Jugurtha ASMA et Hylia BOUDAHBA

Rapport Projet n°1

Algorithme

Pour réaliser l'algorithme Kppv, la stratégie était :

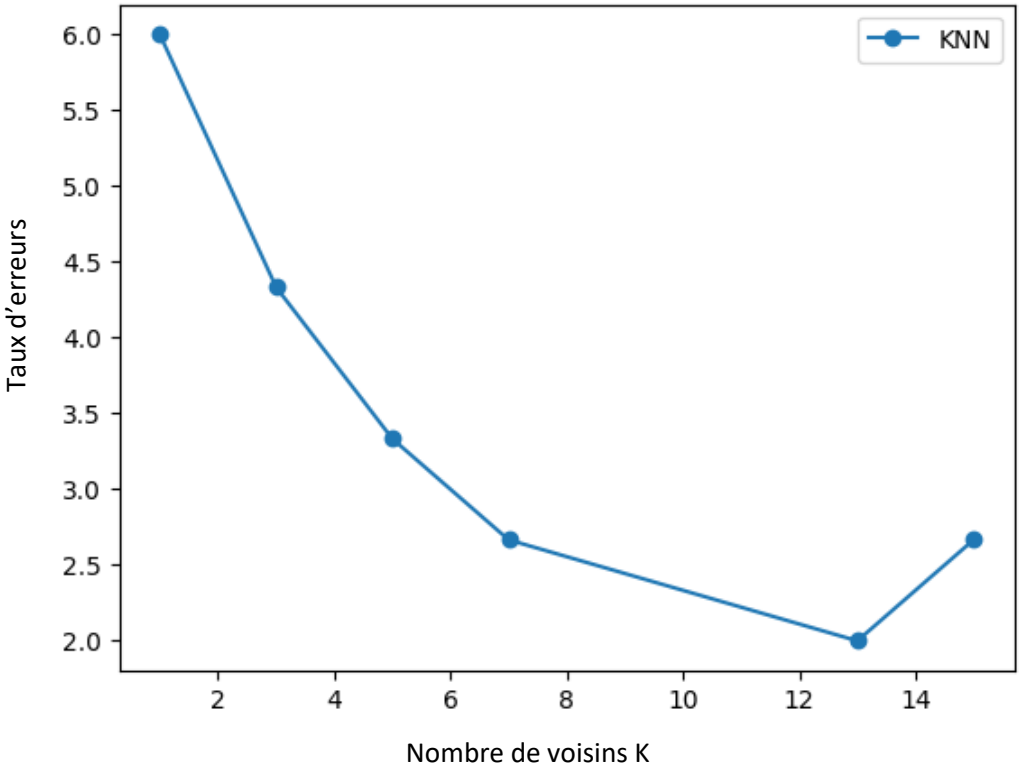
- On crée un tableau où on stockera tous les plus proches voisins
- On parcourt d'abord les individus à classer
- On récupère pour chaque individu sa position.
- On crée un tableau de distances entre l'individu et tous l'apprentissage. (Vide au début)
- On parcourt tous les individus de l'apprentissage pour calculer la distance avec notre individus
    - On va récupérer la position du voisin et sa classe
    - On calcule la distance entre le voisin et cet individu
    - On crée un couple [*distance*, *classVoisin*]
    - On ajoute le couple dans le tableau de distance crée précédemment
- On va maintenant faire un tri par ordre croissant du tableau (de couple) par rapport à la distance
- On crée donc un tableau des plus proches voisins avec uniquement l'attribut *classVoisin* du couple [*distance*, *classVoisin*]
- On ajoute la *classVoisin* qui apparait le plus dans notre tableau de plus proche voisin.

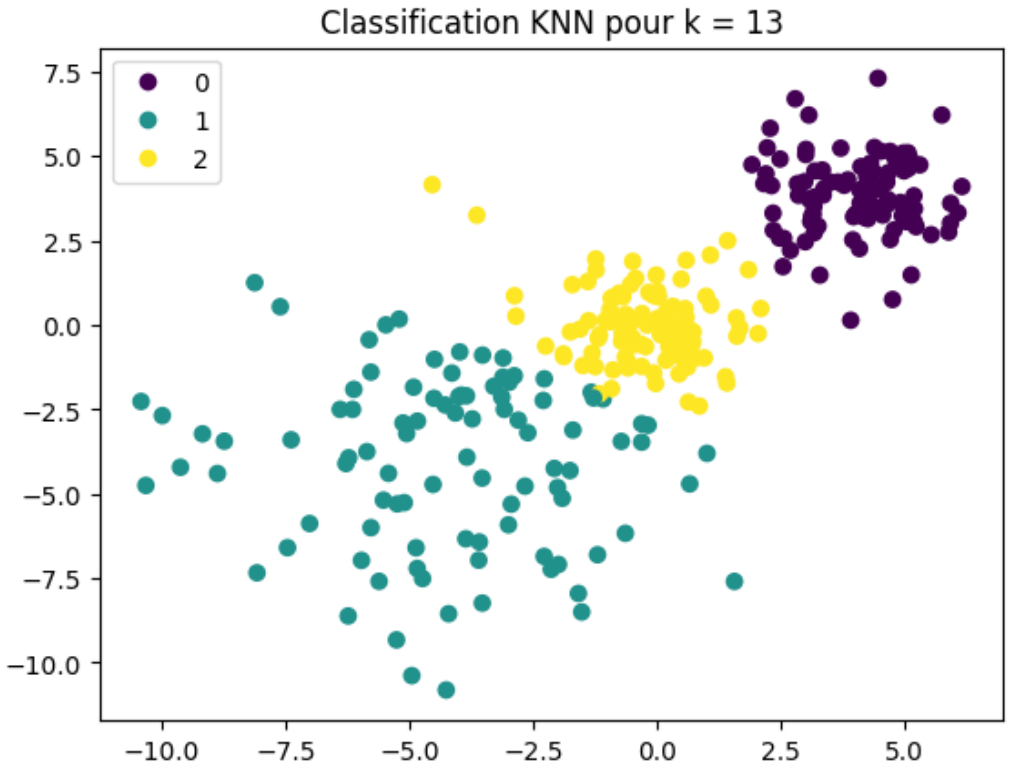
```
def kppv(apprent,class_origine,K,x):
```

