IA : Classification Challenge, Algorithme K-nn

Plan d'action

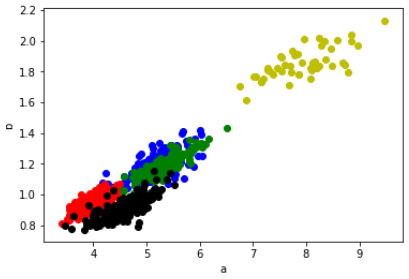
- *Première étape : « data.csv »*

  La première étape de ce projet a été d'adapter le code du K-nn réalisé sur le TD des Iris. Cependant une première difficulté a été rencontrée lorsque j'ai commencé à adapter la classe de l'objet individu, que j'utilise pour manipuler les données. En effet, dans le nouveau fichier data.csv, les données étaient anonymisées. On ne pouvait donc pas savoir ce qu'elles représentaient, ni comment les manipuler. J'ai donc pensé premièrement à tracer l'ensemble des différentes combinaisons de deux variables possibles, afin de déterminer quelles combinaisons permettraient au mieux de déterminer les différents labels. Voici, ci-dessous-le code afin d'effectuer cette première analyse, ainsi que les différents graphiques de deux variables et ceux les plus exploitables :

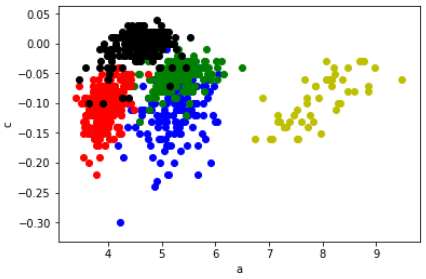
```
def AnalyseGraphiques(population,x,y):

    Ax=[]
    Ay=[]
    Bx=[]
    By=[]
    Cx=[]
    Cy=[]
    Dx=[]
    Dy=[]
    Ex=[]
    Ey=[]
    for e in population:
    if(e.label=="classA"):
            Ax.append(e.x)
            Ay.append(e.y)
            else:
                    if(e.label=="classB"):
                            Bx.append(e.x)
                            By.append(e.y)
                    else:
                            if(e.label=="classC"):
                                    Cx.append(e.x)
                                    Cy.append(e.y)
                            else:
                                    if(e.label=="classD"):
                                            Dx.append(e.a)
                                            Dy.append(e.b)
                                    else:
                                            if(e.label=="classE"):
                                                    Ex.append(e.a)
                                                    Ey.append(e.b)
```

```
plt.scatter(Ax,Ay,color='b')
plt.scatter(Bx,By,color='g')
plt.scatter(Cx,Cy,color='r')
plt.scatter(Dx,Dy,color='k')
plt.scatter(Ex,Ey,color='y')
plt.xlabel(x)
plt.ylabel(y)
plt.show()
```

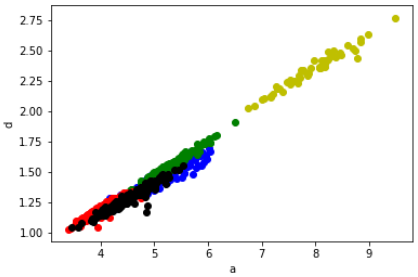
Les différents graphiques obtenus :



