Manhattan	96.7724%

Dataset	K	Distance	Taux de bon placement global
Letter	5	Euclidienne	94.2394%
iris	5	Euclidienne	94,00%
Optics	5	Euclidienne	97.8854%

La base fp est stockée dans le fichier fp.data et nécessite un hold-out c'est-à-dire une séparation aléatoire en ensemble d'apprentissage (2/3 du total) et ensemble de test(1/3 du total). Pour le faire notre programme inclus le rangement aléatoire de tous les individus(total=480) de fp, puis la prise des premiers pour l'apprentissage (320 lignes stockées dans le fichier *fp.trn*) et du reste pour le test (160 lignes stockées dans le fichier fp.tst). Les lignes effectuant le hold-out sont:

```
system("sort -R data/fp/fp.data > data/fp/fp.txt");
system("head -n320 data/fp/fp.txt > data/fp/fp.trn");
system("tail -n160 data/fp/fp.txt > data/fp/fp.tst");
```

Figure5 commandes pour le hold-out de la base fp

Pour une meilleure évaluation, nous proposons de fixer le nombre d'exécutions à 5 avec la même valeur de k, puis nous faisons une moyenne des taux de bon classement global obtenus pour conclure à un taux pour cette valeur de k.

K=2 itération	Taux global de bon classement	K=3 itération	Taux global de bon classement	K=5 itération	Taux global de bon classement
1	88,125	1	90,625	1	85,625
2	89,375	2	86,875	2	89,375
3	88,75	3	90,625	3	86,875
4	87,5	4	84,375	4	92,5
5	93,125	5	88,125	5	89,375
Moyenne=	**89,375**	**Moyenne=**	**88,125**	**Moyenne=**	**88,75**

Tableau 1.4: Expérimentation de fp en faisant la moyenne des taux de bon classement global

L'observation du résultat des nombreuses exécution de notre programme nous montre que le meilleur taux global de bon classement est obtenu avec la base **optics** pour **k=1**. Ce taux est de **97.9967%**. Nous constatons par ailleurs qu'avec la distance Euclidienne le

résultat est meilleur qu'avec la distance Manhattan si k très petit. Aussi, avec **k=5** le taux global de bon classement est suffisamment grand quelque soit la distance utilisée.

Le principal avantage du KNN est que les instances à classifier sont évaluées une à une. Malheureusement cela accroît considérablement le coût de classification. De plus, il n'y pas d'hypothèses et tout l'ensemble d'apprentissage doit être lu pour chaque nouvel individu à classer.

3. Démonstration du théorème [1]

Pour démontrer que dans un plus grand ensemble d'apprentissage de taille m, le taux d'erreur de 1NN est inférieur à deux fois le taux d'erreur minimal obtenu par la classification bayésienne :

R =< 2R*(1-R*) avec :

R : Taux d'erreur 1NN ;

R* : Taux d'erreur de Bayes ou risque de Bayes.

Nous posons la formule suivante :

R = E [2η(X) (1-η(X)], avec X espace métrique séparable et où **η(X) = P {Y = 1 | milieu X}**. C'est la probabilité conditionnelle d'observer un échantillon de la classe 1 donnée, l'observation est X et la précision est prise par rapport à la mesure de X.

Ainsi ;

R = 2E [min {η(X), (1-η(X)} max {η(X), (1-η(X }],

On peut donc considérer cette égalité comme le produit de deux nombres **η(X)** et **(1-η(X)** pouvant se traduire comme étant le produit du maximum fois le plus petit.

Avec **min {η(X), (1-η(X) :** probabilité conditionnelle d'erreur donnée X à une règle qui est optimale à X car celle-ci choisirait donc la classe avec la plus grande probabilité d'occurrence.

Appelons cette perte **r* (X)** car, c'est cette perte qui mène au risque de Bayes.

