|  |  |  |
| --- | --- | --- |
| Numérique et Sciences Informatiques | | |
| 3h30 | **Algorithmique des k plus proches voisins** |  |
| **Objectif** : savoir écrire un algorithme qui prédit la classe d’un élément en fonction de la classe majoritaire de ses k plus proches voisins. | | |
| **Matériel**: Python | | |

**Contexte**

Vidéo de 8min23 sur l'IA :https://www.youtube.com/watch?v=yQLmgw3rClM

Vous pouvez regarder toute la chaine ...

L’intelligence artificielle est l’ensemble des théories et des techniques mises en œuvre afin de réaliser des machines capables de **simuler l’intelligence humaine**.

Les IA sont souvent amenées à classer des données.

Par exemple, dans le CAPTCHA ci-contre, en sélectionnant les carrés qui contiennent une partie du panneau STOP vous ferez un classement vers la catégorie « panneau de STOP », ce qui permettra à une IA de voiture autonome « d’apprendre » à identifier ce type de panneaux.

C’est ce qu’on appelle de l’apprentissage supervisé.

Pour le classement un des algorithmes les plus utilisés est la méthode des **k plus proches voisins (kppv)**,aussi appelée **knn** (k nearest neighbors).

Cette méthode consiste à classer un élément par rapport à ses voisins les plus proches ayant les mêmes caractéristiques afin de lui attribuer une classe (étiquette).

**Étiquette (classe)**

Lors d’un apprentissage supervisé on souhaite classer des éléments d’un **jeu de données (dataset)** à partir de leurs **caractéristiques** (**features**), pour cela il va falloir déterminer des **classes** pour ces éléments (ex : les pommes, les oranges, les kiwis …).

Supposons que sur un plan en deux dimensions nous disposions d’un ensemble de points que nous appellerons **échantillons** (**samples**) et qu’un expert a colorié certains de ces points en rouge et d’autres en bleu en fonction de leurs caractéristiques et à partir d’une logique que vous ne connaissez pas.

Bleu et rouge vont être deux **étiquettes** (**labels**) que nous pouvons attribuer à ces échantillons pour les classer. Un échantillon ne peut avoir qu’une seule étiquette.

Si on ajoute un nouveau point dans l’espace, nous souhaitons déterminer quelle étiquette lui attribuer.

**Les k plus proches voisins**

Le "k"correspond au nombre de voisins que nous allons prendre en compte.

Le but de l’algorithme des **k plus proches voisins** est de regarder les k plus proches voisins d’un **nouveau vecteur de caractéristiques** (point à classer) parmi les **échantillons connus** (points déjà classées) et de lui attribuer l'**étiquette** qui est majoritaire parmi ses voisins.

* Un expert classé des points avec des étiquettes bleus (patient sain) et des étiquettes rouges (patient malade).

En appliquant les k plus proches voisins, déterminez l’étiquette des points **p** et **m** pour :

|  |  |  |  |
| --- | --- | --- | --- |
|  | **k = 1 :** | p=rouge | m=bleu |
| **k = 3 :** | p=bleu | m=rouge |
| **k = 5 :** | p=bleu | m=rouge |
| **k = 9 :** | p=rouge | m=rouge |
| **k = 2 :** | p= ? | m= ? |

On constate que pour appliquer les **k plus proches voisins** il faut **k impair**.

**Les mal classés**

* Un expert a identifié les étiquettes suivantes pour **p** (patient sain) et **m** (patient malade).

À partir des étiquettes attribuées aux questions précédentes, indiquez si **p** et **m** sont **corrects**, **faux-positif** ou **faux-négatif** :

|  |  |  |  |
| --- | --- | --- | --- |
|  | **k = 1 :** | p=faux-positif | m=faux-négatif |
| **k = 3 :** | p=correct | m=correct |
| **k = 5 :** | p=correct | m=correct |
| **k = 9 :** | p=faux-positif | m=correct |

Sur le cas précédent on a pu voir qu’en fonction du k choisis, l’étiquette attribuée n’est pas toujours la même, pourtant chacun de ces points ne peut avoir qu’une seule étiquette !