apprent qui représentera la matrice contenant les différents individus de l'ensemble d'apprentissage
class_origine indique le numéro de classe de l'individu de l'ensemble d'apprentissage.
k indique le nombre de voisins utilisés dans l'algorithme.
x la matrice qui contiendra les différents individus à classer.

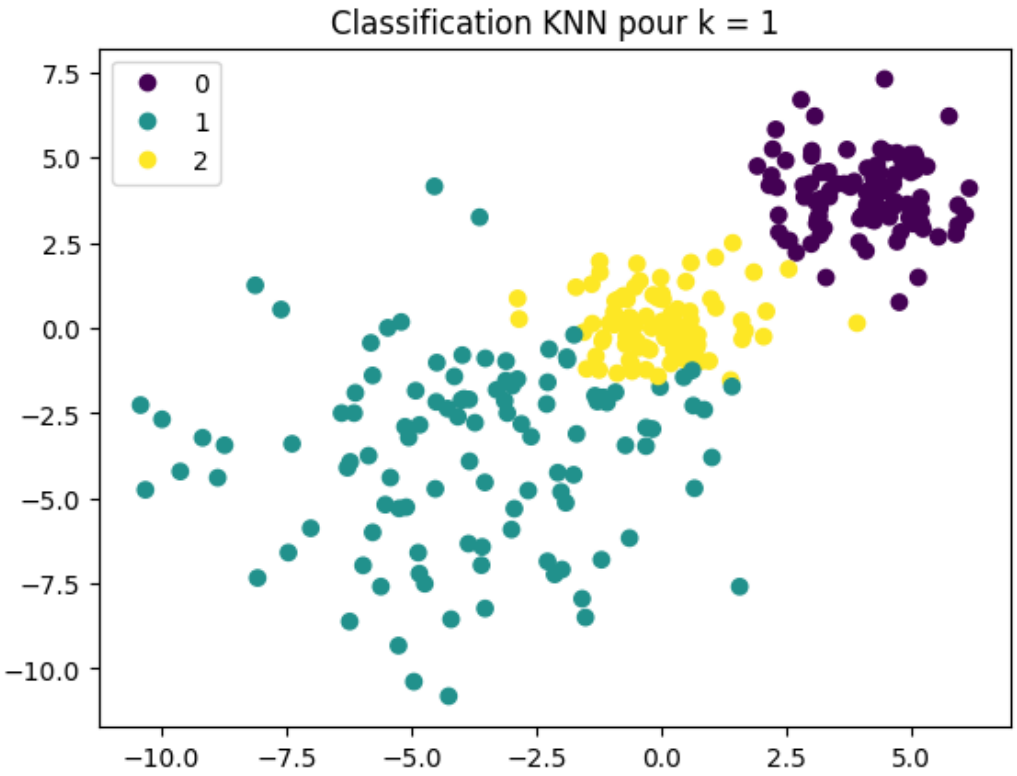
Jugurtha ASMA et Hylia BOUDAHBA

Voici ci-dessous la courbe représentative du taux d'erreurs de l'algorithme kppv en fonction de k. On peut voir que le taux d'erreurs est assez faible, soit le plus grand pourcentage est de 6% et le plus faible est de 2%.





