4A : RESEAUX DE NEURONES

Nom et Prénom: SHE Houyu, GUIZANI Ahmed , MOUDOUB Lila
Classe: 4A41

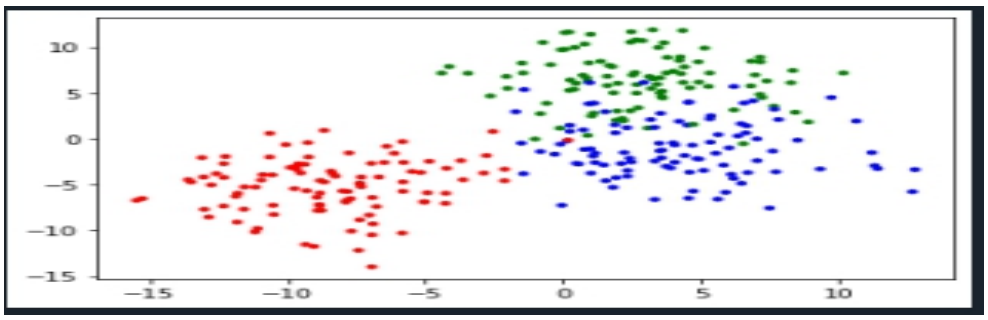
TP1 (durée : 1h30)
Algorithme des plus proches voisins

1) Analyse des données

Charger la base de données (*dataset.dat*). Diviser la base de données aléatoirement (fonction *train_test_split*) en deux corpus (les classes sont équi-représentées dans les deux corpus) :

- base d'apprentissage (*X_train, y_train*) : 70% des données
- base de test (*X_test, y_test*): le reste (30% des données)

Afficher les données d'apprentissage et de test (fonction *pyplot.show*). Préciser la dimension des exemples, le nombre de classe, le nombre d'exemples d'apprentissage et de test.



Grâce aux fonctions ndim, ou encore size, nous obtenons les résultats suivants :

- Nombre de dimensions X : 2
- Nombre de dimensions Y : 1
- Nombre de classes : 3 (0, 1 et 2)
- Le nombre d'exemples d'apprentissage: 210
- Le nombre d'exemples de test : 90

2) Algorithme du plus-proche-voisin

a) Définir un classifieur du (1-)plus-proche-voisin :

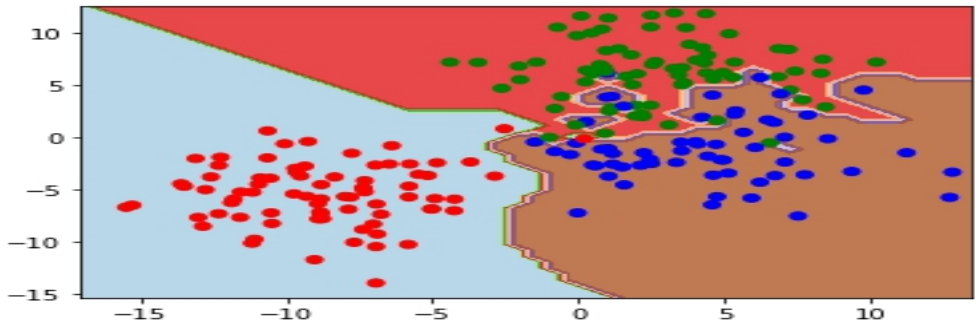
one_NN = KNeighborsClassifier(n_neighbors=1, algorithm='brute')

b) Le régler sur la base d'apprentissage (fonction *fit*).

c) Evaluer le taux de reconnaissance le taux de reconnaissance sur les bases d'apprentissage et de test (fonction *score*). Déterminer la matrice de confusion.