R* = E [r* (X)], par définition au risque de Bayes. Par conséquent, nous avons

$$R = 2E[r^*(X)(1 - r^*(X))]$$

$$= 2E[r*(X) - r*^2(X)]$$

$$= 2E[r*(X) - r*^2(X) + R*^2 - R*^2 + 2R*r*(X) - 2R*r*(X)]$$

$$= 2E[r*(X) + R*^2 - 2R*r*(x) - (r*(X) - R*)^2] \quad (1)$$

$$= 2E[r*(X) + R*^2 - 2R*r*(x)] - var(r*(X)) \quad (2)$$

1 *Nearest Neighbor Classifiers, pdf, Page 5*

$\leq 2E\ r*(X) + R*^2 - 2R*r*(x)$ **(3)**

$= 2(E\ [r*(X)] + E\ [R*^2] - 2E\ [R*r*(x)])$ **(4)**

$= 2(R* + R*^2 - 2R*^{2)}$ **(5)**

$= 2R*\ (1 - R*)$

(1) Réorganisation des termes
(2) R * étant la valeur attendue de r * (X).
(3) L'inégalité découle du fait que la variance est non négative
(4) L'équation découle de la linéarité des prévisions
(5) **R * est la valeur attendue de r * (X)**

II. ARBRE DE DÉCISION

1. Application manuelle pour la construction de l'arbre de décision

Notre ensemble d'apprentissage est constitué est constitué d'un ensemble de conditions ou données météorologiques qui nous permettent de prédire la pratique du golf (Play ou Don't Play). Ce dernier possède cinq (5) attributs parmi lesquels :

- quatre (4) attributs explicatifs ;
- un (1) à prédire (Play, Don't Play) ;
- Quatorze (14) observations.

Tableau 4: Jeu de donnée d'apprentissage pour la construction de l'arbre de décision

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

A partir de notre ensemble de données, nous procédons à la construction de l'arbre de décision par la fonction d'Entropie de Shannon comme suite :

- Le choix de l'attribut qui sera la racine de notre arbre

Pour décider sur l'attribut racine, nous allons tout d'abord calculer à partir de la fonction d'Entropie de Shannon, toutes les info(attribut) des quatre (04) attributs explicatifs. A partir des résultats obtenus, nous retiendrons la plus petite valeur des info(attribut) des attributs calculées et l'attribut correspondant à cette valeur sera retenu comme la racine de notre arbre.

- **Outlook**
  - Outlook = sanny [2 , 3]

Info(Outlook = sanny) = Info([2/5 , 3/5]) = -2/5log(2/5) - 3/5log(3/5) = 0.971

  - Outlook = overcast [4,0]

Info(Outlook = overcast) = Info([4/4 , 0]) = -4/4log(4/4) - 0log(0) = 0

  - Outlook = rain [3 , 2]

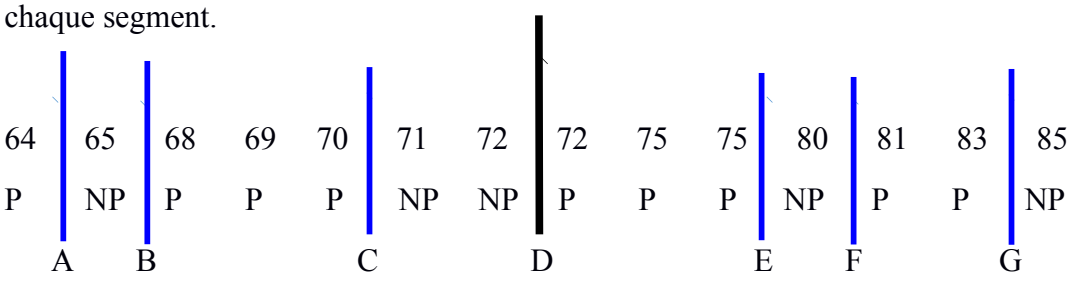
Info(Outlook = rain) = Info([3/5 , 2/5]) = -2/5log(2/5) – 2/5log(2/5) = 0.971

Info(Outlook) = 5/14 Info(Outlook = sanny) + 4/14 Info(Outlook = overcast) + 5/14 Info(Outlook = sanny)

<u>Info(Outlook) = 0.693</u>

- **Temperature**

Pour cet attribut qui est constitué de quatorze (14) valeurs numérique, nous allons procéder à la segmentation selon le changement de la valeur de l'attribut à prédire. Nous aurons sept (07) segments parmi lesquels nous choisirons la valeur minimale des différentes Info calculées de chaque segment.