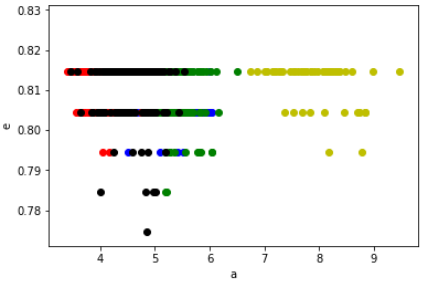
a en fonction de b



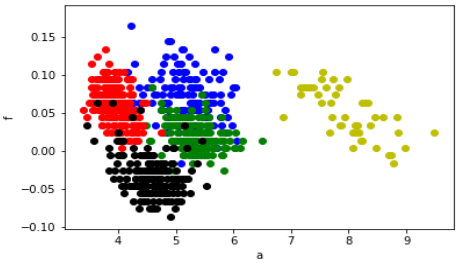
a en fonction de c



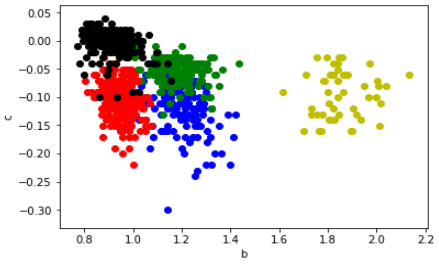
a en fonction de d



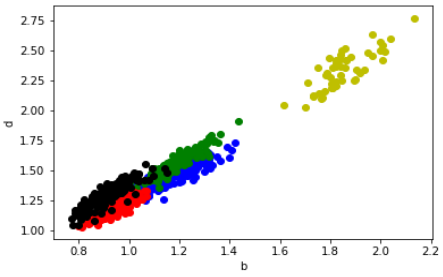
a en fonction de e



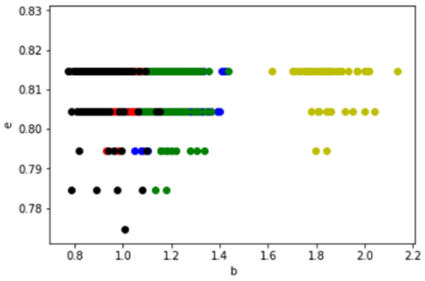
a en fonction de f



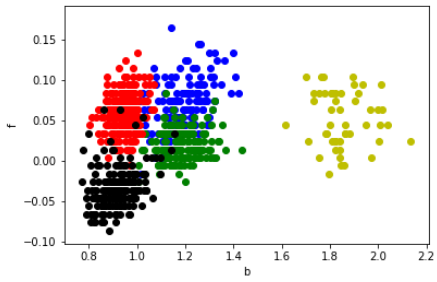
b en fonction de c



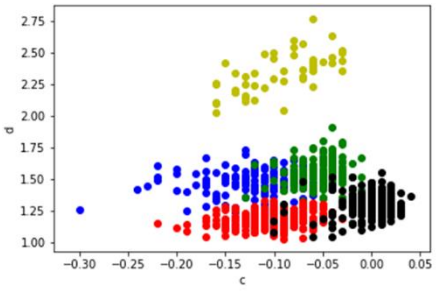
b en fonction de d



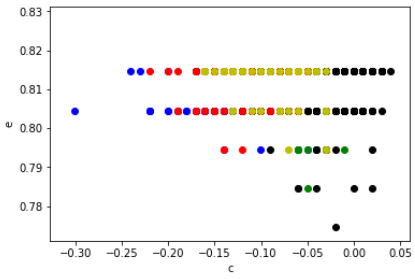
b en fonction de e



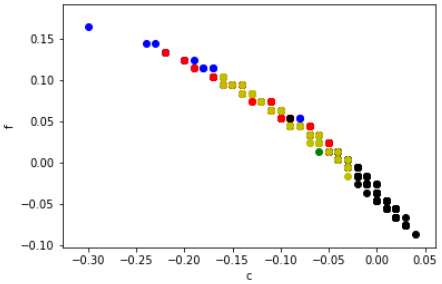
b en fonction de f



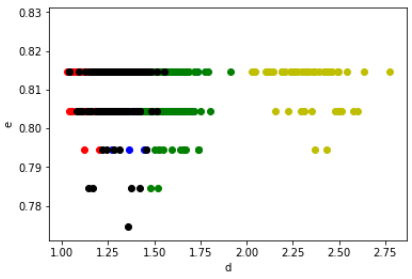
c en fonction de d



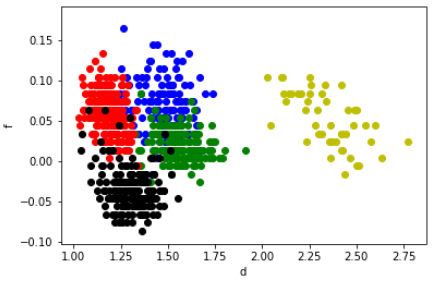
c en fonction de e



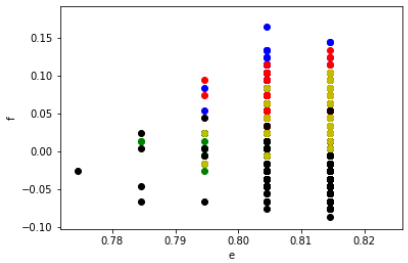
c en fonction de f



d en fonction de e



d en fonction de f

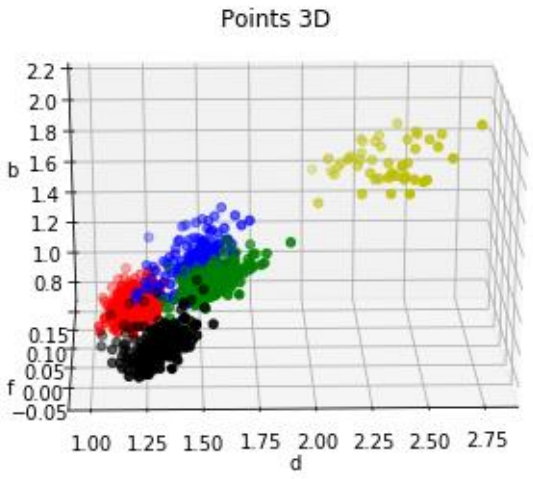


e en fonction de f

J'ai pu, grâce aux tracés de l'ensemble des combinaisons de variables possibles, mettre en évidence que les combinaisons des variables (a ; c), (a ; f), (b ; c), (b ; f) ; (c ; d) et (d ; f) permettaient de distinguer les classes par leurs nuages de points respectifs. J'ai pris la décision d'adapter le code de mon K-nn en calculant 6 distances euclidiennes en fonction des 6 combinaisons mises en évidence ci-dessus, et d'ensuite retourner le label le plus présent dans les 6 différentes listes de plus proches voisins