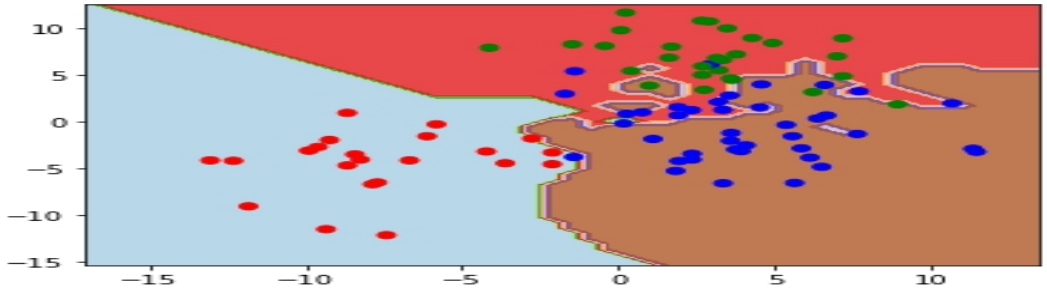
```
le taux de reconnaissance sur les bases d'apprentissage :  1.0
le taux de reconnaissance sur les bases de test :  0.8
La matrice de confusion :
 [[22  0  0]
 [ 0 25  3]
 [ 2 13 25]]
```

d) Afficher les frontières de décision définies par les données d'apprentissage. Vérifier qu'elles correspondent à l'attendu.



Toutes les exemples d'apprentissage sont dans leurs frontières, ainsi que les exemples de test.

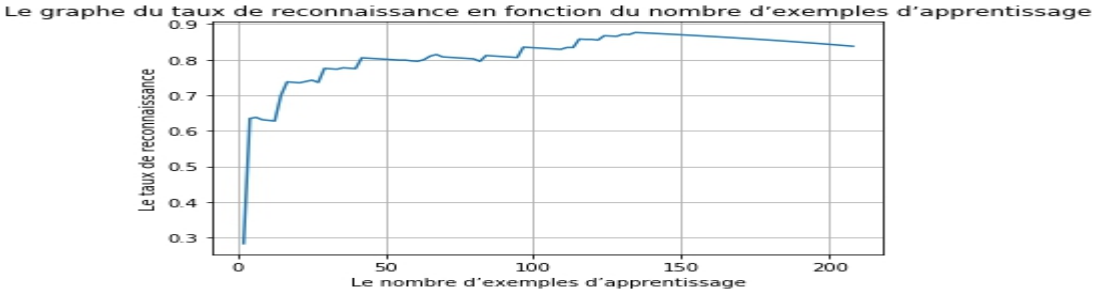
e) Afficher les frontières de décisions et les données de test. Retrouver les erreurs de classification.



```
les erreurs de classification: 6
```

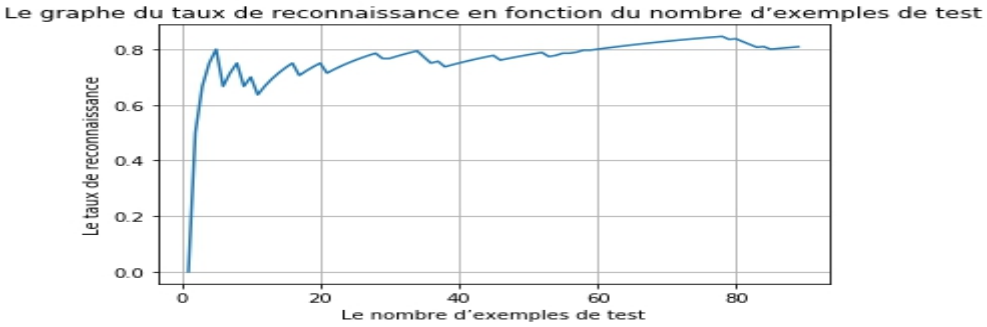
3) Analyse du fonctionnement de l'algorithme

a) On utilise toute la base de test. Générer une base d'apprentissage *X_train1* dont la taille varie de 1% à 100% de *X_train*. Evaluer l'algorithme du ppv sur *X_test*, en utilisant *X_train1*. Stocker le taux de reconnaissance. Tracer le graphe (taux de reconnaissance en fonction du nombre d'exemples d'apprentissage). Expliquer l'influence du nombre d'exemples d'apprentissage sur le taux de reconnaissance et le temps de classification.



le taux de reconnaissance augmente lorsque le nombre d'exemples d'apprentissage augmente et à partir d'un certain nombre d'exemples d'apprentissage(200) il arrive à 100%

b) On utilise toute la base d'apprentissage. Générer une base de test *X_test1* dont la taille varie de 1% à 100% de *X_test*. Evaluer l'algorithme du ppv sur *X_test1*, en utilisant *X_train.* Stocker le taux de reconnaissance. Tracer le graphe (taux de reconnaissance en fonction du nombre d'exemples de test). Expliquer l'influence du nombre d'exemples de test sur le taux de reconnaissance.