Avec P = Play et NP = Don't Play

  - Pour le segment A, nous avons :

Temp < 64.5 [1 , 0] et Temp >= 64.5 [8 , 5]

Info(A) = 0.892

  - Pour le segment B, nous avons :

Temp < 66.5 [1 , 1] et Temp >= 66.5 [8 , 4]

Info(B) = 0.930

  - Pour le segment C, nous avons :

Temp < 70.5 [4 , 1] et Temp >= 70.5 [5 , 4]

Info(C) = 0.894

- **Pour le segment D, nous avons :**

Temp =< 72 [4 , 3] et Temp >= 72 [5 , 2]

Info(D) = 0.824

- Pour le segment E, nous avons :

Temp < 76.5 [7 , 3] et Temp >= 76.5 [2 , 2]

Info(E) = 0.915

- Pour le segment F, nous avons :

Temp < 80.5 [7 , 4] et Temp >= 80.5 [2 , 1]

Info(F) = 0.826

- Pour le segment G, nous avons :

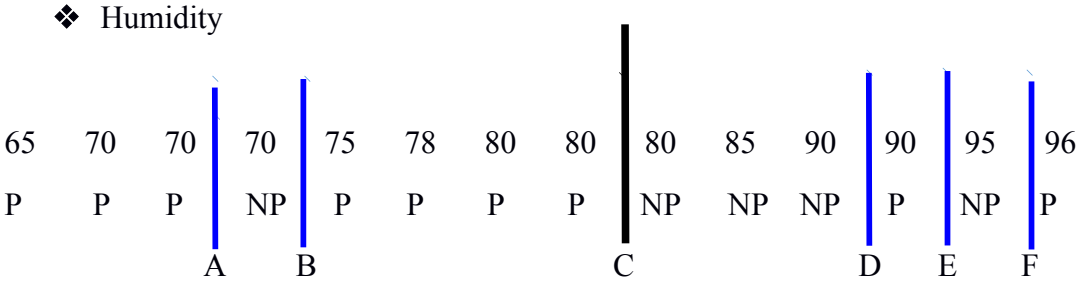
Temp < 84 [9 , 4] et Temp >= 84 [0 , 1]

Info(G) = 0.827

Apres le calcul des info de tous les segments, nous retenons la segment « D » car sa valeur Info est minimale aux autres. Donc :

<u>Info(Temperature) = 0.824</u>

- Humidity



- Pour le segment A, nous avons :

Hum < 70 [3 , 0] et Hum >= 70 [6 , 5]

Info(A) = 0.781

- Pour le segment B, nous avons

Hum < 72.5 [3 , 1] et Hum >= 72.5 [6 , 4]

Info(B) = 0.925

- **Pour le segment C, nous avons**

Hum =< 80 [7 , 1] et Hum >= 80 [2 , 4]

Info(C) = 0.704

- Pour le segment D, nous avons

Hum =< 90 [7 , 4] et Hum >= 90 [2 , 1]

Info(D) = 0.939

- Pour le segment E, nous avons

Hum < 92.5 [8 , 4] et Hum >= 92.5 [1 , 1]

Info(E) = 0.929

- Pour le segment F, nous avons

Hum < 95.5 [8 , 5] et Hum >= 95.5 [1 , 0]

Info(F) = 0.892

Apres le calcul des info de tous les segments, nous retenons le segment « C » car sa valeur Info est minimale aux autres. Donc :

Info(Humidity) = 0.704

- Windy
  - Windy = True [3 , 3]

Info(Windy = True) = Info([3/6 , 3/6]) = -3/6log(3/6) - 3/6log(3/6) = 1

- Windy = False [6 , 2]

