MINISTÈRE DE L'ÉDUCATION NATIONALE, DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

****

ÉCOLE NATIONALE D'INGÉNIEURS

ABDERHAMANE BABA TOURE (ENI-ABT)

RÉPUBLIQUE DU MALI

****

Un peuple - Un but - Une foi

EXPOSE

Thème

ALGORITHME K-VOISINS LES PROCHES

K-NEAREST NEIGHBORS

Présenté et soutenu par : Souleymane Kodjo

Abdrahamane Idrissa Doumbia

Ouzairou Djiré

Responsable pédagogique : Dr A. SIDIBE

Sommaire

Chapitre 1 : Introduction

L'Apprentissage automatique ou machine Learning (en anglais) peut être définie comme une branche de l'intelligence artificiel englobant de nombreux méthode permettant de crée automatiquement les modèles à partir de données. Ces méthodes sont en fait des algorithmes.

Un programme informatique traditionnel effectue une tache en suivant des instruction précise et donc systématiquement de la même façon .Au contraire, un système machine Learning ne suit pas d'instructions, mais à donner la capacité aux ordinateur d'apprendre a partir de données, c'est a dire d'améliore leur performance a résoudre les taches sans être explicitement programmée pour chacun .En conséquent ces performance s'améliore au fils de son « Entrainement » a mesure de l'algorithmique est exposée a d'avantage de donnée .

1.1 Cas d'usage et applications

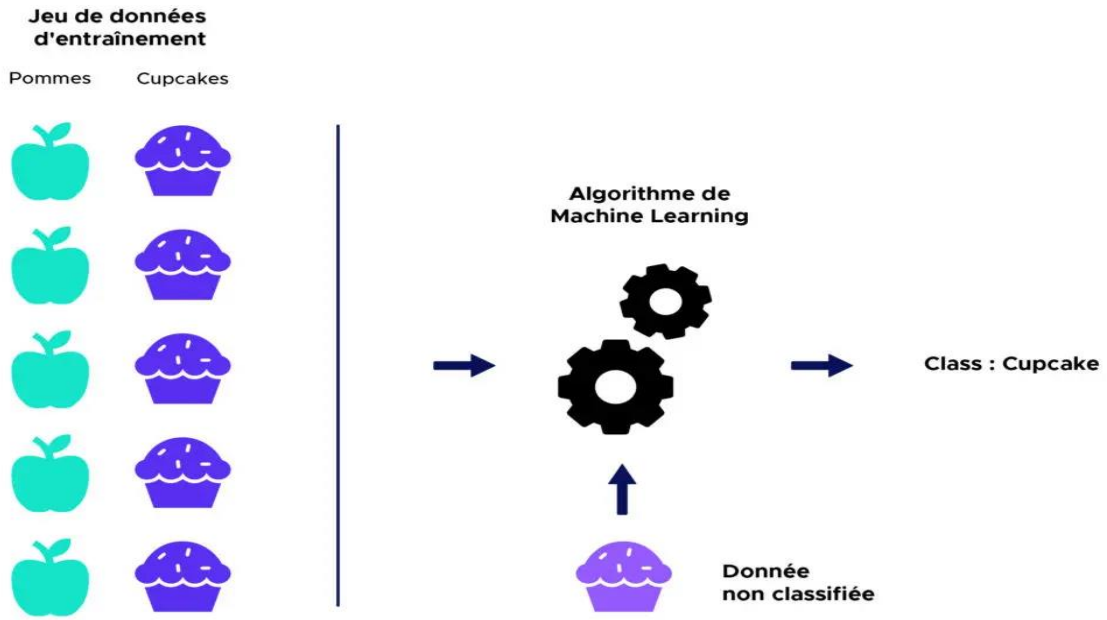
Le Machine Learning alimente de nombreux services modernes très populaires pour recommandation de produit. On peut citer comme : Netflix, YouTube, Amazon …

Il en va de même pour les moteurs de recherche web Google ou Baidu et les assistants vocaux tels Siri et Alexa.