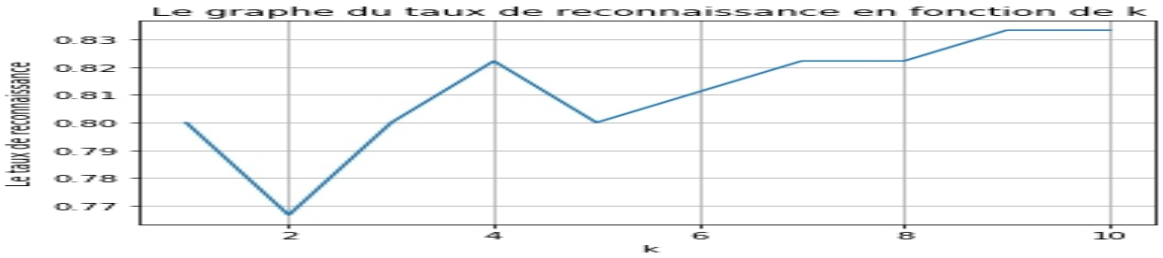


le taux de reconnaissance augmente lorsque le nombre d'exemples de test augmente et à partir d'un certain nombre d'exemples de test (30) il commence à se stabiliser along tours du 80% et a la in il a commence à décroître à cause du sous apprentissage

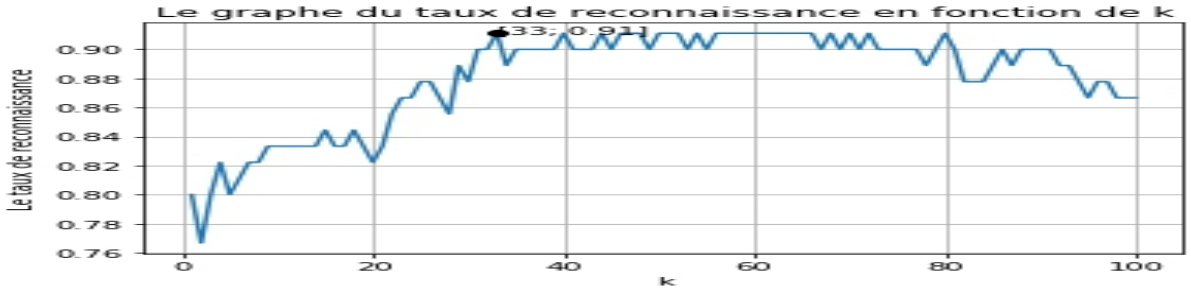
4) Algorithme des k-ppv

a) Tester la fonction *KNeighborsClassifier* en faisant varier le paramètre *k* de 1 à une valeur maximum k_{max} choisie judicieusement et justifiée. Km

k	1	2	3	4	5	6	7	8	9	10
Taux de reconnaissance	0.8	0.76	0.8	0.82	0.8	0.81	0.82	0.82	0.83	0.83



b) Observer l'influence du paramètre k sur le <u>taux d'erreur en test</u> : tracer le graphe (taux de reconnaissance en fonction de *k*). Donner la valeur optimale *k** (qui minimise l'erreur en test) de *k*.