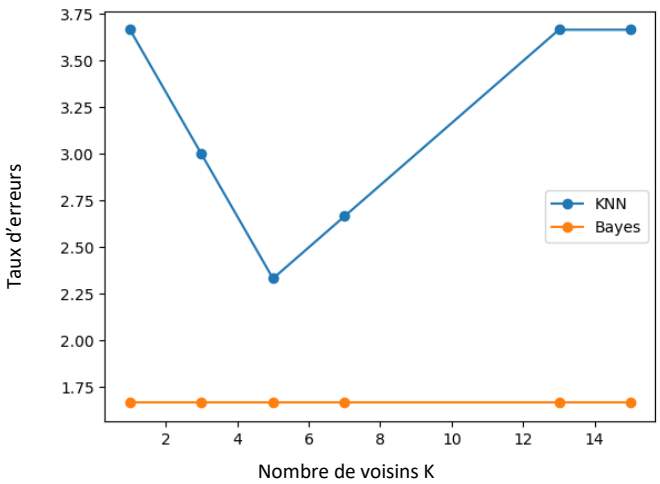
Ci-dessus, le résultat de l'algorithme kppv pour K = 13 donc avec le taux d'erreurs le plus faible (2%)



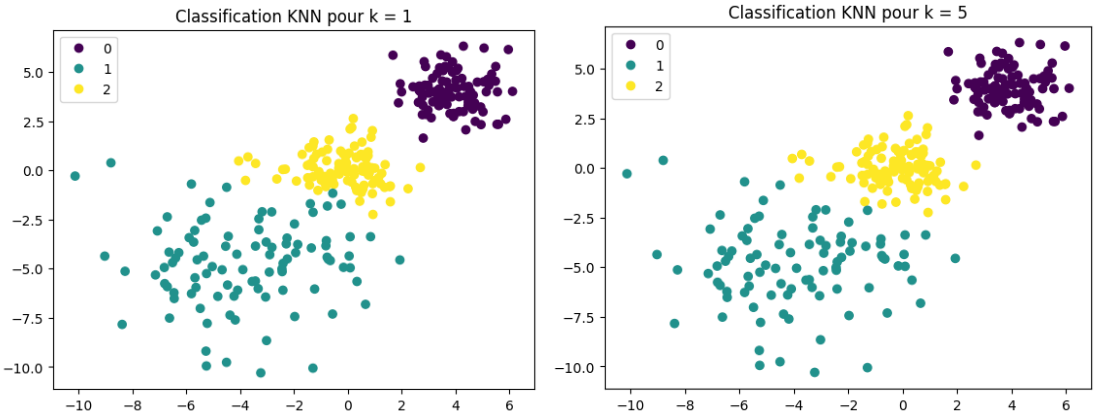
Ci-dessus, le résultat de l'algorithme kppv pour K = 1 donc avec le taux d'erreurs le plus fort (6%).

Influence de la taille de l'ensemble d'apprentissage : taille réduite

k	1	3	5	7	13	15
Taux d'erreurs avec kppv	3.66%	3.0%	2.33%	2.66%	3.66%	3.66%



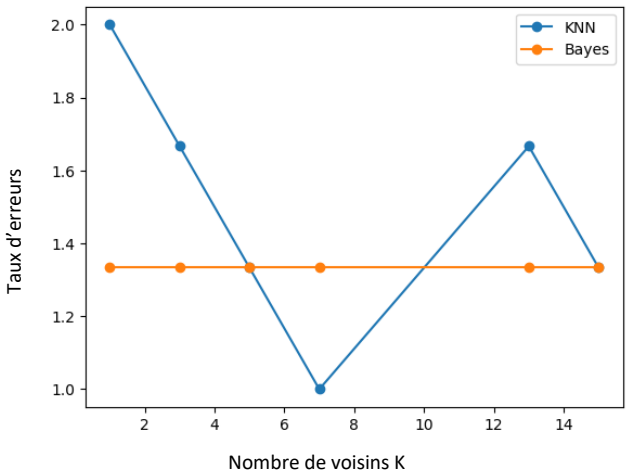
Courbe représentant le taux d'erreurs pour l'algo de Kppv et bayes en fonction de k sur un ensemble de taille réduite



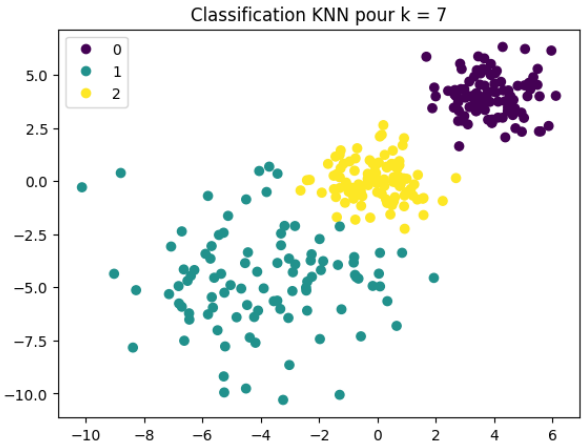
On remarque que le taux d'erreurs est le moins élevés pour k = 5 soit 2.33% pour l'algorithme de Kppv. En revanche, on constate que pour des petites valeurs bayes a un taux d'erreurs constant de 1.6%.