Info(Windy = False) = Info([6/8 , 2/8]) = -6/8log(6/8) - 2/8log(2/8) = 0.631

Info(Windy) = 6/14 Info(Windy = True) + 8/14 Info(Windy = False)

Info(Windy) = 0.789

Les résultats des calcsuls manuels effectué sont comme suit :

- **Info(Outlook) = 0.693**
- Info(Temperature) = 0.824
- Info(Humidity) = 0.704

- ✓ Info(Windy) = 0.789

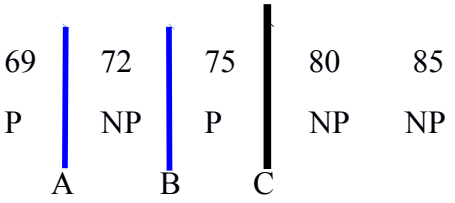
Nous observons tous que l'attribut « **Outlook** » a, la valeur Info minimale. Alors, il est la racine de l'arbre.

Etant donné que cet attribut a trois valeurs possible, nous allons procéder au partitionnement de chaque sous-ensemble « Outlook = sunny » et « Outlook = rain » afin de retenir l'attribut le plus discriminant parmi les trois (03) restant (Temperature, Humidity et Windy)

- ➢ **Partitionnement de « Outlook = sunny »**

Ici, la segmentation des valeurs des attributs restant s'effectuera uniquement sur les valeurs correspondantes à « sunny » de l'attribut « Outlook ».

- ✘ **Cas de l'attribut « Temperature »**



- ✘ Pour le segment A, nous avons :

Temp < 70.5 [1 , 0] et Temp >= 70.5 [1 , 3]

Info(A) = 0.649

- ✘ Pour le segment B, nous avons :

Temp < 73.5 [1 , 1] et Temp >= 73.5 [1 , 2]

Info(B) = 0.950

- ✘ **Pour le segment C, nous avons :**

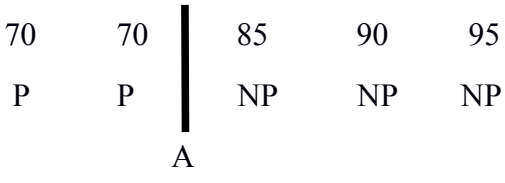
Temp < 77.5 [2 , 1] et Temp >= 77.5 [0 , 2]

Info(C) = 0.550

Après le calcul des info de tous les segments selon la branche « sunny », nous retenons le segment « C » car sa valeur Info est minimale aux autres. Donc :

<u>Info(Temperature) = 0.550</u>

- ✘ **Cas de l'attribut « Humidity »**



Hum < 77.5 [2 , 0] et Hum >= 77.5 [0 , 3]

Info(Humidity) = 0

- **Cas de l'attribut « Windy »**

Windy = True [1 , 1] ; Info(Windy = True) = 1

Windy = False [1 , 2] ; Info(Windy = False) = 0.951

Info(Windy) = 0.951

Nous avons les résultats suivants sur la branche « sunny »:

- ✓ Info(Temperature) = 0.550
- ✓ **Info(Humidity) = 0**
- ✓ Info(Temperature) = 0.951

Au vu des résultats, l'attribut « **Humidity** » est retenu pour la branche « sunny » car possède la valeur Info minimale.

- **Partitionnement de « Outlook = rain »**

Ici, la segmentation des valeurs des attributs restant s'effectuera uniquement sur les valeurs correspondantes à « rain » de l'attribut « Outlook ». Nous avons procédé comme dans le cas de la branche « sunny » et nous obtenons les résultats suivants :

- ✓ Info(Temperature) = 0.231
- ✓ Info(Humidity) = 0.231
- ✓ **Info(Windy) = 0**

L'attribut « **Windy** » est retenu pour la branche « rain » car possède la valeur Info minimale par rapport aux autres.

N'ayant plus de développement possible, notre arbre aura cinq (05) feuilles et deux (02) nœuds à soir « Humidity » et « Windy ».