construites à l'aide des 6 distances euclidiennes respectives. Suite au développement de ces différentes méthodes, j'ai pu obtenir entre 85% et 90% de réussite sur les différents « runs » sur le fichier « data.csv ».

En souhaitant augmenter ce pourcentage de réussite et diminuer les potentielles erreurs de prédiction, j'ai pu constater que la majorité des erreurs étaient dûes la plupart du temps à la confusion entre les labels : « classA » et « classB ». En effet, de nombreux points des deux classes se trouvent dans le nuage de la classe voisine sur les différents graphiques explorables ci-dessus. Je me suis ainsi intéressé aux différentes combinaisons de trois variables afin d'obtenir des graphiques en 3 dimensions et éventuellement obtenir un décalage dans l'espace qui pourrait permettre d'améliorer la différenciation des deux classes et ainsi d'augmenter la précision du modèle. Cependant, à la suite des tracés des graphes, aucune combinaison de 3 variables ne créées suffisamment de décalage dans l'espace pour permettre la différenciation des deux classes.

Un exemple d'une analyse graphique en 3D de la combinaison (b ; d ; f) :



Je suis donc revenu après à la méthode sans doute la plus basique et la plus efficace qui consistait à calculer directement la distance euclidienne des différents voisins sur l'ensemble des 6 paramètres à l'aide de la fonction suivante :