Il est aussi utilisé dans prédiction des prix, détection des spam …

Chapitre 2 : Algorithme K voisins les plus proches

L'algorithme des k plus proches voisins ou encore appelée KNN de l'anglais K-Nearest Neighbors, appartient à la famille des algorithmes d'apprentissage automatique (Machine Learning). L'idée d'apprentissage automatique ne date pas d'hier, puisque le terme de Machine Learning a été utilisé pour la première fois par l'informaticien américain Arthur Samuel en 1959. L'algorithme des k plus proches voisins est un algorithme de d'apprentissage supervisé. En apprentissage supervisé, un algorithme reçoit un ensemble de données qui est étiqueté avec des valeurs de sorties correspondantes sur lequel il va pouvoir s'entraîner et définir un modèle de prédiction. Cet algorithme pourra par la suite être utilisé sur de nouvelles données afin de prédire leurs valeurs de sorties correspondantes.



L'algorithme d'apprentissage supervisé permet de répondre de deux problèmes :

2.1 Classification « Quel classe ? »

L'algorithme de classification est d'abord alimenté par un nombre fini d'exemple catalogués qu'i utilise pour son apprentissage. Cet apprentissage lui permet de sélectionner une « hypothèse », ou règle censée commettre peu d'erreurs de classification sur ces exemples futures

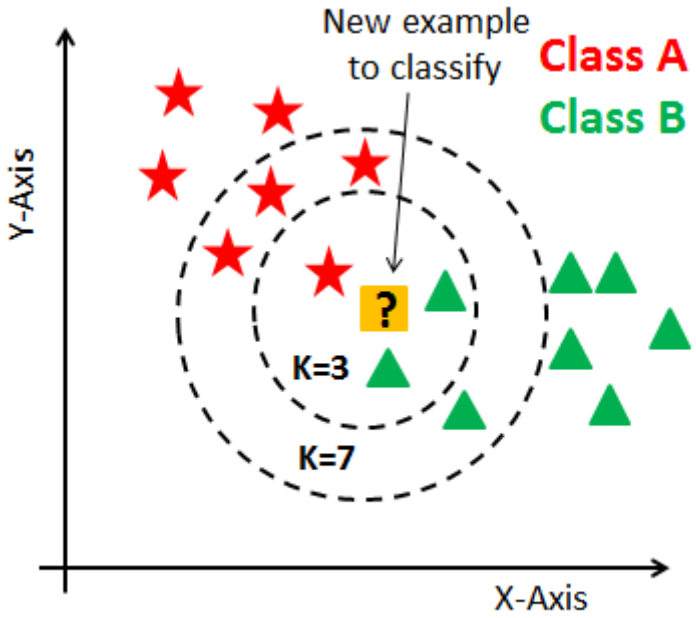
Le résultat est une classe d'appartenance. Un objet d'entrée est classifié selon le résultat majoritaire des statistiques de classes d'appartenance de ses k plus proches voisins.

2.2 Régression « Combien ? »

L'algorithme de régression est alimenté a partir de la variable cible ou de la variable expliquer (Y), le model a pour but de faire une prédiction grâce a des variables dites explicatives (X) ou prédictives

Le résultat est la valeur pour cet Objet. Cette valeur est la moyenne des valeurs des k les plus proches.

KNN est une méthode non paramétrique dans laquelle le model mémorise les observations de l'ensemble d'apprentissage pour la classification des données de l'ensemble de test



En effet , cet algorithme est qualifiée comme paresseux , car i n'apprend rien pendant la phase d'entrainement .Pour prédire la classe d'une nouvelle données d'entrée , il va chercher ses K voisins les plus proches (en utilisant la distance euclidienne , ou autres) et choisira la classe des voisins majoritaires.