Un expert sera en capacité de dire quelle étiquette correspond à tel point et on pourra comparer notre résultat avec celui de l’expert pour vérifier si notre algorithme fonctionne correctement.

Un exemple concret de classement est le **test PCR** (Covid notamment) car il permet de savoir si un patient est atteint d’une maladie virale, il permet donc de donner l'étiquette "malade" ou "pas malade" à un patient.

Si un test PCR est positif alors qu’un patient n’est pas malade → le résultat est un **faux-positif**.

Si un test est négatif alors que le patient est vraiment malade → le résultat est un **faux-négatif**.

**Distance**

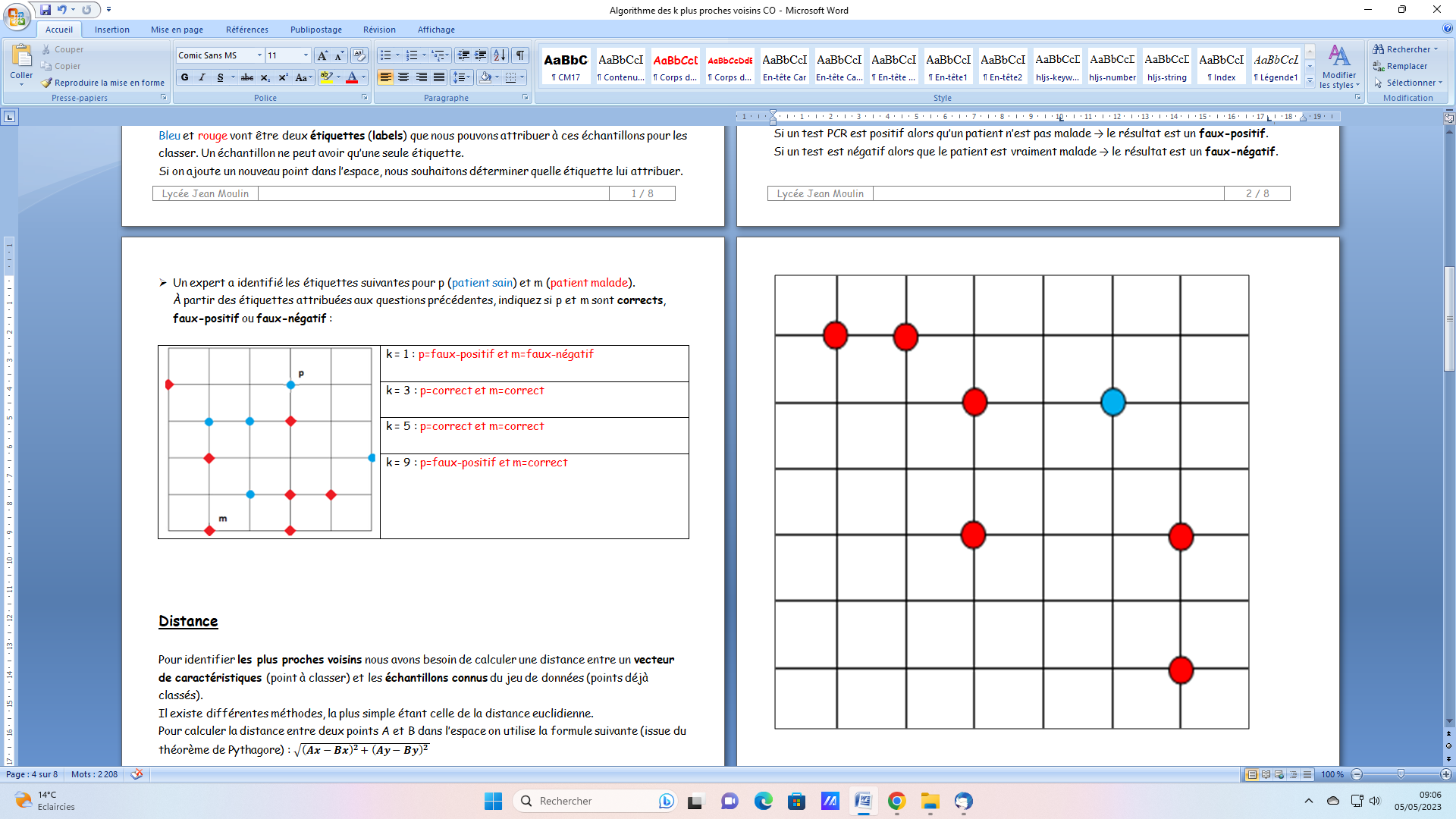
Pour identifier **les plus proches voisins** nous avons besoin de calculer une distance entre un **vecteur de caractéristiques** (point à classer) et les **échantillons connus** du jeu de données (points déjà classés).