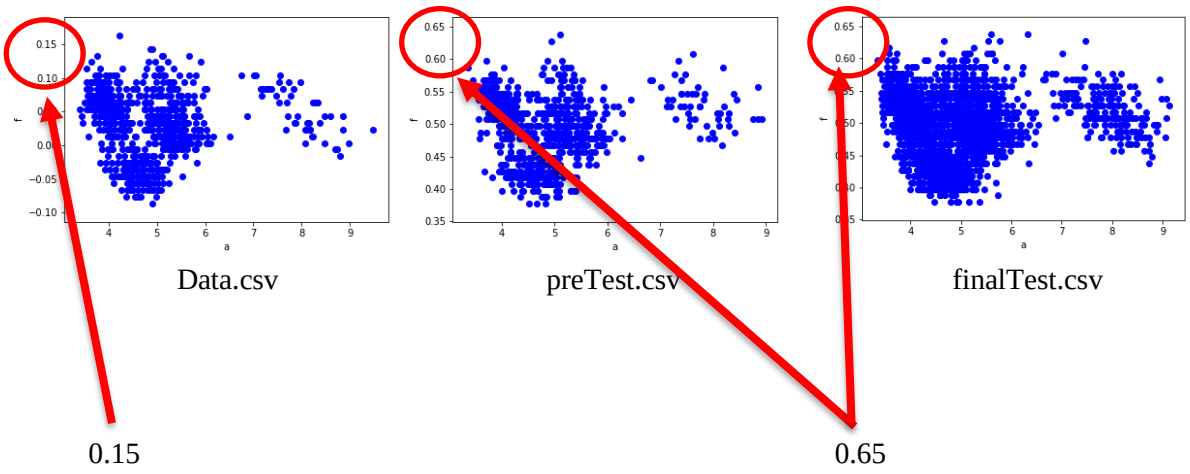
$$D = \sqrt{\sum_{a}^{f}(individu1.i - individu2.i)^2}$$

(Avec les i étant les paramètres allant de a à f des objets de la classe Individu, permettant de manipuler les données des fichiers)

J'ai pu obtenir à l'aide de ce modèle des résultats avoisinant les 90% de réussite dans la majorité des « runs » en prenant 80% des données du fichier « data.csv » pour le training et 20% pour les tests.

- *« preTest.csv » & « finalTest.csv »*

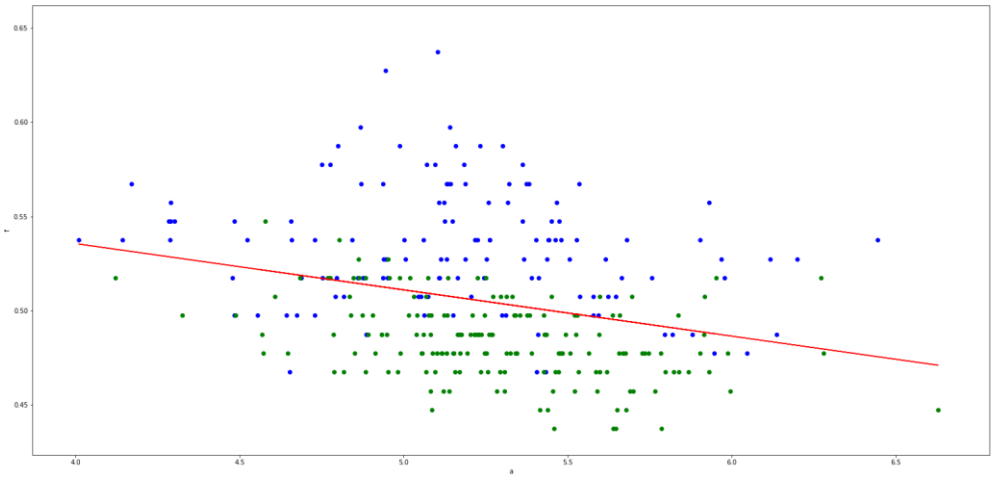
  En obtenant le deuxième fichier de test ainsi que le fichier de données finales à prédire, j'ai commencé par chercher à identifier les similitudes et les différences par rapport au fichier « data.csv ». En traçant les nuages de points, j'ai remarqué que les données des deux derniers fichiers étaient translatées de 0,5 par rapport aux données du premier fichier de données (voir ci-dessous).