Chapitre 3 : Principe de l'algorithme

L'algorithme de k plus proches voisins ne nécessite pas de phase d'apprentissage à proprement parler, il faut juste stocker le jeu de données d'apprentissage. Soit un ensemble E contenant n données labellisées : $E = \{(y_i, \vec{x}_i)\}$ avec i compris entre 1 et n, où y_i correspond à la classe (le label) de la donnée i et où le vecteur $\vec{x}_i$ de dimension p ($\vec{x}_i = (x_{1i}, x2_i, \ldots, x_{pi})$) représente les variables prédicatrices de la donnée i. Soit une donnée u qui n'appartient pas à E et qui ne possède pas de label (u est uniquement caractérisée par un vecteur x_u de dimension p). Soit d'une fonction qui renvoie la distance entre la donnée u et une donnée quelconque appartenant à E. Soit un entier k inférieur ou égal à n. Voici le principe de l'algorithme de k plus proches voisins :

3.1 Sélectionne le nombre k de voisin

On fixe le nombre de k voisin, On détecte les k voisins les plus proches des nouvelles données l'entrée que l'on veut classe

3.2 On calcule les distances entre la donnée u et chaque donnée appartenant à E à l'aide de la fonction d

Il est possible d'utiliser différents types de distance : euclidienne, Manhattan, ...

3.2.1 Distance Euclidienne

- Sur une droite graduée :

Distance entre A et B est : d(A,B) = |abscisse de B – abscisse de A|

= la plus grande abscisse – la plus petite abscisse

On peut donc écrire une fonction distance, avec Python, qui prend en arguments d'entrée deux coordonnées x1 et x2 et qui renvoie la valeur de la distance entre les deux points de coordonnées x1 et x2

- Dans un plan :

On suppose que le plan est muni d'un repère orthonormal (O;I;J).

Soient A $(x_a ; y_a)$ et B $(x_b ; y_b)$ deux points du plan, la distance euclidienne entre A et B est :

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

3.2.2 Distance Manthattan

La distance de Manhattan a été utilisée dans une analyse de régression en 1757 par Roger Joseph Boscovich. L'interprétation géométrique remonte à la fin du XIXe siècle et au développement de géométries non euclidiennes, notamment par Hermann Minkowski et son

inégalité de Minkowski, dont cette géométrie constitue un cas particulier, particulièrement utilisée dans la géométrie des nombres (Minkowski 1910).

La distance de Manhattan est appelée aussi taxi-distance, est la distance entre deux points

Parcourue par un taxi lorsqu'il se déplace dans une ville où les rues sont agencées selon un réseau ou quadrillage. Un taxi-chemin est le trajet fait par un taxi lorsqu'il se déplace d'un nœud du réseau à un autre en utilisant les déplacements horizontaux et verticaux du réseau.

Entre deux points A et B, de coordonnées respectives (x_A, y_A) et (x_B, y_B)

La distance de Manhattan est définie par :

$$d\,(A,B) \;=\; |\,x_A \;-\; x_b\,| \;+\; |\,y_A \;-\; y_b\,|.$$

3.3 On retient les k données du jeu de données E les plus proches de u

3.4 On attribue à u la classe qui est la plus fréquente parmi les k données les plus proches

Suivant que l'on raisonne sur une, deux, trois dimensions, le calcul de la distance entre deux points est plus au moins simple.

Pour appliquer ce principe, il faudra :

- ✓ Évaluer la distance qui sépare le nouvel élément de chacun des autres points de l'ensemble E. Chaque point de l'ensemble est caractérisé par son indice i.
- ✓ Stocker ces valeurs de distance d dans une liste du type : [[d,i],[...],...], où d est la distance qui sépare le nouvel élément du point d'indice i.
- ✓ Trier la liste selon les valeurs des distances d.
- ✓ Choisir les k premiers points de la liste triée qui sont donc les k-plus proches voisins.
- ✓ Assigner une classe au nouvel élément en fonction de la majorité des classes représentées parmi les k-plus proches voisins.

On aura donc besoin de trois fonctions :

- ✓ une fonction distance pour calculer la distance entre deux points de coordonnées connues.
- ✓ Une fonction kvoisins qui détermine les k-plus proches voisins d'un nouvel élément.
- ✓ Une fonction predire_classe qui détermine le résultat majoritaire des classes d'appartenance des k-plus proches voisins et assigne la classe du nouvel élément à cette classe majoritaire.