2. Représentation de l'arbre

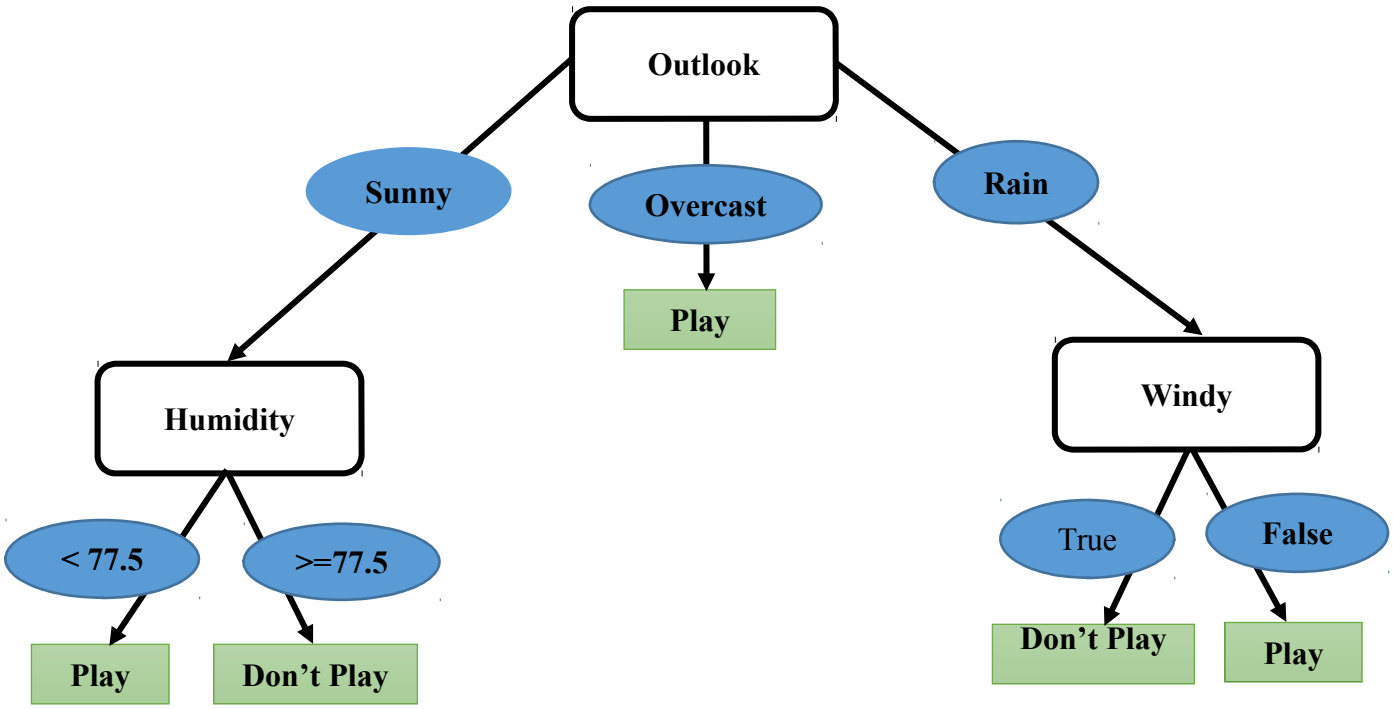


Figure 1: Arbre de décision

3. Extraction des règles d'induction (Si…Alors) à partir de l'arbre

- ✓ **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;

4. Classification des nouvels individus

A partir des règles d'induction établies ci-dessus, nous pouvons classifier ces nouvels individus comme suite.

Tableau 5: Classification des individus à partir de l'arbre de décision

Outlook	Temperature	Humidity	Windy	**Class**
Overcast	63	70	False	**Play**
Rain	73	90	True	**Don't Play**
Sunny	70	73	True	**Play**

5. Expliquer pourquoi l'ensemble d'arbres de décision améliore la prédiction d'un seul ?

En construisant une forêt d'arbres de décision à partir d'un ensemble de données nous assurons un meilleur apprentissage. Car, chacun des arbres apprend différemment le même ensemble et lors de la classification on choisit la classe majoritairement prédite par les arbres. Ce qui assure d'une meilleure classification que si on ne considérait qu'un seul arbre. Le principe est que l'erreur de classification réalisée par un groupe d'arbre est inférieure à celle réalisée par un seul arbre. En effet en termes statistiques si on a les arbres décorrélés cela permet de réduire la variance des prévisions.