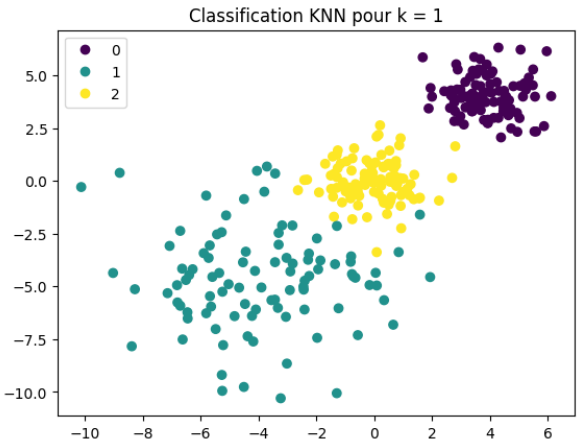
Influence de la taille de l'ensemble d'apprentissage : taille importante

k	1	3	5	7	13	15
Taux d'erreurs avec kppv	2.0%	1.66%	1.33%	1.0%	1.66%	1.33%



Courbe représentant le taux d'erreurs pour l'algo de Kppv et bayes en fonction de k sur un ensemble de taille importante





On remarque que le taux d'erreurs est le moins élevés pour k = 7 soit 1% avec l'algorithme de Kppv. Pour Bayes, il a un taux d'erreurs plus faible entre k compris entre 1 inclus et 5 exclus ainsi que k compris entre 10 exclus et 14 exclus. En revanche pour les valeurs comprise entre 5 exclus et 10 exclus l'algorithme de Kppv a un taux d'erreurs plus faible que celui de bayes. (Soit 1,3%> 1%). Mais on reste sur un pourcentage d'erreurs faibles malgré tout.

Etude de l'évolution de l'erreur en fonction de K

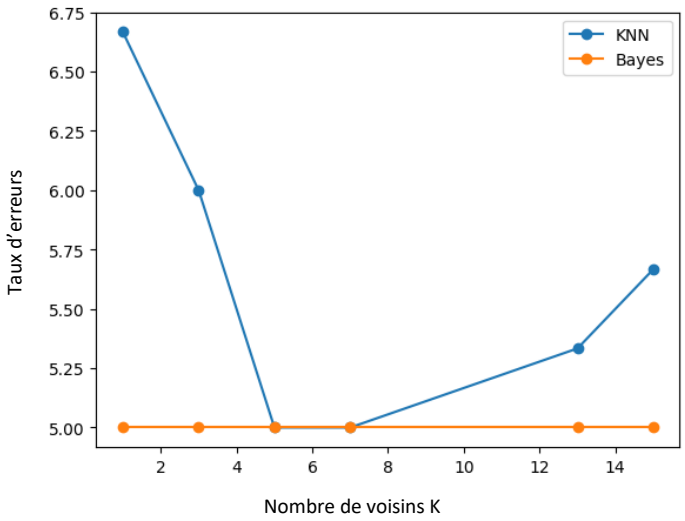
Pour un ensemble d'apprentissage de taille importante, bayes est vraiment meilleurs sur le calcul d'erreurs pour n'importe quelle valeur de K, son taux d'erreurs est plus faible qu'en utilisant l'algorithme de Kppv.

En revanche pour un ensemble d'apprentissage de taille petite, bayes a un taux d'erreurs pour tout k, plus faibles que celui du kppv. Il est donc mieux d'utiliser bayes pour un ensemble d'apprentissage de taille petite. Pour k = 5 et k = 15, le taux d'erreurs est identique pour bayes et Kppv.

Tests de discrimination sur un ensemble pré-classé par Coalescence

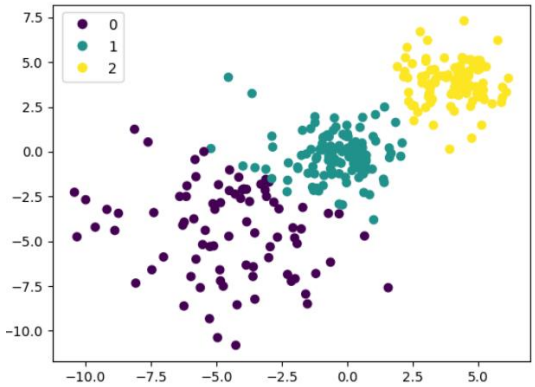
k	1	3	5	7	13	15
Taux d'erreurs avec kppv	6.66%	6.0%	5.0%	5.0%	5.33%	5.66%

Jugurtha ASMA et Hylia BOUDAHBA



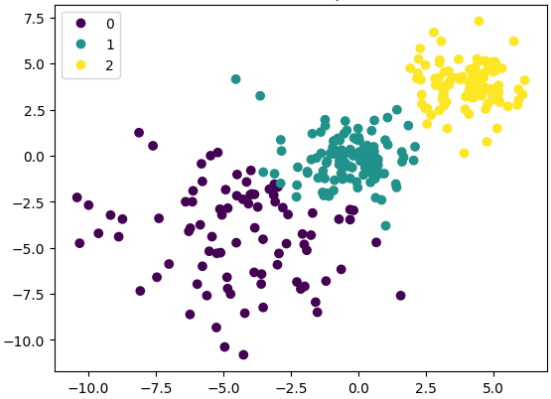
Courbe représentant le taux d'erreurs pour l'algo de Kppv et bayes en fonction de k