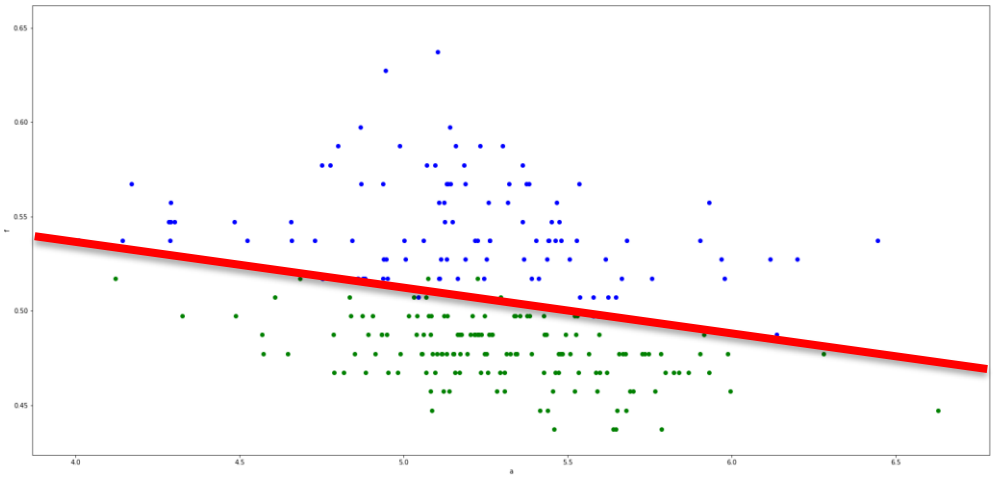
On voit ici la translation de +0.5 sur les combinaisons des caractéristiques (a ; f). J'en ai déduit qu'il était inutile d'utiliser le fichier « data.csv » pour les prédictions finales, et que la combinaison des deux jeux de données d'entrainement n'était pas nécessaire.

Remarque : Il y a de plus 3000 données dans le dernier fichier, c'est pour cela que le nuage est plus intense que les deux autres fichiers n'ayant que ≈800 données.

Enfin, étant donné de la confusion entre les classes A et B, les erreurs de prédictions sont encore d'une dizaine de pourcent sur l'ensemble du fichier « preTest.csv ». J'ai donc eu l'idée d'essayer un traitement des données du deuxième fichier d'entrainement en nettoyant les points de la classe A dans le nuage de la classe B et des points de cette dernière dans la classe A à l'aide du tracé de la droite de régression entre les deux nuages de points. Le principe est donc de vérifier si les points sont au-dessus ou en dessous de cette droite et en fonction de leur classe, et de les supprimer en les considérant comme valeur aberrante nuisible au modèle de prédiction. On peut donc voir une illustration ci-dessous montrant les nuages avant et après nettoyage pour les paramètres (a ; f) :



On peut voir sur ce schéma la droite de régression et certains points bleus polluant le nuage de points verts, ainsi que des points verts polluant le nuage bleu. En obtenant les coefficients caractéristiques de la droite, nous pouvons vérifier si un point est au-dessus ou en dessous et le « remove » de la liste si besoin. Après nettoyage, j'ai donc diminué la présence de voisins potentiellement trompeur dans les nuages des deux classes comme nous pouvons le voir ci-dessous :



On voit donc qu'il y a beaucoup moins de points bleus dans la zone verte et beaucoup moins de points verts dans la zone bleue.

Après le nettoyage, j'ai donc relancé mon algorithme sur une centaine de « runs » afin de récupérer les statistiques de mon nouveau modèle et dans le but de trouver le « k » le plus pertinent. J'ai donc eu des résultats bien plus précis descendant même parfois a 3 ou 4 % d'erreur sur certains essais. J'ai gardé la valeur de k=3, étant dans 44% des cas le k le plus efficace pour mon modèle, avec en moyenne 7.86% d'erreur sur les différents essais.

J'ai donc conclu que le nettoyage des données par régression pouvait améliorer le modèle, mais qu'en supprimant certaines valeurs, cela pouvait sans doute augmenter le risque d'erreur sur d'autres classes. De plus, pour les valeurs tangentes entre deux classes, l'algorithme aura toujours des difficultés pour déterminer leur label. Ainsi il est extrêmement difficile de faire tendre le taux d'erreur vers 0%.

Je vous remercie pour l'attention porté à ce rapport,

PUJOL Corentin, TD K.