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

Bagging siginifie Bootstrap aggregating. Son but est de faire un compromis entre le biais et la variance pour réduire l'erreur (Biais 2 +variance). Il est adapté pour les algorithmes d'apprentissage instables: arbre de décision, réseau de neurones, ... Mais, ne convient pas aux algorithmes d'apprentissage stables: KNN, ...

En effet la qualité du modèle obtenu avec le KNN n'est pas trop altérée par un petit changement dans l'ensemble d'apprentissage.

De ces analyses, nous concluons que procéder par un bagging du KNN va plutôt détériorer la stabilité du KNN

TP1 : Réseaux de neurones et SVM

1. Apprendre le modèle perceptron à partir de l'ensemble au-dessus pour classifier des individus en ±1.

$$y = \begin{cases} +1, & \sum_{i=0}^{2} x_i w_i > 0 \\ -1, & \sum_{i=0}^{2} x_i w_i \leq 0 \end{cases}$$

Poids initiaux : w0 = -0.5 ; w1 = 0.4 ; w2 = 0.5

Pas d'apprentissage $\eta = 0.2$

x_1	x_2	classe
0	0	-1
0	1	+1
1	0	+1
1	1	+1

Tableau 1: Ensemble à apprendre par le perceptron simple

récapitulatif des étapes d'apprentissage de l'ensemble de données avec le perceptron simple.

Itération	Couche en entree			Sortie attendue	Poids de depart			Sortie retrouvee	Erreur	Poids apres mise a jour		
	X0	X1	X2	Y	W0	W1	W2	O	e	W'0	W'1	W'2
1ere	1	0	0	-1	-0,5	0,4	0,5	-1	0	-0,5	0,4	0,5
	1	0	1	1	-0,5	0,4	0,5	-1	2	-0,1	0,4	0,9
	1	1	0	1	-0,1	0,4	0,9	1	0	-0,1	0,4	0,9
	1	1	1	1	-0,1	0,4	0,9	1	0	-0,1	0,4	0,9
2eme	1	0	0	-1	-0,1	0,4	0,9	-1	0	-0,1	0,4	0,9
	1	0	1	1	-0,1	0,4	0,9	1	0	-0,1	0,4	0,9
	1	1	0	1	-0,1	0,4	0,9	1	0	-0,1	0,4	0,9
	1	1	1	1	-0,1	0,4	0,9	1	0	-0,1	0,4	0,9

Fin des Iterations car les sorties retrouvees dans la 2eme interation correspondent donc au classe dans la base apprentissage

Tableau 2: Étapes d'apprentissage avec le perceptron simple

2. Visualisons dans l'espace 2D les individus et le droit obtenu par l'algorithme perceptron

Les **2 itérations** qui sont séparées par la droite **(D): -0,1 + 0,4x1 +0,9x2=0** constitues l'ensemble de données appris par le perception et contient les points: **A(0, 0.11) et B(0.25, 0).**

Le schéma ci-dessous permet de visualiser la classification de l'ensemble de données par le perceptron.

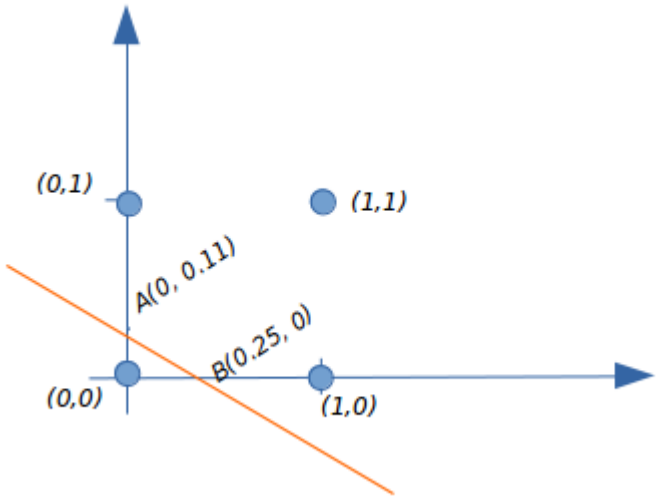


Figure1 Visualisation 2D des individus et de la droite obtenue par l'algorithme du perceptron

Nous constatons que a droite obtenue fait une bonne séparation de l'ensemble de données en deux groupes bien distincts.