3.5 Ecriture algorithmique

On peut schématiser le fonctionnement de K-NN en l'écrivant en pseudo-code suivant :

Début Algorithme
Données en entrée :

- Un ensemble de données D .
- Une fonction de définition distance d.
- Un nombre entier K

Pour une nouvelle observation X dont on veut prédire sa variable de sortie y Faire :

1. Calculer toutes les distances de cette observation X avec les autres observations du jeu de données
2. Retenir les K observations du jeu de données D les proches de X en utilisation la fonction de calcul de distance d
3. Prendre les valeurs de y des K observations retenues :
    1. Si on effectue une régression, calculer la moyenne (ou la médiane) y de retenues
    2. Si on effectue une classification, calculer le mode de y retenues
4. Retourner la valeur calculée dans l'étape 3 comme étant la valeur qui a été prédite par K-NN pour l'observation X .

Fin Algorithme

Chapitre 4 : Etude d'un Exemple

4.1 Jeux de données

Nous avons choisi ici de nous baser sur le jeu de données « iris de Fisher ».

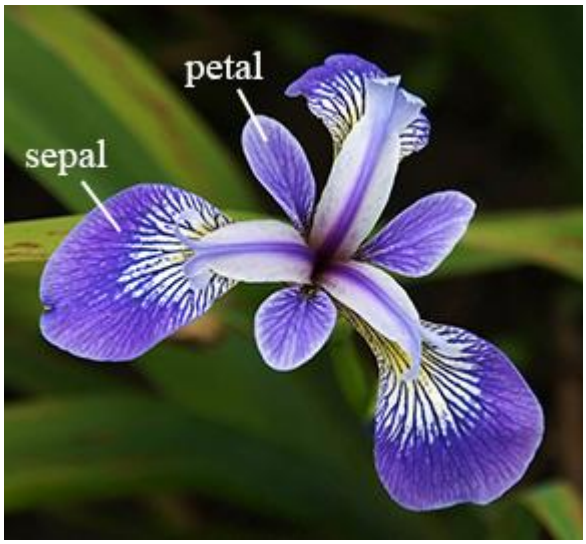


Photo 1 IRIS

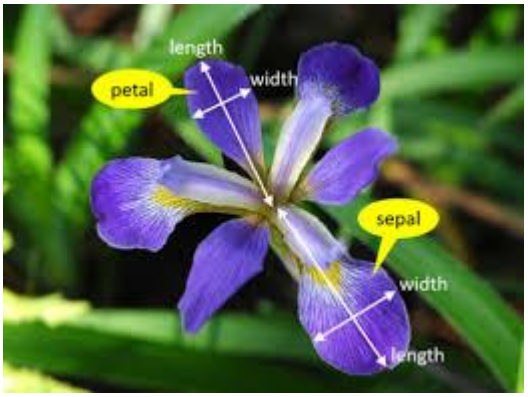


Photo 2 IRIS

Ce jeu de données est composé de 50 entrées, pour chaque entrée nous avons :

▷ la longueur des sépales (en cm)

▷ la largeur des sépales (en cm)

▷ la longueur des pétales (en cm)

▷ la largeur des pétales (en cm)

▷ l'espèce d'iris : Iris setosa

Iris virginica

Iris versicolor

4.2 Les logiciels utilisés

Anaconda navigator

Jupyter notebook

Chrome

4.3 Bibliothèques Python utilisées

4.3.1 Pandas

Pandas est une excellente librairie pour importer vos tableaux Excel (et autres formats) dans Python dans le but de tirer des statistiques et de charger votre Dataset dans Sklearn.

4.3.2 NumPy

Numpy est la librairie qui permet de créer et manipuler des matrices ou tableau simplement et avec efficacité. Ainsi le calcul matriciel représente l'essentiel du Machine Learning. Il est important de le comprendre, mais les fonctions présentes dans Numpy font les calculs matriciels à notre place… Magique !