Classification par bayes naïf



Classification pour un ensemble pré-classé par coalescence

Classification KNN pour k = 5



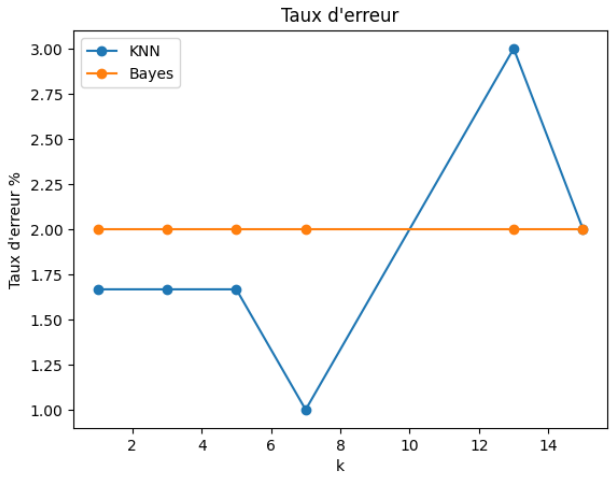
Pour bayes, on a un taux d'erreur de 5%, donc identique pour k = 5 et k = 7 avec l'algorithme de Kppv. Pour toutes les autres valeurs de K, bayes a un taux d'erreurs plus faible que Kppv. On peut expliquer visuellement ces erreurs par le fait qu'avec l'algorithme Kppv, des données de la classe 0 sont dans la classe 0 au lieu d'être dans la classe 1 comme le montre l'algorithme de bayes qui est plus précis.

Par rapport au test précédent, la différence entre l'utilisation d'un ensemble pré-classé par coalescence est entre 2 à 3 fois plus grande. Comme on peut le voir sur le graphique ci-dessus, l'utilisation d'un ensemble pré-classé est moins efficace que les tests précédents faits.

Tests de discrimination sur un ensemble ne suivant pas une distribution Gaussienne

k	1	3	5	7	13	15
Taux d'erreurs avec kppv	1.66%	1.66%	1.66%	1.0%	3.0%	2.0%

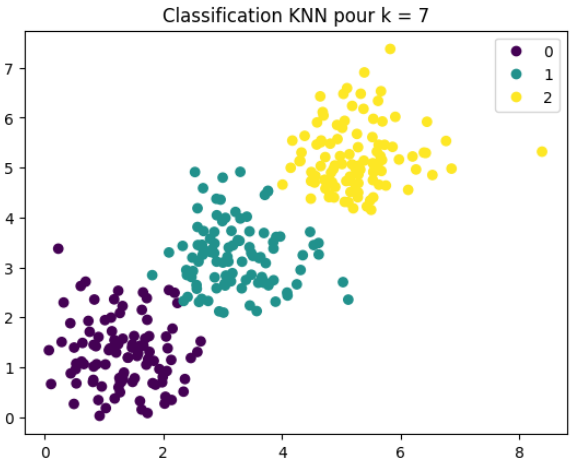
Le taux d'erreur avec bayes est de 2.0%.



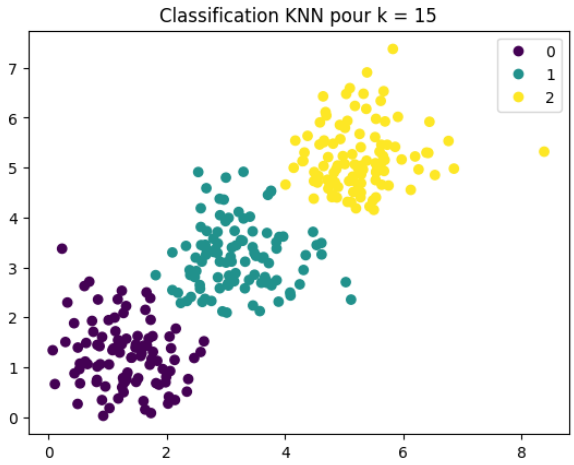
Comme on peut le voir, lorsque l'on n'utilise pas la distribution Gaussienne, bayes naïf n'a pas les meilleurs résultats comparés à Kppv. Pour K compris entre 1 et 9 exclus, l'algo Kppv a un taux d'erreur plus faible que celui de bayes. Pour k =7, le taux d'erreur est de 2% pour bayes naïf et de 1% pour Kppv.

En revanche pour K compris entre 9 exclus et 14 exclus, bayes naïf a un taux d'erreur plus faible que kppv. Soit pour k = 13 le taux d'erreur est de 3% pour kppv et de 2% pour bayes naïf.