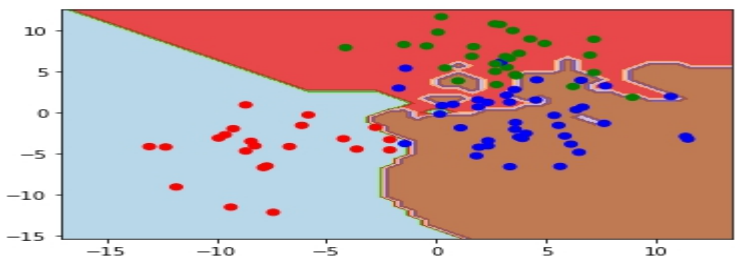


La valeur optimal k=33*

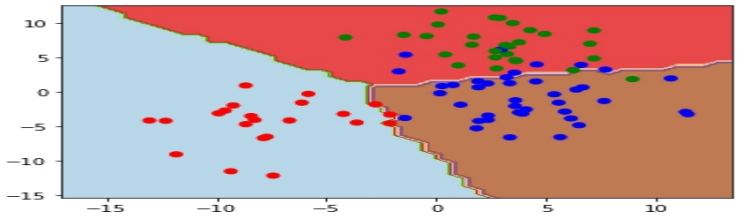
```
k optimal pour taux de reconnaissance max en test =  33
```

c) Afficher les frontières de décision pour :

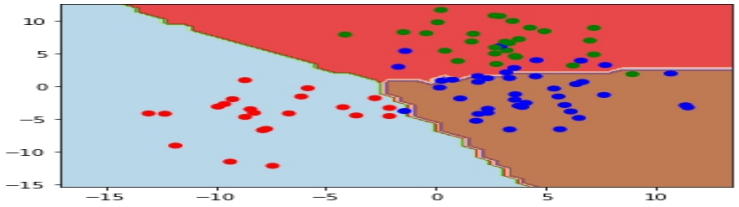
- *k* = 1



- $k = k^*$



- $k = k_{max}$



Conclure sur l'impact de k sur la forme des frontières et justifier qualitativement la valeur k^*.

```
k optimal pour taux de reconnaissance max en test =  33
```

K a amélioré la forme des frontières et spécialement les frontières associé à k*

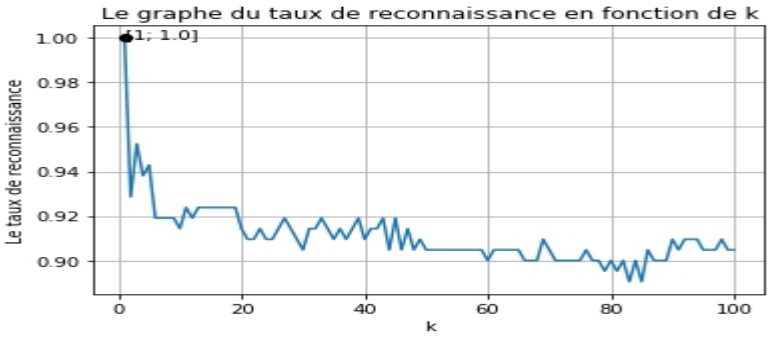
Dans ces trois situations, que peut-on dire du biais et de la variance ?

```
MSE:0.273
Bias:0.270
Variance:0.003
```

Erreur quadratique moyenne MSE = 0.273 , Biais = 0.27 , Variance = 0.003

Le biais et la variance sont très faibles.

d) Observer l'influence du paramètre k sur le <u>taux d'erreur en apprentissage</u> : tracer le graphe (taux de reconnaissance en fonction de k).



e) Le protocole mis en œuvre pour régler le paramètre k est-il satisfaisant ? Justifier votre réponse.

- Le biais est l'erreur provenant d'hypothèses erronées dans l'algorithme d'apprentissage. Un biais élevé peut être lié à un algorithme qui manque de relations pertinentes entre les données en entrée et les sorties prévues (sous-apprentissage).
- La variance est l'erreur due à la sensibilité aux petites fluctuations de l'échantillon d'apprentissage. Une variance élevée peut entraîner un surapprentissage, c'est-à-dire modéliser le bruit aléatoire des données d'apprentissage plutôt que les sorties prévues.

Le protocole mis en œuvre pour régler le paramètre k est satisfaisant, car on a un biais et une variance faibles alors on est loin du sous-apprentissage et du surapprentissage