4.3.3 Matpotlib

Matplotlib est la librairie qui permet de visualiser nos Datasets, nos fonctions, nos résultats sous forme de graphes, courbes et nuages de points

4.3.4 Sklearn

Sklearn est la librairie qui contient toutes les fonctions de l'état de l'art du Machine Learning. On y trouve les algorithmes les plus importants ainsi que diverses fonctions de pre-processing.

4.4 Importation des bibliothèques

Importation des bibliothèques requis

Entrée [1]:

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from sklearn.neighbors import KNeighborsClassifier
```

4.5 Importation des données

Importation des données

Entrée [3]:

```
1 df = pd.read_csv(r"C:\Users\Kodjo\data_science\iris_min.csv")
```

Entrée [ ]:

```
1
```

4.6 Formatage

Ici on va remplacer

Iris-setosa par 0

Iris-virginica par 1

Iris-versicolor par 2

Formatage

Entrée [4]:
```
df['class'] = df['class'].replace("Iris-setosa", 0)
df['class'] = df['class'].replace("Iris-virginica", 1)
df['class'] = df['class'].replace("Iris-versicolor", 2)
```

Entrée [ ]:
```

```

4.7 Première visualisation des données

Une fois le fichier csv modifié, il est possible d'écrire un programme permettant de visualiser les données sous forme de graphique (abscisse : "petal_length", ordonnée : "petal_width")

Visualisation en nuage de point

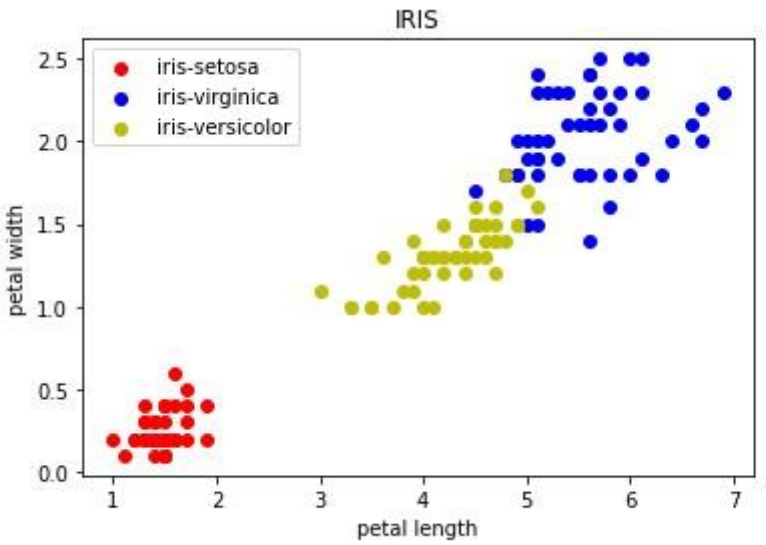
Entrée [ ]:
```
x=df['petal_length']
y=df['petal_width']
label=df['class']
plt.scatter(x[label==0], y[label==0], c='r', label='iris-setosa')
plt.scatter(x[label==1], y[label==1], c='b', label='iris-virginica')
plt.scatter(x[label==2], y[label==2], c='y', label='iris-versicolor')
plt.xlabel("petal length")
plt.ylabel("petal width")
plt.title("IRIS")
plt.legend()
plt.show()

```

Entrée [ ]:
```

```

Graphe :



4.8 Echantillon a prédire