Il existe différentes méthodes, la plus simple étant celle de la distance euclidienne.

Pour calculer la distance entre deux points A et B dans l’espace on utilise la formule suivante (issue du théorème de Pythagore) :

* Calculez la distance entre le point **A=(5,5)** et le point **B=(1,6)**.

**=**4,12

**Code 1\*** : Écrivez la fonction distance() qui permet de calculer les distances entre le point A=(5,5) et la liste des points suivants : **P1=(1,6)**, **P2=(3,5)**, **P3=(6,3)**, **P4=(3,3)**, **P5=(6,1)**, **P6=(2,6)**.

Aide :

62=6\*\*2

=sqrt(6) (from math import sqrt)

Déduisez-en quel est le point le plus proches du point A ?

C'est le point P2

**Encore plus de données !**

On peut calculer une distance avec autant de caractéristiques qu'on veut.

Prenons par exemple les **caractéristiques : âge, taille, poids**, pour distinguer deux personnes A et B.

Pour calculer la distance entre ces deux personnes nous utilisons la formule suivante :

Pour A : **Aa**=âge, **At**=taille, **Ap**=poids.

Pour B : **Ba**=âge, **Bt**=taille, **Bp**=poids.

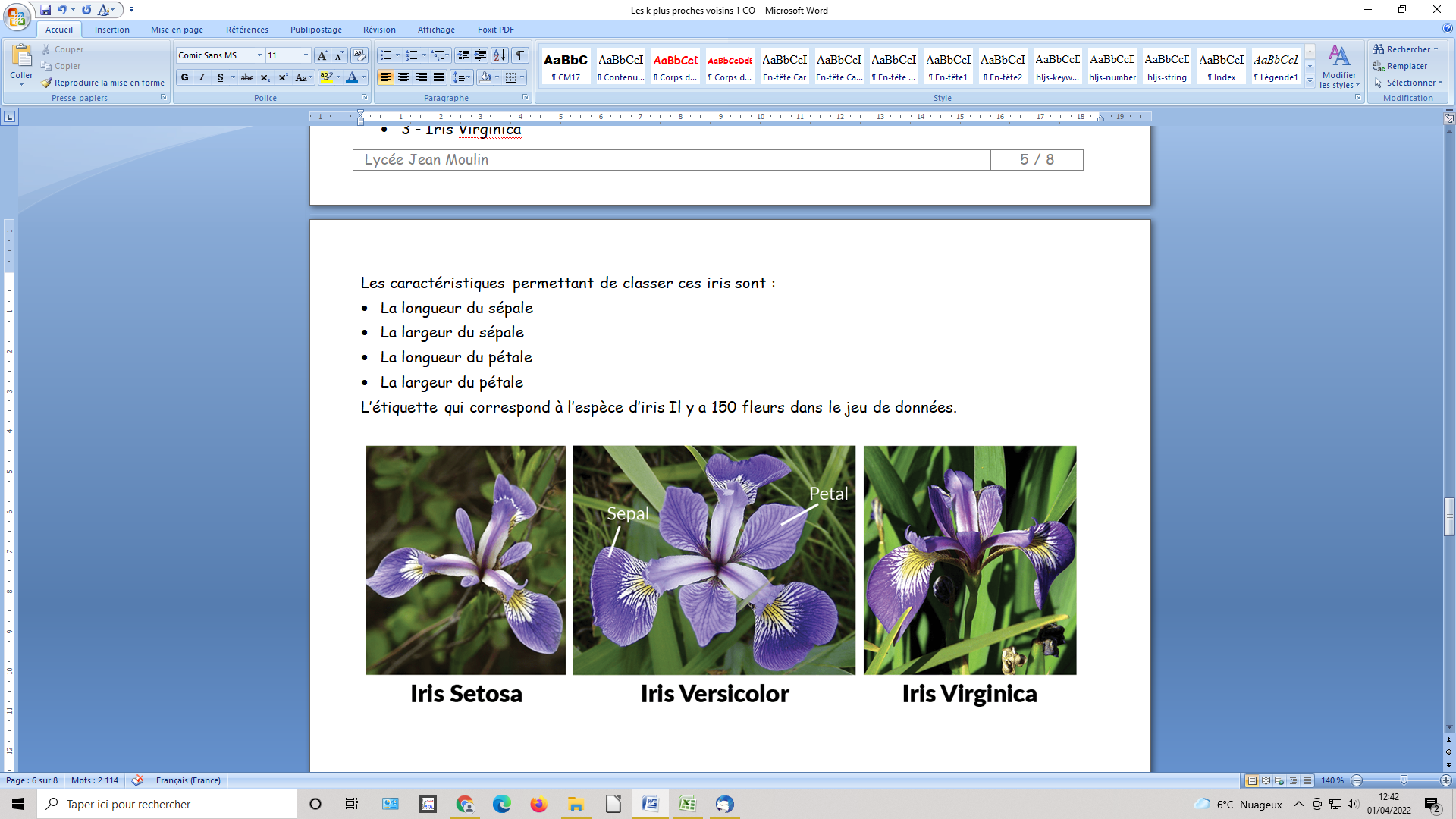
* Calculez la distance entre le point **A={"age":50.0,"taille":168.0,"poids":59.0}** et le point **B={"age":51.0,"taille":183.0,"poids":80.0}**.

**=**25,82

* Donnez le code python pour coder le calcul de cette distance.

Sqrt((A["age"]-B["age"])\*\*2+(A["taille"]-B["taille"])\*\*2+(A["poids"]-B["poids"])\*\*2)

Application au classement des Iris



* Quelle sont les différences entre ces 3 **iris** qui ont atteint leur taille maximale ?

Nous possédons un jeu de données (fichier **iris.csv**) avec les caractéristiques de **150 iris**.