3. **Apprentissage du SVM à partir du même ensemble**

En observant la figure 2.4 on remarque que les données sont linéairement séparables et qu'il y a trois supports de vecteurs à savoir : s1(0,0) s2(0,1) et s3(1,0). Ainsi, les supports de vecteurs se repatissent comme suit :

1- Le support positif qui passent par les points A et B de coordonnees successive (0, 1) et (1, 0).

2- Le support negatif qui passent par le point C de coordonnees (0, 0) et est donc dans ce cas parrallele a la droite de support positif.

Et, a partir de ses supports de vecteurs on retrouvera le systeme suivant :

Positif 1 : W1(0) + W2(1) + W3 = 1 $\quad\quad$ W2 + W3 = 1 (1)

Positif 2 : W1(1) + W2(0) + W3 = 1 $\longrightarrow$ W1 + W3 = 1 (2)

Négatif : W1(0) + W2(0) + W3 = - 1 $\quad\quad$ W3 = - 1 (3)

Résolution

(2) et (3) donne : W1 + W3 = 1 $\quad$ * (-1) $\longrightarrow$ W1 + W3 = 1

W3 = - 1 $\quad\quad$ - W3 = 1

W1 = 2

Portons W1 dans (2) ⟶ 2 + W3 = 1

⟶ W3 = - 1

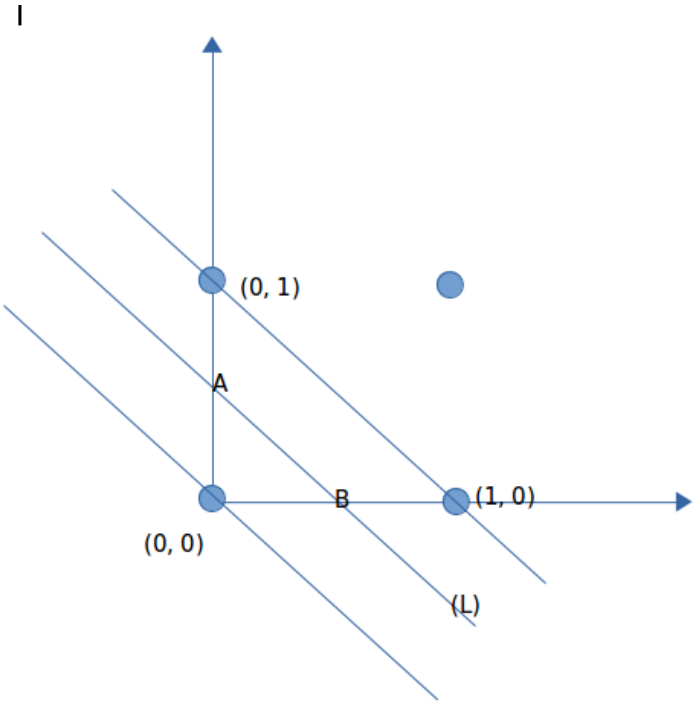
Enfin portons W3 dans (1) ⟶ W2 – 1 = 1

⟶ W2 = 2

Ayant donc la forme L : **W1 X + W2 Y + W3 = 0.**

Donc, la droite idéale est (L) et l'équation de celle-ci est la suivante : (L) : 2X1 + 2X2 – 1 = 0

Avec les points A(0, 1/2) et B(1/2, 0).



NB : **(L)** est la droite de séparation optimale

Figure 2: Droite idéale trouvée par le SVM