```
# Echantillion predire
```

Entrée [6]:
```
#Les attribut de echantillion
```

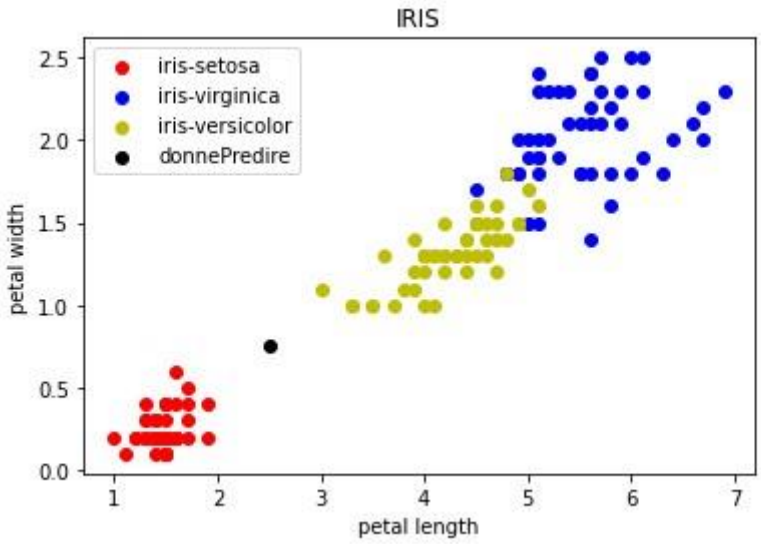
Entrée [7]:
```
#Valeur
longueur = 2.5
largeur = 0.75
k=3
```

4.9 Visualisation de Echantillon a prédire dans le graphe

```
# Visualisation
```

Entrée [8]:
```
plt.scatter(x[label==0], y[label==0], c='r', label='iris-setosa')
plt.scatter(x[label==1], y[label==1], c='b', label='iris-virginica')
plt.scatter(x[label==2], y[label==2], c='y', label='iris-versicolor')
#On a joute au graphe
plt.scatter(longueur, largeur, color='k')
plt.xlabel("petal length")
plt.ylabel("petal width")
plt.title("IRIS")
plt.legend()
plt.show()
```

Graphe



4.10 Séparation du jeux de donnée en training set et testing set

Separation du jeux de donnée en training set et testing set

Entrée [25]:
```
#feature
feature_columns = ['petal_length', 'petal_width']
X = df[feature_columns].values
```

Entrée [26]:
```
#target
label = df['class']
```

Entrée [27]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,label, train_size=0.8, random_state=0)
```

Entrée [ ]:
```

```

4.11 Algorithmique KNN pour K=2

Utilisation d'algorithme KNN pour K=2 et le taux d'erreur de prédiction

Algorithme KNN

Entrée [28]:
```
#algo KNN
k=2
#Creation de model de classification
model = KNeighborsClassifier(n_neighbors=k)
#(data, target)
model.fit(X_train,y_train)
model.score(X_test, y_test)
```

Out[28]: 0.9333333333333333

4.12 Optimisation du taux d'erreurs

Optimisation du score sur les données test

Entrée [30]:
```
errors = {}
valeur_k = list(range(2,15))
```

Entrée [31]:
```
for k in valeur_k:
    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(X_train, y_train)
    score = 100*(1-model.score(X_test, y_test))
    error = {k:score}
    errors.update(error)
```

Entrée [32]:
```
errors
```

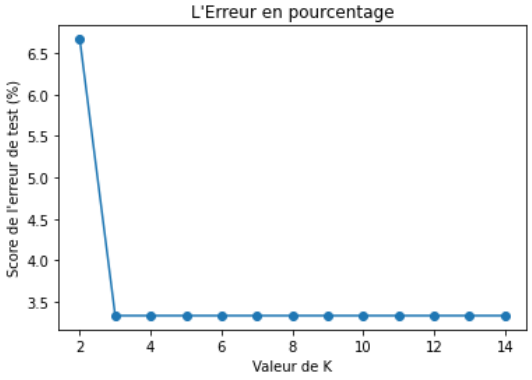
Out[32]:
```
{2: 6.666666666666665,
 3: 3.3333333333333326,
 4: 3.3333333333333326,
 5: 3.3333333333333326,
 6: 3.3333333333333326,
 7: 3.3333333333333326,
 8: 3.3333333333333326,
 9: 3.3333333333333326,
 10: 3.3333333333333326,
 11: 3.3333333333333326,
 12: 3.3333333333333326,
 13: 3.3333333333333326,
 14: 3.3333333333333326}
```

- Sous Forme Graphique

Visualiser du Score d'erreur sous forme graphique

Entrée [34]:
```
plt.plot( valeur_k, list(errors.values()), 'o-' )
plt.title("L'Erreur en pourcentage")
plt.xlabel("Valeur de K")
plt.ylabel("Score de l'erreur de test (%)")
```

Out[34]:
```
Text(0, 0.5, "Score de l'erreur de test (%)")
```



4.13 Création d'une fonction python résultat qui retourne la prédiction

C'est la classe dans lequel échantillon a appartient.