**Dans le fichier, les caractéristiques (features) des 150 iris sont :**

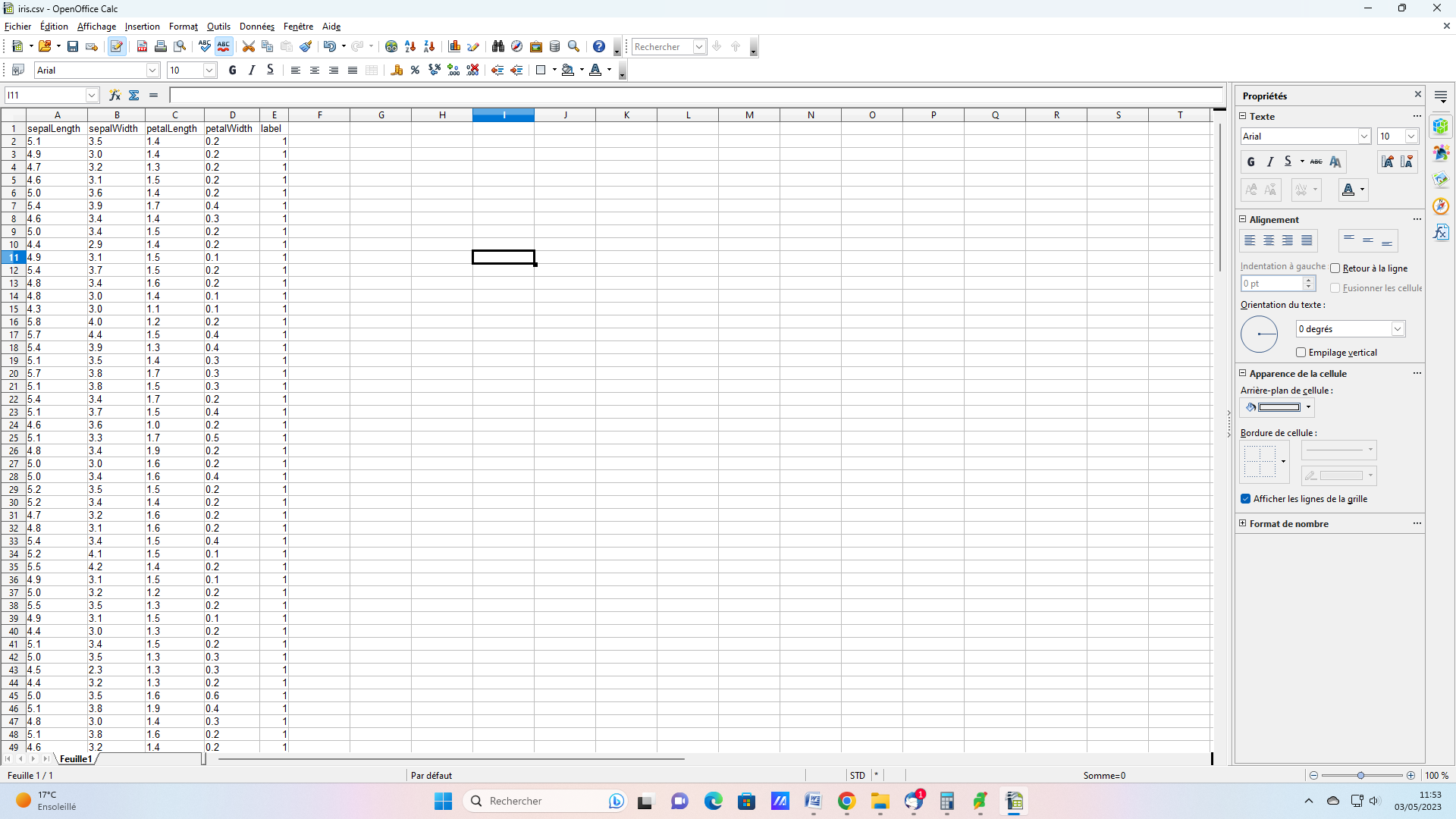
* La longueur du sépale
* La largeur du sépale
* La longueur du pétale
* La largeur du pétale

Nous allons appliquer l’algorithme des **k plus proches voisins** sur ce jeu de données afin de classer des iris qui ne sont pas présents dans le fichier **iris.csv**