C'est assez surprenant étant donné que l'on a utilisé la gaussienne



Classification pour un ensemble ne suivant pas une distribution Gaussienne avec le taux d'erreur le plus faible avec Kppv



Classification pour un ensemble ne suivant pas une distribution Gaussienne avec le taux d'erreurs le moins fort avec Kppv

Bilan

D'après tous les tests faits précédemment, on peut conclure que si un ensemble d'individus utilisent une distribution gaussienne, il serait plus judicieux d'utiliser l'algorithme de bayes naïf version gaussienne. Pour les algorithmes de discriminations des valeurs trop grandes ou trop petites de K ne donnent pas de résultats satisfaisants. Si k est trop petit, knn sera sensible au bruit et si k est trop grand, le voisinage pourrait inclure des points d'autres classes. Mais on a remarqué que l'algorithme de discrimination utilisant des grandes tailles nous donnaient des taux d'erreurs plus faible que celui avec les petites tailles, donc à privilégier.

Volet chargement des descripteurs

Dans un premier temps, nous avons récupérer les 5 ensembles de mesures donnés lors du TP, et chacun de ses vecteurs de mesures correspondent à des images. Chaque image est stockée dans le dossier Wang et porte un numéro allant de 0 à 999.

On va classer ces différentes images par leur numéro d'image, qu'on peut appeler des étiquettes. On va procéder de la manière suivante : prendre les images entre [0 et 99] et les diviser par 100 qui donnera des valeurs comprises entre 0 et 0 ,99 donc on aura notre première catégorie qui sera désigné la catégorie numéro 0. Puis on va passer pas les images numérotées entre [100 et 199] divisé par 100 qui sera donc des valeurs comprises entre 1 et 1 ,99, on aura la catégorie numéro 1. Puis on répète le même procédé pour toutes les images.

On aura donc les classes suivantes : 0,1,2,3,4,5,6,7,8,9.

On a ensuite divisé en deux ensembles nos données : un ensemble de donnée d'apprentissage et un ensemble de tests. Les données d'apprentissage sont celles qui présentent déjà la valeur réelle que le modèle aurait dû prédire. L'algorithme va donc modifier la valeur des paramètres pour tenir compte des données de l'ensemble d'apprentissage.

Jugurtha ASMA et Hylia BOUDAHBA

Nous avons divisé les données grâce à la fonction train_test_split de python, elle prend 4 variables en sortie que nous avons nommé : X_train (Données de formation en entrée), X_test (Données de test en entrée), Y_train (Données de formation en sortie) et Y_test (Données de test en sortie)

On utilise aussi un autre paramètre random_state qui permet d'initialiser le générateur de nombres aléatoires interne. Il permet aussi de décider pour la division des données en valeurs de tests et d'apprentissage. La valeur de formation est de 80% des données de bases et la valeur de test est de 20% des donnés originales.

Nous avons fait une fonction findRowNumber qui selon le numéro de l'image nous renvoie la ligne dans l'Excel.

Classification

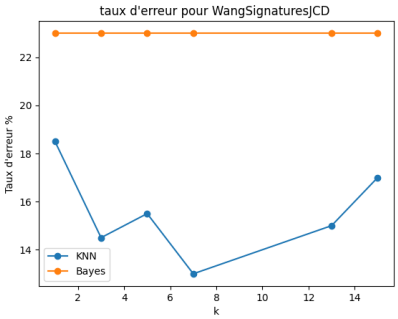
Descripteurs	JCD	PHOG	CEDD	FCTH	ColorHistogr	Cumul des mesures
Bayes	23 %	47 %	21 %	24%	36%	18%

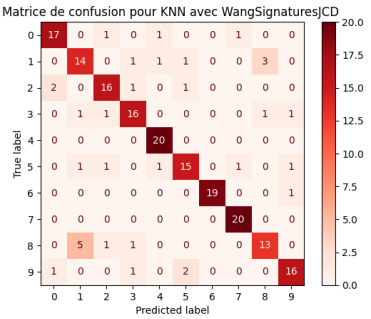
Résultat du taux d'erreur avec bayes pour chacune des discriminations

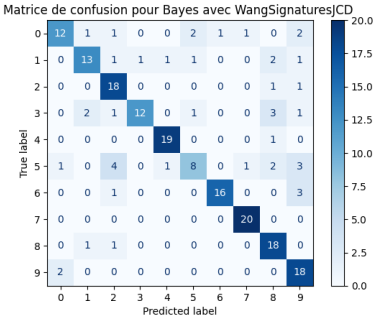
Comme on peut le voir sur notre tableau de comparaison des taux d'erreurs avec bayes naïf ci-dessus,

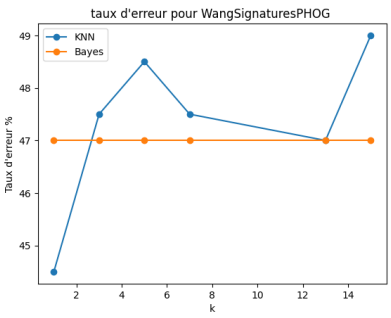
CEDD a le plus faible taux d'erreur, soit 21% parmi tous les discriminants. JCD et FCTH qui ne sont pas loin de CEDD également avec un taux de 23% et 24%. ColorHistogr qui lui a un taux d'erreur de 36%. En revanche PHOG a le taux d'erreurs le plus élevé qui est de 47%.