Entrée [ ]:
```
#Fonction resltat
def resultat():
    if prediction == 0 :
        return "Resultat : setosa"
    if prediction == 1 :
        return "Resultat : virginica"
    if prediction == 2 :
        return "Resultat : verginicolor"
```

4.14 Visualisation finale et prédiction

Entrée [ ]:
```
#Donnée d'entr"
longueur = 2.5
largeur = 0.75
```

Entrée [36]:
```
#L'algorithmique est deja importée
d = list(zip(x,y))
model = KNeighborsClassifier(n_neighbors=3)
model.fit(d, label)
prediction = model.predict([[longueur, largeur]])
```

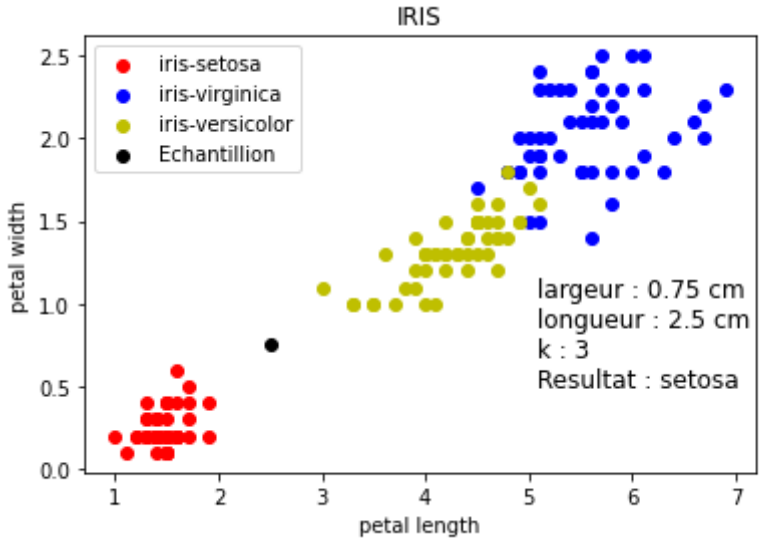
Entrée [40]:
```
#Fonction resltat
def resultat():
    if prediction == 0 :
        return "Resultat : setosa"
    if prediction == 1 :
        return "Resultat : virginica"
    if prediction == 2 :
        return "Resultat : verginicolor"
resultat()
```

Out[40]: 'Resultat : setosa'

Visualisation dans

Entrée [14]:
```
plt.scatter(x[label==0], y[label==0], c='r', label='iris-setosa')
plt.scatter(x[label==1], y[label==1], c='b', label='iris-virginica')
plt.scatter(x[label==2], y[label==2], c='y', label='iris-versicolor')
plt.scatter(longueur, largeur, color='k', label="Echantillion")
res = resultat()
plt.text(5,0.5, f" largeur : {largeur} cm\n longueur : {longueur} cm\n k : {k}\n {res}", fontsize=12)
plt.xlabel("petal length")
plt.ylabel("petal width")
plt.title("IRIS")
plt.legend()
plt.show()
```

Graphe : Le résultat de prédiction sur le graphe



5 Conclusion

Dans ce rapport, vous avez découvert l'algorithme k-NN qui est l'un des algorithmes de l'apprentissage automatique supervisée il est simple et facile à mettre en œuvre. il n'a aucune hypothèse sur les données (linéaires, affines…) en plus de cela il est polyvalent. Il peut être utilisé pour la classification, la régression.

Vous avez appris également que :

- K-NN stocke tout le jeu de données pour effectuer une prédiction,
- K-NN ne calcule aucun modèle prédictif et il rentre dans le cadre du Lazy Learning