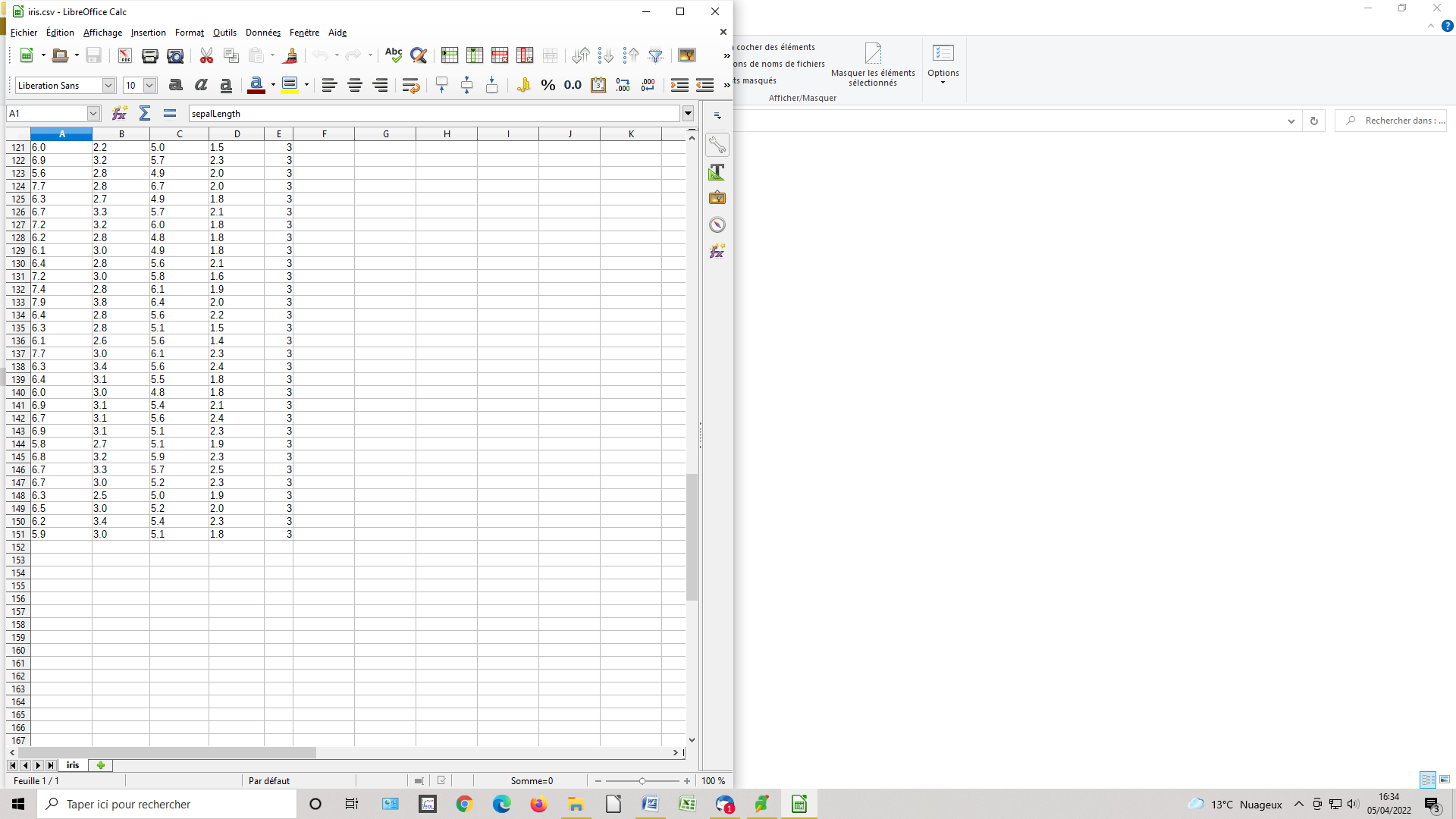
**Dans le fichier, les 150 iris sont classés selon trois étiquettes (labels) :**

* 1:Iris Setosa
* 2:Iris Versicolor
* 3:Iris Virginica

Fichier **iris.csv :**

****

**...**



Lenght : longueur – Width : largeur

Le fichier **kppv.py** contient déjà la fonction lecture qui lit le fichier **iris.csv** et renvoie la liste des caractéristiques (features), la liste des vecteurs de caractéristiques (vectors) et la liste des étiquettes associées aux différents iris (labels).