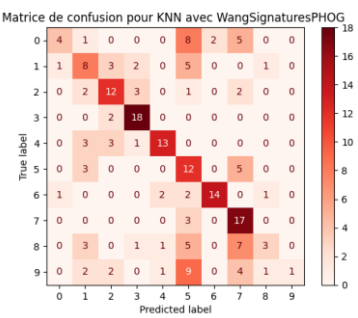
Vous trouverez ci-joint les différents graphiques du taux d'erreurs en fonction de k, qui vaut 15 pour nos 5 descripteurs, ainsi que les différentes matrices de confusion avec les algorithmes de discrimination knn et bayes.

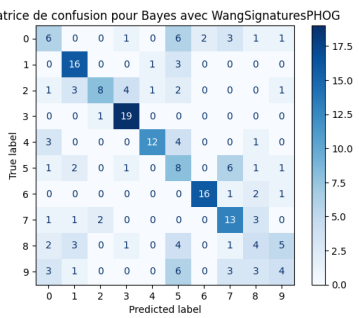


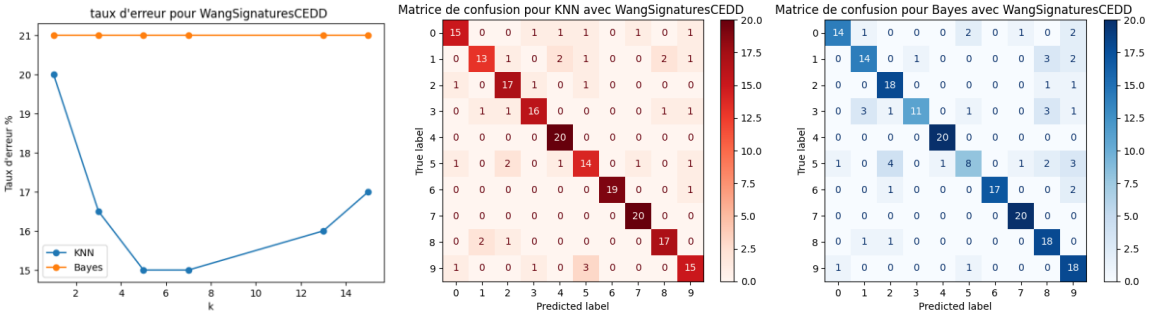


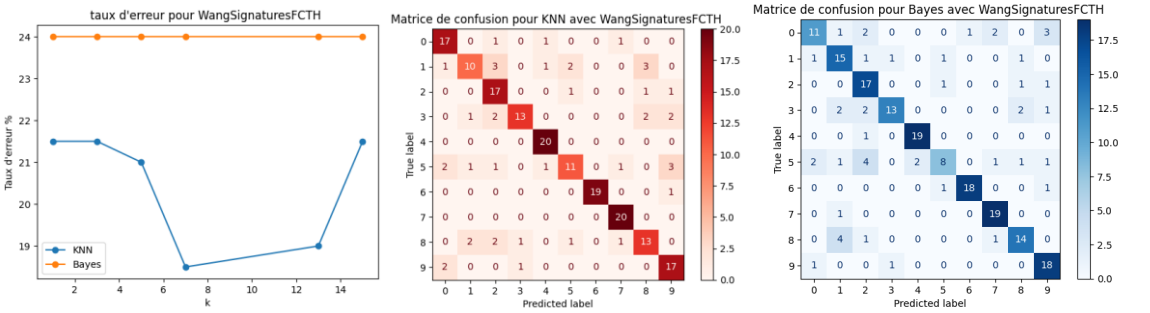


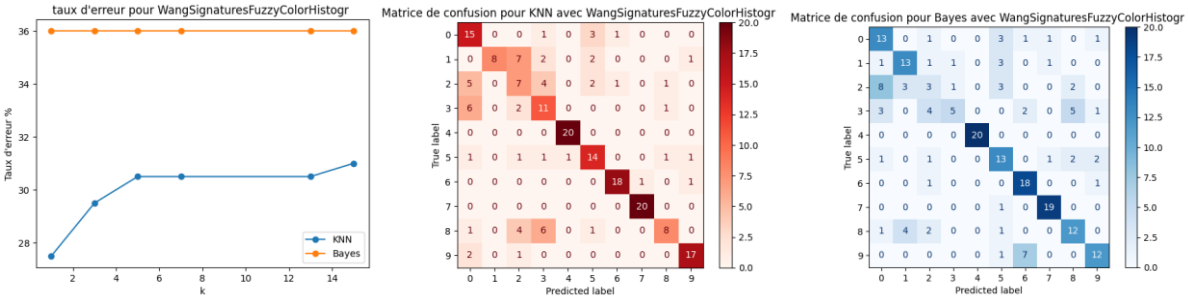






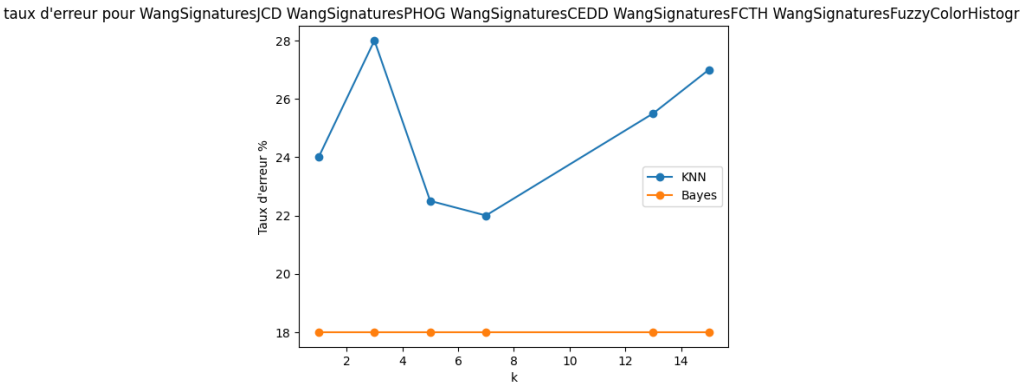




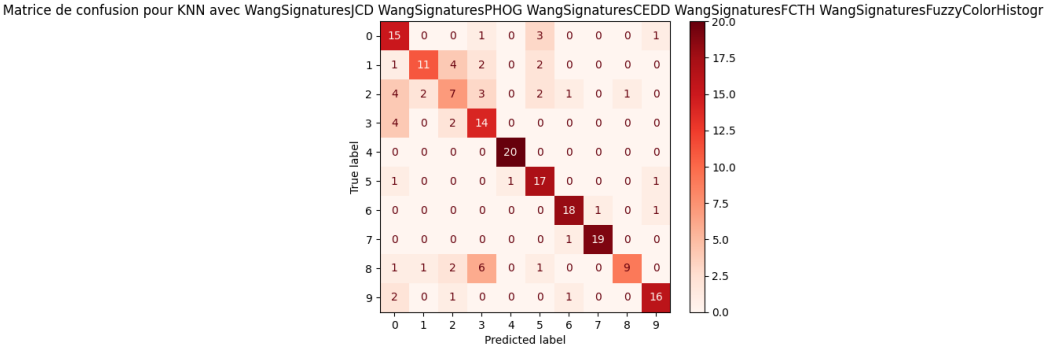


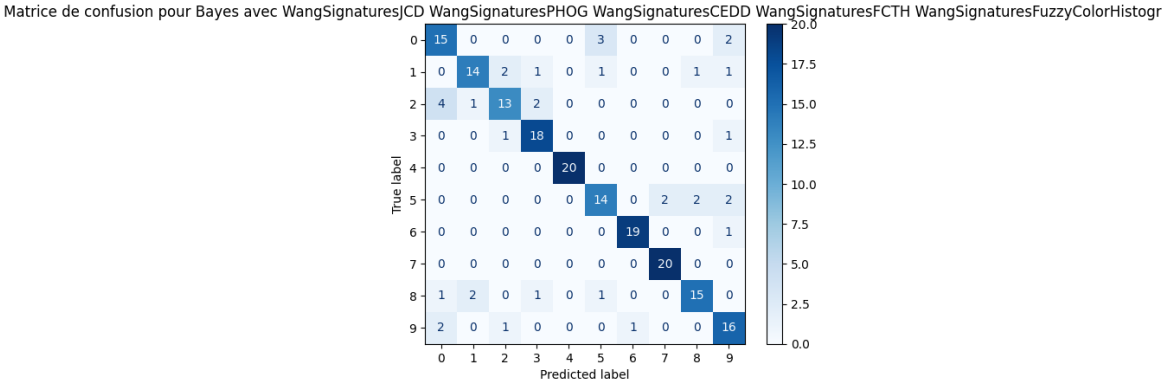
Nous avons remarqué qu'en prenant la meilleure configuration de K, l'algorithme de Kppv a le meilleur résultat donc un taux d'erreur le plus faible.

Ci-dessous le diagramme du cumul de toutes les mesures ainsi que les matrices de confusion avec kppv et bayes. On peut remarquer que le taux d'erreurs est plus faible avec bayes lorsqu'on prend en compte le cumul.



Jugurtha ASMA et Hylia BOUDAHBA





Pour la dernière partie l'utilisateur pourra tester avec ses saisies :

Le numéro de l'image (ex : 155 pour l'image 155.jpg)

Le descripteur qu'il veut utiliser (ex : JCD)

La technique de son choix soit KNN ou bayes (ex : KNN)

Pour KNN, choisir après le nombre des plus proches voisins k (ex : 7)

```
taux d'erreur =  15.0 %
rowNumber =  24
L'image 12 est prédite comme appartenant à la classe Jungle

<matplotlib.image.AxesImage at 0x1405198e040>
```

Exemple de résultat obtenu avec l'image 12, avec JCD, knn et k = 5, on a un taux d'erreur de 15% ce qui est bien et l'image a bien été prédite comme convenu dans la classe Jungle.