* Testez la fonction lecture et donnez le contenu des variables suivantes :

feat = ['sepalLength', 'sepalWidth', 'petalLength', 'petalWidth', 'label']

features = ['sepalLength', 'sepalWidth', 'petalLength', 'petalWidth']

* Testez la fonction lecture et donnez un échantillon du contenu des variables suivantes :

li (la première) = ['6.3', '3.3', '6.0', '2.5', '3']

vectors = [{'sepalLength': 5.1, 'sepalWidth': 3.5, 'petalLength': 1.4, 'petalWidth': 0.2}, {'sepalLength': 4.9, 'sepalWidth': 3.0, 'petalLength': 1.4, 'petalWidth': 0.2}]

labels = ['1', '1', '1', '1']

**Code 3\*** : Testez la fonction distance avec distance(premier\_iris,vectors[0],features) et valider son fonctionnement.

premier\_iris={'sepalLength':5.1,'sepalWidth':3.5,'petalLength':1.4,'petalWidth':0.2}

0.0

**Code 4\*** : Écrivez la fonction distanceAll qui renvoie une liste contenant les distances entre un vecteur de caractéristiques et tous les échantillons du jeu de données.

Cette fonction prend en entrée :

* un vecteur de caractéristiques d’un **nouvel iris** (vector) : pour l’instant on prendra un **iris connu**.
* la liste des vecteurs de caractéristiques (échantillions) du fichier **iris.csv** (vectors)
* la liste des caractéristiques (features)

Cette fonction renvoie :

* une liste (L\_distances) contenant les distances entre vector et les vectors

La fonction distanceAll appellera la fonction distance.

Testez la fonction avec distanceAll(premier\_iris,vectors,features) qui doit vous renvoyer L\_distances = [0.0, 0.5385164807134502, 0.509901951359278, 0.648074069840786, 0.1414213562373093, ...]

**Code 5\*\*** : Écrivez la fonction k\_plus\_proches qui renvoie une liste contenant les indices des k plus proches voisins.

Cette fonction prend en entrée :

* la liste des distances (L\_distances)
* une valeur k

Cette fonction renvoie :

* une liste (k\_indices\_plus\_proches) contenant les **indices** des k distances les plus petites, correspondant aux vecteurs de caractéristiques (échantillons) les plus proches.

Testez la fonction avec k\_plus\_proches(L\_distances,5) qui doit vous renvoyer k\_indices\_plus\_proches =[0, 17, 4, 39, 27].

Aide : vous pouvez utiliser la méthode index sur une liste ainsi que la fonction sorted.

**Code 6\*\*** : Écrivez la fonction classification qui donne une étiquette à un vecteur de caractéristiques en fonction de ses k plus proches voisins.

Cette fonction prend en entrée :

* la liste de tous les labels (labels)
* la liste contenant les indices des k distances les plus petites (k\_indices\_plus\_proches)

Cette fonction renvoie :

* l’étiquette attribuée à l'iris

Testez la fonction avec classification(labels,['2','3','3','1','3','1','3','2']) qui doit renvoyer l'étiquette (label) "3".

Testez la fonction avec classification(labels,k\_indices\_plus\_proches) qui doit renvoyer l'étiquette (label) "1".

* Testez également l'étiquette (label) renvoyée pour le dernier iris afin de valider le fonctionnement de votre code.
* Testez le fonctionnement avec un nouvel iris qu'un expert a classé comme étant un iris Virginica.

new\_iris={"sepalLength":4.0,"sepalWidth":4.0,"petalLength":4.0,"petalWidth":4.0}

* Testez le fonctionnement avec différentes valeurs de k impaires.
* A partir de quelle valeur de k impaire on commence à trouver des labels différents dans la liste ?